Introduction to computational complexity

Martin Branda

Charles University in Prague
Faculty of Mathematics and Physics
Department of Probability and Mathematical Statistics

Computational Aspects of Optimization

Software

General mathematical and statistical software:

- **Matlab** and (free) OPTI toolbox: http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php/Main/HomePage
- **SAS**: OR package
- **R**: suitable packages
- **Mathematica**: findminimum, minimize, ...
- ...

Software

Modelling tools for optimization:

- **GAMS**: http://www.gams.com
- **AIMMS**: http://www.aimms.com
- **Gurobi**: http://www.gurobi.com
- **AMPL**: http://ampl.com
- **CPlex Studio**: http://www-03.ibm.com/software/products/cs/ibmilogcpleoptistud
- **MPL**: http://www.maximalsoftware.com/mpl
- ...

Software

Open Source libraries

- **COIN-OR**: http://www.coin-or.org/

Software

Solvers

- Baron (LP, NLP, MIP, MINLP, ...)
- Bonmin (NLP, MIP, MINLP, ...)
- Conopt (LP, NLP, ...)
- CPlex (LP, MIP, MIQCP, ...)
- Dicopt (MIQCP, MINLP, ...)
- Gurobi
- Knitro (LP, MINLP, MIQCP, NLP, ...)
- Lindo (LP, MINLP, MIP, MIQCP, NLP, ...)
- Minos (LP, NLP, ...)
- Mosek (LP, MIP, MIQCP, NLP, ...)
- Xpress (LP, MIP, MIQCP, ...)
- ...

Introduction to complexity theory

Wolsey (1998): Consider **decision problems** having YES–NO answers.
Optimization problem

$$\max_{x \in M} c^T x$$

can be replaced by (for some k integral)

$$\text{Is there an } x \in M \text{ with value } c^T x \geq k?$$

For a problem instance X, the **length of the input** $L(X)$ is the length of the binary representation of a standard representation of the instance.
Instance $X = \{c, M\}$, $X = \{c, M, k\}$

Example: Knapsack decision problem

For an instance

$$X = \left\{ \sum_{i=1}^{n} c_i x_i \geq k,\ \sum_{i=1}^{n} a_i x_i \leq b,\ x \in \{0,1\}^n \right\},$$

the length of the input is

$$L(X) = \sum_{i=1}^{n} \lceil \log c_i \rceil + \sum_{i=1}^{n} \lceil \log a_i \rceil + \lceil \log b \rceil + \lceil \log k \rceil$$

Running time

Definition

- $f_A(X)$ is the **number of elementary calculations** required to run the algorithm A on the instance $X \in P$.
- **Running time of the algorithm** A

$$f_A^*(l) = \sup_X \{f_A(X) : \; L(X) = l\}.$$

- An algorithm A is **polynomial** for a problem P if $f_A^*(l) = O(l^p)$ for some $p \in \mathbb{N}$.

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

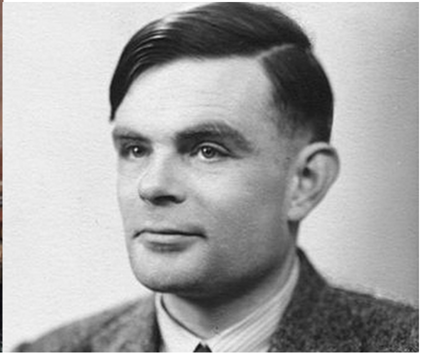
Definition

- $\mathcal{NP}$ (Nondeterministic Polynomial) is the class of decision problems with the property that: for any instance for which the answer is YES, there is a polynomial proof of the YES.
- $\mathcal{P}$ is the class of decision problems in $\mathcal{NP}$ for which there exists a polynomial algorithm.

$\mathcal{NP}$ may be equivalently defined as the set of decision problems that can be solved in polynomial time on a non-deterministic Turing machine[1].

[1] NTM writes symbols one at a time on an endless tape by strictly following a set of rules. It determines what action it should perform next according to its internal state and what symbol it currently sees. It may have a set of rules that prescribes more than one action for a given situation. The machine "branches" into many copies, each of which follows one of the possible transitions leading to a "computation tree".

Alan Turing

The Imitation Game (2014)

Polynomial reduction and the class $\mathcal{NPC}$

Definition

- If problems $P, Q \in \mathcal{NP}$, and if an instance of P can be converted in polynomial time to an instance of Q, then P is **polynomially reducible** to Q.
- $\mathcal{NPC}$, the class of $\mathcal{NP}$-**complete** problems, is the subset of problems $P \in \mathcal{NP}$ such that for all $Q \in \mathcal{NP}$, Q is polynomially reducible to P.

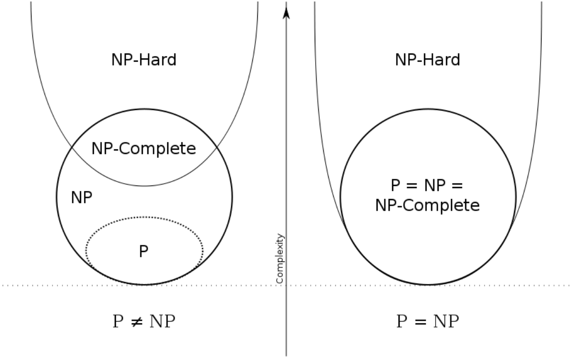
Proposition: Suppose that problems $P, Q \in \mathcal{NP}$.

- If $Q \in \mathcal{P}$ and P is polynomially reducible to Q, then $P \in \mathcal{P}$.
- If $P \in \mathcal{NPC}$ and P is polynomially reducible to Q, then $Q \in \mathcal{NPC}$.

Proposition: If $\mathcal{P} \cap \mathcal{NPC} \neq \emptyset$, then $\mathcal{P} = \mathcal{NPC}$.

Open question & Euler diagram

Is $\mathcal{P} = \mathcal{NP}$?



$\mathcal{NP}$-hard optimization problems

Definition

An optimization problem for which the decision problem lies in $\mathcal{NPC}$ is called $\mathcal{NP}$-hard.

Literature

- G.L. Nemhauser, L.A. Wolsey (1989). Integer Programming. Chapter VI in Handbooks in OR & MS, Vol. 1, G.L. Nemhauser et al. Eds.
- L.A. Wolsey (1998). Integer Programming. Wiley, New York.
- L.A. Wolsey, G.L. Nemhauser (1999). Integer and Combinatorial Optimization. Wiley, New York.