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

Session 5:
NP-completeness and polynomial-time reduction

Reminders

The class P is the class of the problems that are decided by a deterministic Turing machine that runs in polynomial time.

Similarly, the class NP is the class of the problems that are decided by a non-deterministic Turing machine that runs in polynomial time. Equivalently, a problem A over Σ^* belongs to NP if and only if there exists a polynomial p, a set $C \subseteq \Gamma^*$ of objects, called *certificates*, and a mapping $\varphi : \Sigma^* \times C \to \{\text{True}, \text{False}\}$ such that:

- an instance i of A is positive if and only if there exists a certificate $c \in C$ with $|c| \leq p(|i|)$ such that $\varphi(i, c)$ is true;
- given i and c, deciding whether $\varphi(i, c)$ is true can be done in polynomial time.

A *polynomial-time reduction* from a problem A to a problem B is a mapping from the instance set of A to the instance set of B that is computable in polynomial time. Then, if B belongs to P (resp. NP), then so does A.

We say that a problem A is NP-*easy* when it belongs to NP. We say that it is NP-*hard* when every NP-easy problem B reduces to A in polynomial time. Thus, if there exists a polynomial time reduction from an NP-hard problem A to some other problem C, then C is also NP-hard.

Finally, we say that A is NP-*complete* when it is both NP-easy and NP-hard.

According to *Cook-Levin theorem*, the problem Sat:

$$\textsc{Sat} : \begin{cases} \textit{Input:} & \text{A propositional logic formula } \varphi, \text{ written in conjunctive normal form} \\ \textit{Question:} & \text{Is } \varphi \text{ satisfiable?} \end{cases}$$

is NP-complete. So are the problem:

$$\textsc{SubsetSum} : \begin{cases} \textit{Input:} & \text{A set } S \subseteq \mathbb{N} \text{ and an integer } t \\ \textit{Question:} & \text{Is there a subset } S' \subseteq S \text{ such that } \sum_{s \in S'} s = t \end{cases}$$

and the problem:

$$\textsc{Partition} : \begin{cases} \textit{Input:} & \text{A set } S \subseteq \mathbb{N} \\ \textit{Question:} & \text{Is there a partition } S = S_1 \cup S_2 \text{ such that } \sum_{s \in S_1} s = \sum_{s \in S_2} s? \end{cases}$$

1 NP-complete graph problems

1. Explain briefly why the two definitions of NP are equivalent.
2. Show that Sat is reducible in polynomial time to the problem:

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

3. Prove NP-hardness for the problem:

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

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

4. Prove NP-completeness for the problem:

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

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]?
2. Given a Python program P (without arguments) and a maximal duration D, will P last less than D?
3. Given two Facebook users Alice and Bob, do Alice and Bob have a common friend[4]?
4. Given an integer k, does there exist a group of k Facebook users who are all friends with each other?
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?

[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

[4] In Facebook's definition of the term