Léonard Brice, Mathieu Sassolas
leonard.brice@ulb.be, mathieu.sassolas@ulb.be

Session 5:
NP-completeness and polynomial-time reduction

Solutions

1 NP-complete graph problems

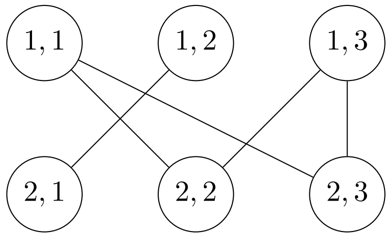
1. *Explain briefly why the two definitions of* NP *are equivalent.*

   ☛ If a problem can be decided by a non-deterministic Turing machine in polynomial time, then accepting computations of that machine form a notion of certificate that has polynomial size and that can be checked in polynomial time. Conversely, if such a notion of certificate exists, then a non-deterministic Turing machine can, in polynomial time, guess that certificate, and check it.

2. *Show that* Sat *is reducible in polynomial time to the problem:*

$$\textsc{Clique} : \begin{cases} \textit{Input:} & \textit{A graph } G \textit{ and an integer } k \\ \textit{Question:} & \textit{Is there a clique of size } k \textit{ in } G\textit{?} \end{cases}$$

   ☛ Let $\varphi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \ell_{ij}$ be a Sat instance. Let us define an instance $(G, k) = f(\varphi)$ of Clique. The graph G is defined as follows: the vertices are the pairs (i, j), with $i \in \{1, \dots, n\}$, and $j \in \{1, \dots, m_i\}$. There is an edge between the vertex (i, j) and the vertex (i', j') if and only if $i \neq i'$ and $\ell_{ij} \neq \bar{\ell}_{i'j'}$. Moreover, we define $k = n$. Thus, the pair (G, k) can be constructed in polynomial time from φ. Here is the graph G when $\varphi = (x \vee y \vee x) \wedge (\neg x \vee \neg y \vee \neg y)$.



   Note that there are five cliques of size 2 in that graph (one for each edge).

   Let us now show that G and k form a positive instance of Clique if and only if φ is a positive instance of 3Sat, i.e. if φ is satisfiable, then let ν be a valuation that satisfies it. For each i, the clause $\ell_{i1} \vee \cdots \vee \ell_{im_i}$ is satisfied by ν: let j_i be an index such that the literal ℓ_{ij_i} is satisfied. Then, the set $\{(1, j_1), \dots, (n, j_n)\}$ is a clique of size $k = n$ in G: if there were two vertices (i, j_i) and $(i', j_{i'})$ not linked by an edge, it would mean that $\ell_{ij_i} = \bar{\ell}_{i'j_{i'}}$, which is impossible since both ℓ_{ij_i} and $\ell_{i'j_{i'}}$ are satisfied by ν.

   Conversely, if there is a clique in G, then necessarily it contains at most one vertex of the form (i, j) for each i, since two vertices (i, j) and (i, j') are never linked. Therefore, if there is a clique of size $k = n$ in G, then it contains exactly one such vertex for each i: let us write them $(1, j_1), \dots, (n, j_n)$. Let now ν be the valuation defined by $\nu(x) = 1$ if and only if there exists i such that $\ell_{ij_i} = x$. Then, for each clause $\ell_{i1} \vee \cdots \vee \ell_{im_i}$, the literal ℓ_{ij_i} is satisfied by ν: indeed, if it is a positive literal $\ell_{ij_i} = x$, then $\nu(x) = 1$, and if it is a negative literal $\ell_{ij_i} = \neg x$, then there is no i' such that $\ell_{i'j_{i'}} = x$, otherwise the vertices (i, j) and $(i', j_{i'})$ would not be linked, which is impossible in a clique; and since there is no such i', we have $\nu(x) = 0$, as desired.

3. *Prove* NP*-hardness for the problem:*

$$\text{IndependentSet} : \begin{cases} \textit{Input:} & \textit{A graph } G \textit{ and an integer } k \\ \textit{Question:} & \textit{Is there an independent set}^{1} \textit{ of size } k \textit{ in } G\textit{?} \end{cases}$$

☛ By the two previous questions, the problem Sat reduces in polynomial time to the problem Clique, which is therefore NP-hard. Let us therefore prove the NP-hardness of IndependentSet by reduction from Clique.

Let G and k form an instance of Clique. Let us define an instance $(G', k') = f(G, k)$ of IndependentSet: we define $k' = k$, and G' is the graph that has the same vertices as G, and in which for every two vertices u and v, there is an edge between u and v if and only there is no such edge in G. Thus, G' and k' have been constructed in polynomial time from G and k, and clearly, they form a positive instance of IndependentSet if and only if G and k form a positive instance of Clique.

Therefore, Clique reduces in polynomial time to IndependentSet, and if the former is NP-complete, then so is the latter.

4. *Prove* NP*-completeness for the problem:*

$$SI : \begin{cases} \textit{Input:} & \textit{Two graphs } G \textit{ an } H \\ \textit{Question:} & \textit{Is there an isomorphism}^{2} \textit{ from } H \textit{ to a subgraph of } G\textit{?} \end{cases}$$

☛ Let us show, first, that this problem is NP-easy, and then that it is NP-hard.

- *Easiness.* A valid notion of certificate for that problem is a mapping ι from the vertex set of H to the vertex set of G, which can be expressed by the list of pairs $(v, \iota(v))$, whose size is polynomial in the sizes of H and G. If there exists such a mapping that is an isomorphism, then G and H form a positive instance of SubgraphIsomorphism; and given ι, checking that it is an isomorphism can be done in polynomial time.
- *Hardness.* We proceed by reduction from the NP-hard problem Clique. Let (G, k) form an instance of Clique. Then, let us define an instance $(G', H) = f(G, k)$ of SubgraphIsomorphism. We define $G' = G$, and H as a clique of size k. Then, the graph G' contains a subgraph that is isomorphic to H if and only if the graph G contains a clique of size k.

  Is that reduction made in polynomial time? **Be careful:** the graph H has size k, which is exponential in the size of k... But we can assume that the graph G has at least k vertices (otherwise both problems are trivial). Then, the construction of H can be done in a time that is polynomial not in the size of k, but in the size of G.

  Thus, the NP-hard problem Clique reduces in polynomial time to the problem SubgraphIsomorphism, which is therefore also NP-hard.

2 Applications

In your opinion, how hard are the following computer-related problems? An intuitive proof is sufficient.

1. *Given a Python program P, and an index i, is the line i dead code*[3]*?*

   ☛ That problem is undecidable, by reduction from the halting problem. Indeed, given a Turing machine M and a word w, the Python program:

[1]i.e., a set of vertices that are not linked by any edge

[2]i.e., a bijection ι from the vertex set of H to the vertex set of G such that each pair $\{u, v\}$ is an edge in H if and only if $\{\iota(u), \iota(v)\}$ is an edge in G

[3]Dead code is code that is never executed: for example, in the code `if` $0 = 1$ `then return` 0, the instruction `return` 0 is dead code

```
┌──────────────────────┐
│ (Simulate $M$ on $w$) │
└──────────────────────┘
return 0
```

is such that the last line is dead code if and only if M halts on w.

2. *Given a Python program P (without arguments) and a maximal duration D, will P last less than D?*

   ☛ If D is expressed as a number of basic operations, a simple algorithm that solves it is to run P for that duration and to check whether it stops or not. If D is expressed in binary, that algorithm is exponential. No better algorithm is known, and that problem is actually EXP-complete — but the proof is beyond the scope of this course.

3. *Given two Facebook users Alice and Bob, do Alice and Bob have a common friend*[4]*?*

   ☛ As a first approximation, we can assume that Facebook is a big adjacency list (with a few additional features). Then, searching for a common friend of Alice and Bob amounts to searching for a vertex that is in the neighborhood of both corresponding vertices, which can be done in linear time.

4. *Given an integer k, does there exist a group of k Facebook users who are all friends with each other?*

   ☛ Small trap here: if the graph was included in the instance, then this would be more or less a translation of the problem CLIQUE. But when the graph G is fixed, the problem:

$$\textsc{Clique}_G : \begin{cases} \textit{Input:} & \text{An integer } k \\ \textit{Question:} & \text{Does there exist a clique of size } k \text{ in } G? \end{cases}$$

   is not NP-complete: in the fixed graph G, there exists a maximal number K such that there is a clique of size K, and this problem is simply equivalent to decide whether $k \leq K$, which can be done in a time that is linear in the size of k.

   In practice, that means that the problem given in the question is theoretically easy, for someone *prepared* to it: if we pre-compute the size of the biggest group of friends in Facebook, and update it each time the friends relation is changed, then we can answer the question almost immediately when asked. But if that preliminary work is not done, then the problem is actually hard (if k is big).

5. *Given a number m of processors (each of them being able to achieve one task at once), a set T of tasks (each $t \in T$ requiring a duration of $d(t) \in \mathbb{N}$ time units), and a maximal duration D (expressed in time units), is it possible to distribute those tasks between the processors so that they can achieve them in D time units?*

   ☛ This is a NP-hard problem (if the durations are given in binary). Indeed, let us call it MULTIPROCESSORSCHEDULING, first, that it is NP-easy, and then that it is NP-hard.

   - *Easiness.* A valid notion of certificate for that problem is a partition $T_1 \cup \cdots \cup T_m$ of T. If there exists such a partition satisfying $\sum_{t \in T_i} d(t) \leq D$ for each i, then m, T, d and D form a positive instance of MULTIPROCESSORSCHEDULING; and given $T_1 \cup \cdots \cup T_m$, checking that $\sum_{t \in T_i} \ell(t) \leq D$ for each i can be done in polynomial time.
   - *Hardness.* We proceed by reduction from the NP-hard problem PARTITION. Let S form an instance of PARTITION. Then, let us define an instance $(m, T, \ell, D) = f(S)$ of MULTIPROCESSORSCHEDULING. We define $m = 2$, $T = S$, $\ell = \mathrm{id}_{\mathbb{N}}$, and $D = \frac{1}{2} \sum_{s \in S} s$. Then, there is a partition $T = T_1 \cup T_2$ with $\sum_{t \in T_1} \ell(t), \sum_{t \in T_2} \ell(t) \leq D$ if and only if there is a partition $S = S_1 \cup S_2$ with $\sum_{s \in S_1} s = \sum_{s \in S_2} s$.

     Thus, the NP-hard problem PARTITION reduces in polynomial time to the problem MULTIPROCESSORSCHEDULING, which is therefore also NP-hard.

[4] In Facebook's definition of the term