Complexity Issues

- The worst case time complexity.
- We call an algorithm for a problem $\pi$ polynomial if its running time on a computer in terms of the number of required elementary operations (such as arithmetic operations, comparisons, branching instructions, ...) is, in the worst case, bounded from above by a polynomial of degree $p$ in the size $L$ of the input data.
- We say that the algorithm runs in $O(L^p)$ time.

Example:

- The standard simplex algorithm for LP requires, in the worst case, a number of steps which is exponential in the size of the input data (e.g., Klee–Minty example).
- Kachiyan’s ellipsoid algorithm (or Karmarkar’s interior point algorithm) requires only a polynomial number of steps, and each step in these algorithms consists of a polynomial number of elementary operations.

In complexity theory, the collection of problems that can be solved in polynomial time (i.e., by a polynomial algorithm) is denoted by $P$.

Another important complexity class is $NP$, the set of all problems solvable by a “nondeterministic algorithm” in polynomial time. That is, $NP$ is the class of problems for which the correctness of a claimed solution (that may have been computed by a tedious procedure) can be verified in polynomial time.
Complexity Issues

- Clearly P is a subset of NP, and it appears natural that P \neq NP.
- However, despite enormous research efforts, it remains one of the most famous unsolved problems in theoretical computer science whether the two classes P and NP are different or not.

- We say that a problem \( \pi_1 \) is polynomially transformable to a problem \( \pi_2 \) if a polynomial algorithm for \( \pi_2 \) would imply a polynomial algorithm for \( \pi_1 \).
- A problem \( \pi \) is NP-complete if \( \pi \in \text{NP} \) and if every other problem in \( \text{NP} \) can be polynomially transformed to it.
- Every NP-complete problem has the following property: if it can be solved in polynomial time, then all problems in NP can be solved in polynomial time. In other words, if \( \pi \) is NP-complete and if \( \pi \in \text{P} \) then \( \text{P} = \text{NP} \).

Complexity Issues

- Let \( x_1, \ldots, x_n \) be a set of Boolean variables whose value is either true or false, and let \( \overline{x}_i \) denote the negation of \( x_i \).
- A literal is either a variable or its negation.
- A Boolean formula is an expression that can be constructed using literals, and the operations “and” (\( \wedge \) or \( \cdot \)) and “or” (\( \vee \) or \( + \)).
- A Boolean formula which can be made true by assigning some values to its variables is said to be satisfiable.
Complexity Issues

- The SATISFIABILITY problem is to check whether a Boolean formula of the (conjunctive normal) form \( F = \bigwedge_{i=1}^{k} (\bigvee_{j=1}^{n_i} \ell_{ij}) \), where \( \ell_{ij} \) denotes a literal, is satisfiable.
- Cook’s Theorem (1971). SATISFIABILITY is \( \text{NP–complete} \).

- Soon after the appearance of Cook’s proof, the list of \( \text{NP–complete} \) problems was substantially enriched. Another “classical” \( \text{NP–complete} \) problem is, for example, to check whether a single linear constraint
  \[ \sum_{i=1}^{n} a_i x_i = b, \quad a_i, b \text{ integers}, \]
  has a solution in \( x_i \in \{0, 1\} \) \( (i = 1, \ldots, n) \) (knapsack problem).
- Other well–known examples include the traveling salesman problem, the maximum clique problem, and many classes of nonconvex quadratic optimization problems.

- How can we prove that some problem is \( \text{NP–complete} \)?
- The following obvious consequence of the definition of \( \text{NP–completeness} \) is often used: If a problem \( \pi_1 \) is \( \text{NP–complete} \) and \( \pi_1 \) is polynomially transformable to a problem \( \pi_2 \in \text{NP} \), then \( \pi_2 \) is \( \text{NP–complete} \).
- Note, however, that one cannot conclude \( \text{NP–completeness} \) of \( \pi_2 \) by transforming it polynomially to another \( \text{NP–complete} \) problem \( \pi_1 \).

- A problem \( \pi \) is called \( \text{NP–hard} \) if there is an \( \text{NP–complete} \) problem which can be polynomially transformed to \( \pi \).
- Thus, an \( \text{NP–hard} \) problem shares with \( \text{NP–complete} \) problems the basic property of being at least as difficult as any other problem in \( \text{NP–complete} \), but it may not belong to \( \text{NP} \).
Consider the following quadratic problem:

\[
\begin{align*}
\min & \quad f(x) = c^T x + \frac{1}{2} x^T Q x \\
\text{s.t.} & \quad x \geq 0
\end{align*}
\]

where \( Q \) is an \( n \times n \) symmetric matrix, and \( c \in \mathbb{R}^n \).

The KKT conditions for this problem become the following so-called linear complementarity problem (denoted by \( \text{LCP}(Q, c) \)):

Find \( x \in \mathbb{R}^n \) (or prove that no such \( x \) exists) such that

\[
\begin{align*}
Q x + c & \geq 0, \quad x \geq 0 \\
x^T (Q x + c) & = 0
\end{align*}
\]

Hence, the complexity of finding (or proving existence of) KKT points for the above quadratic problem is reduced to the complexity of solving the corresponding (symmetric) LCP.

**Theorem.** The Problem LCP \((Q, c)\) is \(\text{NP-hard}\).

**Proof:** Consider the following LCP \((Q, c)\) problem in \(\mathbb{R}^{n+3}\) defined by

\[
Q_{(n+3) \times (n+3)} = \begin{bmatrix}
-I_n & e_n & -e_n & 0_n \\
e_n^T & -1 & -1 & -1 \\
-e_n^T & -1 & -1 & -1 \\
0_n^T & -1 & -1 & -1
\end{bmatrix},
\]

\[
c_{n+3}^T = (a_1, \ldots, a_n, -b, b, 0),
\]

where \(a_i, i = 1, \ldots, n, \) and \( b \) are positive integers, \( I_n \) is the \((n \times n)\)–unit matrix and \( e_n \in \mathbb{R}^n, \ 0_n \in \mathbb{R}^n \) are the vectors of all ones and zeros, respectively.
Complexity: KKT Points in QP

- Define the following knapsack problem. Find a feasible solution to the system
  \[ \sum_{i=1}^{n} a_i x_i = b, \quad x_i \in \{0, 1\} \quad (i = 1, \ldots, n). \]

- This problem is known to be NP-complete. Next we show that the LCP \((Q, c)\) is solvable iff the associated knapsack problem is solvable.

- If \(x\) solves the knapsack problem, then \(y = (a_1 x_1, \ldots, a_n x_n, 0, 0, 0)^T\) solves LCP \((Q, c)\).

Complexity of Local Minimization

- Therefore, in quadratic programming, the problem of deciding whether a Kuhn–Tucker point exists is NP-hard.

- Next we investigate the complexity of finding locally optimal solutions to nonlinear optimization problems.

Complexity: KKT Points in QP

- Conversely, assume the point \(y\) solves the LCP \((Q, c)\) given above.

- Since \(Q y + c \geq 0, y \geq 0\) we obtain \(y_{n+1} = y_{n+2} = y_{n+3} = 0\). This in turn implies that \(\sum_{i=1}^{n} y_i = b\) and \(0 \leq y_i \leq a_i\).

- Finally, if \(y_i < a_i\), then \(y^T (Q y + c) = 0\) enforces \(y_i = 0\). Hence, \(x = \left(\frac{y_1}{a_1}, \ldots, \frac{y_n}{a_n}\right)\) solves the knapsack problem.

Complexity: KKT Points in QP

- Computing locally optimal solutions is presumably easier than finding globally optimal solutions.

- However, from the complexity point of view we will show that the problem of checking local optimality for a feasible point and the problem of checking whether a local minimum is strict, are NP-hard even for problems with a simple structure in the constraints and the objective.
Complexity of Local Minimization

- We focus our investigation on problems that have nonconvex quadratic objective and linear constraints, that is, problems of the form:

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad Ax \geq b, \ x \geq 0
\end{align*}
\]

where \( f(x) \) is an indefinite quadratic function.

Complexity of Local Minimization

- Consider now the 3–satisfiability (3–SAT) problem: Given a set of Boolean variables \( x_1, \ldots, x_n \) and given a Boolean expression \( S \) (in conjunctive normal form) with exactly 3 literals per clause,

\[
S = (\ell_{11} + \ell_{12} + \ell_{13})(\ell_{21} + \ell_{22} + \ell_{23}) \cdots (\ell_{m1} + \ell_{m2} + \ell_{m3})
\]

where each literal \( \ell_{ij} \) is either some variable \( x_k \) or its negations \( \bar{x}_k \), is there a truth assignment for the variables \( x_i \) which makes \( S \) true?

- Cook: 3–SAT is \( \text{NP} \)-complete.

Complexity of Local Minimization

- For each instance of 3–satisfiability we construct an instance of an optimization problem in the real variables \( x_0, x_1, \ldots, x_n \).

- Clause in \( S \) \( \leftrightarrow \) a linear inequality

\[
\begin{align*}
\ell_{ij} &= x_k \\
\ell_{ij} &= \bar{x}_k
\end{align*}
\]

\[
\cdots + x_0 \geq \frac{3}{2}.
\]

- Example: for the clause \( x_1 + x_2 + \bar{x}_3 \) we have \( x_1 + x_2 + (1 - x_3) + x_0 \geq \frac{3}{2} \).

Complexity of Local Minimization

- Thus, we associate to \( S \) a system of linear inequalities

\[
A_S x \geq (\frac{3}{2} + c)
\]

where \( A_s \) is a (sparse) matrix with entries in \( \{0, 1, -1\} \) and \( x^T = (x_0, \ldots, x_n) \).

- Let us consider the set \( D(S) \subset \mathbb{R}^{n+1} \) of feasible points satisfying the following linear constraints

\[
A_S x \geq (\frac{3}{2} + c)
\]

\[
1/2 - x_0 \leq x_i \leq 1/2 + x_0, \ x_i \geq 0, \ i = 1, \ldots, n
\]
With a given instance $S$ of the 3–satisfiability problem we associate the following indefinite quadratic problem:

$$\min_{x \in D(S)} f(x) = -\sum_{i=1}^{n} (x_i - (1/2 - x_0))(x_i - (1/2 + x_0)),$$

Note that $f(x) = -\sum_{i=1}^{n} (x_i - 1/2)^2 + nx_0^2$, i.e., the objective function is a separable indefinite quadratic function with one convex and $n$ concave terms.

In addition, we have the following:

a) $f(x) \geq 0$ for all feasible points $x$. Therefore, the feasible point $x^* = (0, 1/2, \ldots, 1/2)^T$ is a local (global) minimum of $f(x)$ since $f(x^*) = 0$.

b) $f(x) = 0$ if and only if $x_i \in \{1/2 - x_0, 1/2 + x_0\}$, for $i = 1, \ldots, n$.

Recall that a strict local minimum for the above quadratic problem is a feasible point $x^*$ for which there exists an $\epsilon > 0$ such that $f(x^*) < f(x)$ for all $x \in D(S) \cap \{x : 0 < \|x - x^*\| \leq \epsilon\}$.

The following theorem implies that checking strict local optimality is NP–hard. Therefore, we cannot expect to find a polynomial time algorithm for this problem (assuming $P \neq NP$).

**Theorem.** $S$ is satisfiable iff $x^* = (0, 1/2, \ldots, 1/2)^T$ is not a strict local minimum.

**Proof:** Let $x_1, \ldots, x_n$ be a truth assignment satisfying $S$. For any $x_0$ and $i = 1, \ldots, n$ consider

$$x^0_i = \begin{cases} 1/2 - x_0 & \text{if } x_i = 0 \\ 1/2 + x_0 & \text{if } x_i = 1. \end{cases}$$

For $x^0 = (x_0, x_1^0, \ldots, x_n^0)^T$ we have $f(x^0) = 0$. Since $x_0$ can be chosen to be arbitrarily close to zero, $x^*$ is not a strict local minimum.

Suppose now that $x^* = (0, 1/2, \ldots, 1/2)^T$ is not a strict local minimum, that is, there exists $y \neq x^*$ such that $f(y) = f(x^*) = 0$; therefore, $y_i \in \{1/2 - y_0, 1/2 + y_0\}$, $i = 1, \ldots, n$. Then the variables $x_i$, $i = 1, \ldots, n$ defined by

$$x_i(y) = \begin{cases} 0 & \text{if } y_i = 1/2 - y_0 \\ 1 & \text{if } y_i = 1/2 + y_0 \end{cases}$$

satisfy $S$. 

$S$ is satisfiable iff $x^* = (0, 1/2, \ldots, 1/2)^T$ is not a strict local minimum.
Complexity of Local Minimization

- If we fix \( x_0 = 1/2 \) in the above indefinite quadratic problem, then the objective function \( f(x) \) is concave with \( x^* \) as the global minimum. Therefore, the problem of checking if a given point is a strict global minimum of a concave minimization problem is \( \text{NP–hard} \).

- Consider now the problem of checking local optimality. We prove that this problem is \( \text{NP–hard} \).

Given the 3–satisfiability problem, consider the following indefinite quadratic program:

\[
\min_{x \in \mathbb{D}(S)} \phi(x) = -\sum_{i=1}^{n} (x_i - (1/2 - x_0)) (x_i - (1/2 + x_0)) - \frac{1}{2n} \sum_{i=1}^{n} (x_i - 1/2)^2.
\]

**Theorem.** \( S \) is satisfiable iff \( x^* = (0, 1/2, \ldots, 1/2) \) is not a local minimum.

Proof: Let \( x_1, \ldots, x_n \) be a truth assignment satisfying \( S \). Given any \( x_0 \) arbitrary close to zero, define for \( i = 1, \ldots, n \)

\[
x_i^0 = \begin{cases} 
1/2 - x_0 & \text{if } x_i = 0 \\
1/2 + x_0 & \text{if } x_i = 1.
\end{cases}
\]

Then we can easily see that \( x^0 = (x_0, x_1^0, \ldots, x_n^0) \) is feasible and

\[
\phi(x^0) = -\frac{x_0^2}{2} < 0 = \phi(x^*).
\]

Hence, \( x^* \) is not a local minimum.

Suppose now that \( x^* \) is not a local minimum. Then there exists a point \( x = (x_0, \ldots, x_n)^T \) such that \( \phi(x) < 0 \). We will now show, by contradiction, that we can find in each clause of \( S \) one literal of value > 1/2. This would imply that \( S \) is satisfiable with

\[
\bar{x}_i = \begin{cases} 
0 & \text{if } x_i \leq 1/2 \\
1 & \text{if } x_i > 1/2.
\end{cases}
\]
For contradiction, assume that the value of each literal in some clause is \( \leq 1/2 \). For instance, consider a constraint (clause) of the form

\[
    x_1 + x_2 + \bar{x}_3 + x_0 \geq 3/2.
\]

For this inequality to hold, we must have a value \( \geq 1/2 - \frac{x_0}{3} \) for at least one literal \( l \). Consider the case \( l = x_1 \) (the other cases follow by an analogous argument).

By assumption we have that \( x_1 \leq 1/2 \), so

\[
    \frac{1}{2} - \frac{x_0}{3} \leq x_1 \leq \frac{1}{2} \Rightarrow -\frac{x_0}{3} \leq x_1 - \frac{1}{2} \leq 0.
\]

Hence,

\[
(x_1 - 1/2)^2 \leq \frac{x_0^2}{9}.
\]

Let

\[
p(x) = -\sum_{i=1}^{n} (x_i - (1/2 - x_0))(x_i - (1/2 + x_0))
\]

\[
= -\sum_{i=1}^{n} ((x_i - 1/2)^2 - x_0^2)
\]

be the “penalty term” in the objective function.

Then, since \( (x_1 - 1/2)^2 \leq \frac{x_0^2}{9} \),

\[
p(x) \geq -(x_1 - 1/2)^2 + x_0^2 \geq \frac{8}{9} x_0^2.
\]

On the other hand, for the “payoff term” \( q(x) = -\frac{1}{2n} \sum_{i=1}^{n} (x_i - 1/2)^2 \) we obtain \( q(x) \geq -\frac{x_0^2}{2} \).

Hence \( \phi(x) \geq \frac{8}{9} x_0^2 - \frac{1}{2} x_0^2 > 0 \), a contradiction. \( \square \)

\begin{itemize}
  <li>Complexity analysis is fundamental in order to understand the inherent difficulty of nonconvex problems and has been a motivation to develop new algorithms.</li>
  <li>It is not clear whether nonconvexity is the only source of complexity, since some classes of nonconvex problems can be solved by polynomial time algorithms.</li>
  <li>Furthermore, there is no easy way to check if a given complicated function is convex or not (even in the case of multivariable polynomials).</li>
\end{itemize}