A reminder on NP-completeness

Kassian Kobert & Roland Wittler

Seminar "Karp's 21 problems"
(SoSe 2024)

Outline

Problems

Optimization problem: Find a solution optimizing an objective function, e.g., minimizing a cost or maximizing a score function.

Decision problem: Yes/No question: Is there a solution?

Complexity: Optimization at least as hard as decision. (Otherwise we could optimize to decide.)

$\Rightarrow$ To show hardness, we usually use the decision problem.

Complexity classes

P contains all problems for which a deterministic algorithm exists that can solve a problem instance in polynomial time.

NP contains all problems for which a non-determ. algorithm exists that can solve a problem instance in polynomial time.

NP contains all problems for which a deterministic algorithm exists that can verify whether a given certificate is a correct solution in polynomial time.

Or short:

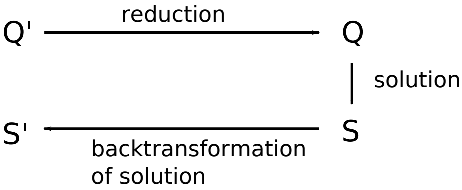
P = efficiently solvable,

NP = efficiently verifiable.

Obviously $\mathrm{P} \subseteq \mathrm{NP}$, but: $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$

Reducibility

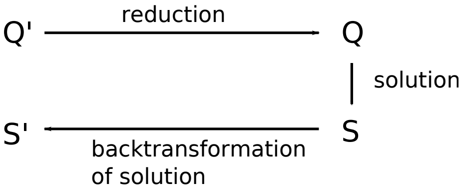
A problem Q' is reducible to a problem Q if every instance of Q' can be formulated as an instance of Q so that the solution S of problem Q corresponds to the solution S' of problem Q'.



[Seq-Anal. lecture notes]

Reducibility

A problem Q' is reducible to a problem Q if every instance of Q' can be formulated as an instance of Q so that the solution S of problem Q corresponds to the solution S' of problem Q'.



[Seq-Anal. lecture notes]

Problem Q' is polynomial-time reducible to Q (shortly: $Q' \propto Q$) if the reduction takes only polynomial time.

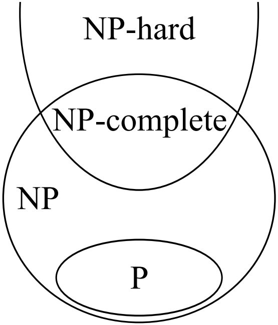
NP-hardness, NP-completeness

A problem Q is NP-hard if $Q' \propto Q$ for every $Q' \in \mathrm{NP}$.
(...at least as hard as any problem in NP.)

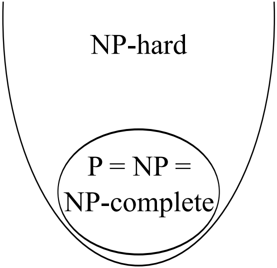
A problem Q is NP-complete if it is NP-hard and $Q \in \mathrm{NP}$.
(...not harder than NP.)

- If some NP-complete problem is solvable in polynomial time, then P = NP.
- If some problem of NP is not solvable in polynomial time, then no NP-complete problem is solvable in polynomial time.

NP-hardness, NP-completeness



$P \neq NP$



$P = NP$

[Seq-Anal. lecture notes]

NP-completeness

"Hard" does not mean "impossible"

- small instances
- run time heuristics
- exactness heuristics
- approximation
- restriction
- parametrization (fixed parameter tractability)

Outline

NP-completeness

Proving NP-completeness

Example: Clique (from Satisfiability)

Proving NP-completeness

To prove NP-completeness of a problem Q, you usually do:

(A) Show that $Q \in \text{NP}$, i.e. poly-time verifiable.

(B) Show that Q is NP-hard:

  (1) Choose a known NP-complete problem Q'.
  (2) Find a reduction f from Q' to Q.
  (3) Show that f is poly-time.
  (3) Show that for any instance I:
  There is a solution for I on Q' $\Leftrightarrow$ there is a solution for $f(I)$ on Q.

  ($\Rightarrow$ Q at least as hard as Q'
  $\Rightarrow$ at least as hard as any problem in NP.)

Karp provides B.1 and B.2.
Our exercise: A, B.3 and B.4.

Outline

NP-completeness

Proving NP-completeness

Example: Clique (from Satisfiability)

Example: Clique (from Satisfiability)

CLIQUE

INPUT: graph G, positive integer k

PROPERTY: G has a set of k mutually adjacent nodes.

Example: Clique (from Satisfiability)

CLIQUE variants

Maximum clique problem: output a maximum clique.

Weighted maximum clique problem: for a weighted graph, output a clique with maximum total weight.

Maximal clique listing problem: list all maximal cliques.

k-clique problem: output a clique with k vertices.

Clique decision problem: Boolean version of k-clique problem ("Karp's variant").

Example: Clique (from Satisfiability)

History of CLIQUE

- Both the clique listing problem and the term clique itself from the social sciences: find groups of people who all know each other.
- NP-hardness of decision problem: implicitely by Cook (1970) and also known to Reiter. Explicit proof by Karp (1972)
- hard to approximate (Garey & Johnson, 1978)
- no fixed-parameter tractable algorithm is possible (Chen et. al, 2006)
- many applications: Chemistry (molecular docking), bioinformatics (phylogenetics, protein structure prediction), sociology (social networks), mathematics, . . .

Example: Clique (from Satisfiability)

SATISFIABILITY

INPUT: Clauses C_1, $C_2, \ldots,$ C_p

PROPERTY:
The conjunction of the given clauses is satisfiable;
i.e., there i a set $S \subseteq \{x_1, x_2, \ldots, x_n; \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$ such that
    a) S does not contain a complementary pair of literals
and b) $S \cap C_k \neq \emptyset$, k=1,2,...,p.

Example: Clique (from Satisfiability)

$\texttt{SATISFIABILITY} \propto \texttt{CLIQUE}$

$\texttt{N} = \{\langle \sigma, i \rangle \mid \sigma \texttt{ is a literal and occurs in } \texttt{C}_\texttt{i}\}$

$\texttt{A} = \{\{\langle \sigma, i \rangle, \langle \delta, j \rangle\} \mid i \neq j \texttt{ and } \sigma \neq \overline{\delta}\}$

$\texttt{k} = \texttt{p}$, the number of clauses.