Intermezzo: Complexity Theory

- mathematical framework to study the difficulty of algorithmic problems

Notations/Definitions

- problem: generic description of a problem (e.g. $1||\sum C_j$)
- instance of a problem: given set of numerical data (e.g. n, $p_1, \dots, p_n$)
- size of an instance I: length of the string necessary to specify the data (Notation: $|I|$)
  - binary encoding: $|I| = n + \log(p_1) + \ldots + \log(p_n)$
  - unary encoding: $|I| = n + p_1 + \ldots + p_n$

Notations/Definitions

- efficiency of an algorithm: upper bound on number of steps depending on the size of the instance (worst case consideration)
- big O-notation: for an $O(f(n))$ algorithm a constant $c > 0$ and an integer n_0 exist, such that for an instance I with size $n = |I|$ and $n \geq n_0$ the number of steps is bounded by $cf(n)$
  Example: $7n^3 + 230n + 10\log(n)$ is $O(n^3)$
- (pseudo)polynomial algorithm: $O(p(|I|))$ algorithm, where p is a polynomial and I is coded binary (unary)
  Example: an $O(n\log(\sum p_j))$ algorithm is a polynomial algorithm and an $O(n\sum p_j)$ algorithm is a pseudopolynomial algorithm

Classes $\mathcal{P}$ and $\mathcal{NP}$

- a problem is (pseudo)polynomial solvable if a (pseudo)polynomial algorithm exists which solves the problem
- Class $\mathcal{P}$: contains all decision problems which are polynomial solvable
- Class $\mathcal{NP}$: contains all decision problems for which - given an 'yes' instance - the correct answer, given a proper clue, can be verified by a polynomial algorithm

Remark: each optimization problem has a corresponding decision problem by introducing a threshold for the objective value (does a schedule exist with objective smaller k?)

Polynomial reduction

- a decision problem P polynomially reduces to a problem Q, if a polynomial function g exists that transforms instances of P to instances of Q such that I is a 'yes' instance of P if and only is $g(I)$ is a 'yes' instance of Q
  Notation: $P \propto Q$

NP-complete

- a decision problem $P \in \mathcal{NP}$ is called NP-complete if all problems from the class $\mathcal{NP}$ polynomially reduce to P
- an optimization problem is called NP-hard if the corresponding decision problem is NP-complete

Examples of NP-complete problems:

- SATISFIABILITY: decision problem in Boolean logic, Cook in 1967 showed that all problems from $\mathcal{NP}$ polynomially reduce to it
- PARTITION:
  - given n positive integers $s_1, \ldots, s_n$ and $b = 1/2 \sum_{j=1}^{n} s_j$
  - does there exist a subset $J \subset I = \{1, \ldots, n\}$ such that

$$\sum_{j \in J} s_j = b = \sum_{j \in I \setminus J} s_j$$

Examples of NP-complete problems (cont.):

- 3-PARTITION:
  - given $3n$ positive integers $s_1, \ldots, s_{3n}$ and b with $b/4 < s_j < b/2, \; j = 1, \ldots, 3n$ and $b = 1/n \sum_{j=1}^{3n} s_j$
  - do there exist disjoint subsets $J_i \subset I = \{1, \ldots, 3n\}$ such that

$$\sum_{j \in J_i} s_j = b; \quad i = 1, \ldots, n$$

<u>Proving NP-completeness</u>

If an NP-complete problem P can be polynomially reduced to a problem $Q \in \mathcal{NP}$, then this proves that Q is NP-complete (transitivity of polynomial reductions)

<u>Example</u>: $PARTITION \propto P2||C_{max}$
Proof: on the board

<u>Famous open problem</u>: Is $\mathcal{P} = \mathcal{NP}$?

- solving one NP-complete problem polynomially, would imply $\mathcal{P} = \mathcal{NP}$