Discrete Optimization and Integer Programming.

Introduction

“Nothing takes place in the world whose meaning is not that of some maximum or minimum.” (L. Euler)

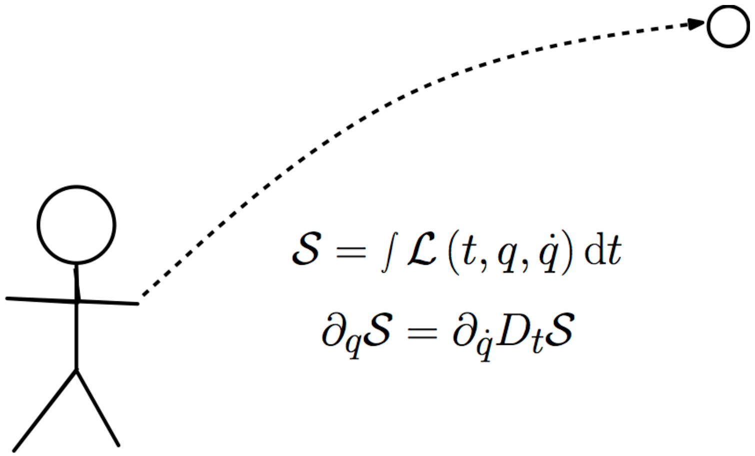
Contents

I. Popular Optimization Problems

- ***(Strong) Fermat's principle***

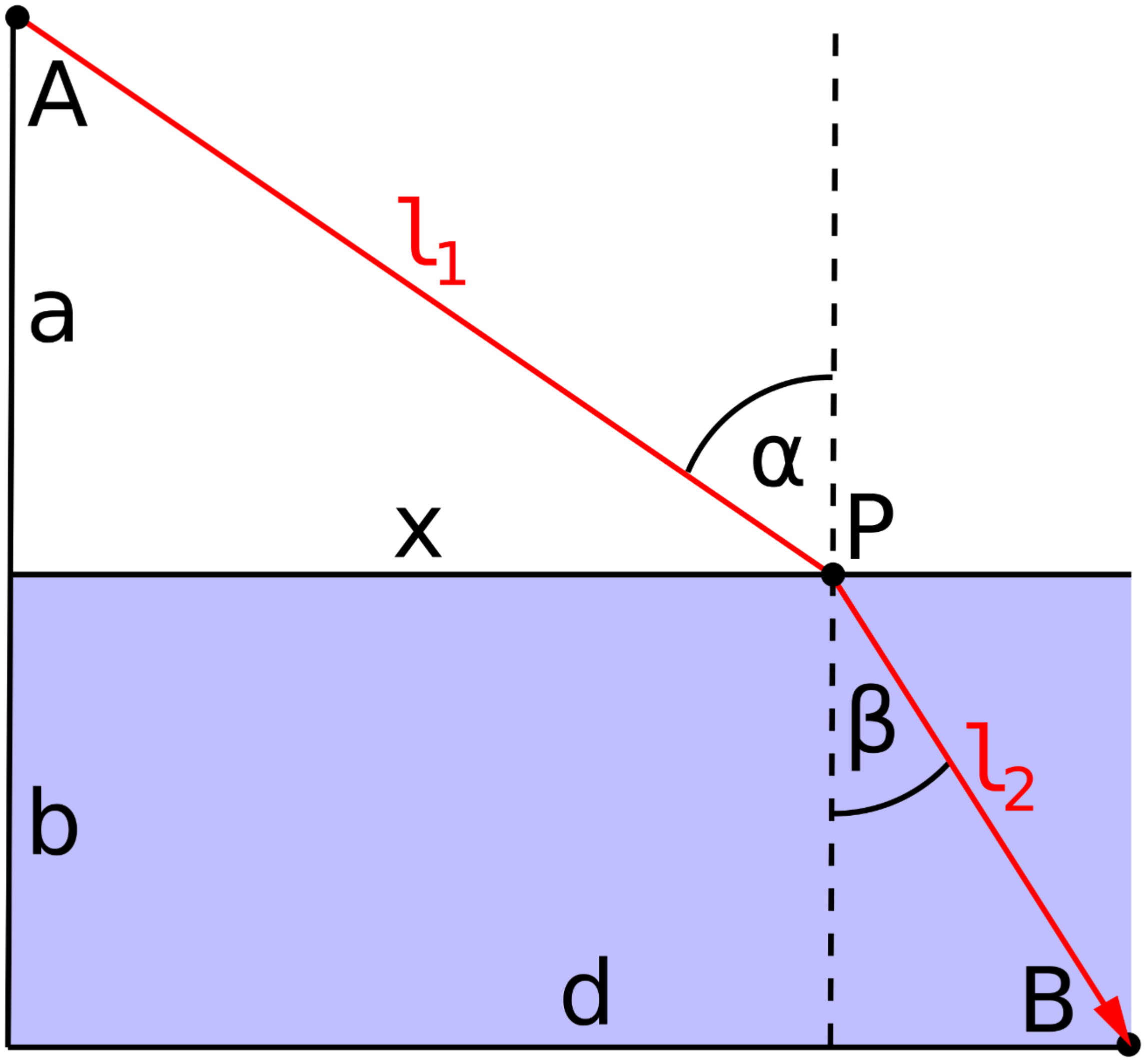
In the geometrical optics the path taken by a ray between two given points (from A to B) is the path that can be traveled in the least time.

$$\boldsymbol{T = \int_A^B dt = \int_A^B \frac{ds}{v_r} \rightarrow min}$$



$$\mathcal{S} = \int \mathcal{L}\left(t, q, \dot{q}\right) \mathrm{d}t$$

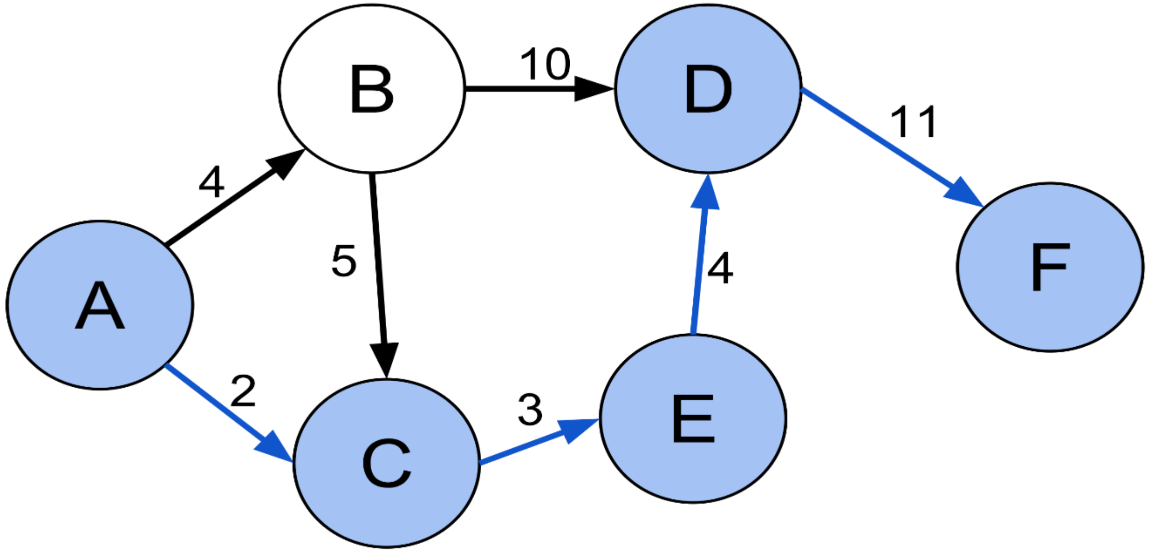
$$\partial_q \mathcal{S} = \partial_{\dot{q}} D_t \mathcal{S}$$



I. Popular Optimization Problems

- ***Shortest Path Problem***

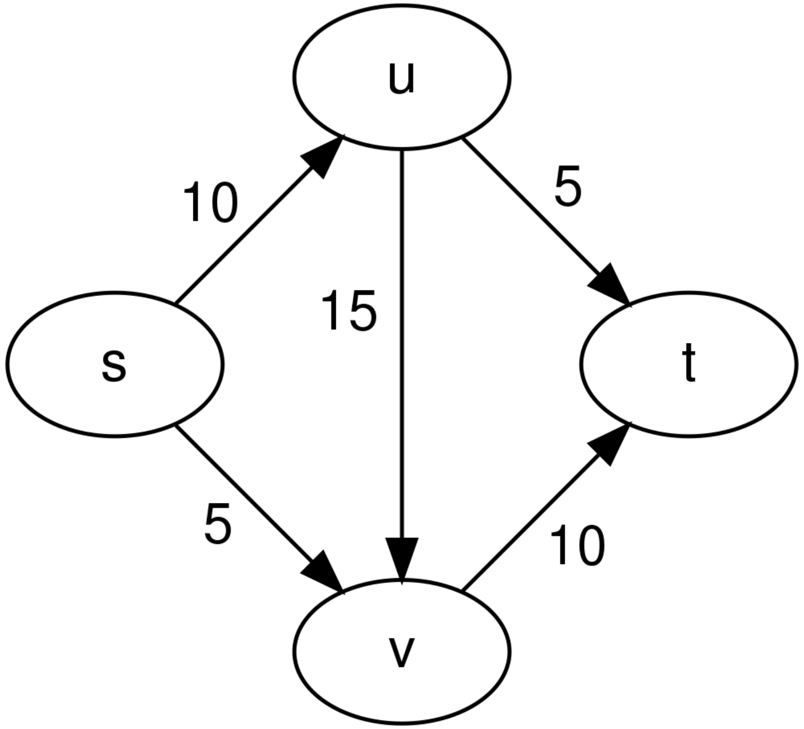
The shortest path problem is the problem of finding a path between two vertices (source and target) in a graph such that the sum of the weights of its constituent edges is minimized.



I. Popular Optimization Problems

- ***Maximum Flow Problem***

The maximum flow problem is to route as much flow as possible from the source to the sink.

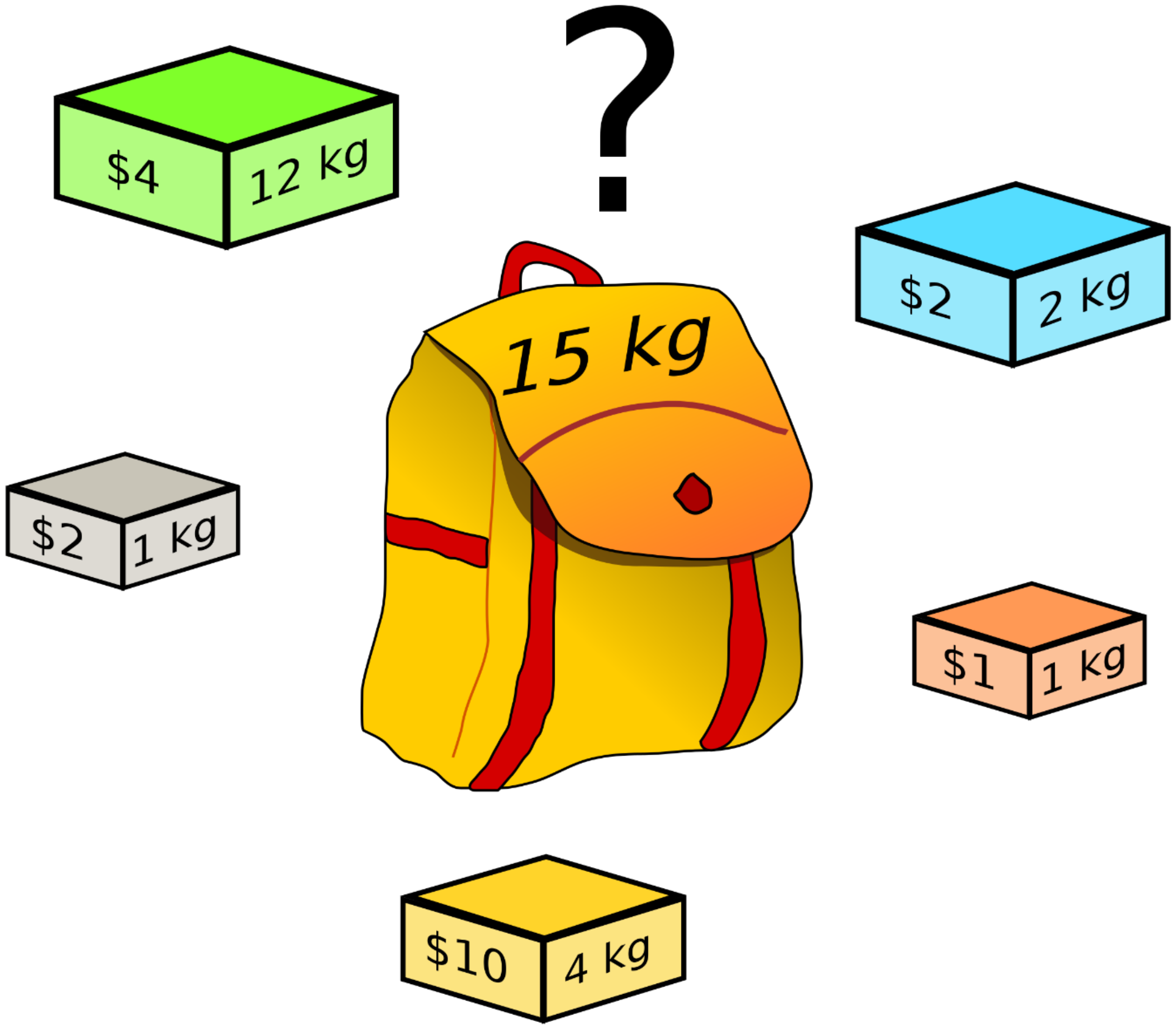


I. Popular Optimization Problems

- ***Knapsack Problem***

Given a set of items, each with a weight (w_i) and a value (v_i), determine the number (x_i) of each item to include in a collection so that the total weight is less than or equal to a given limit (W) and the total value is as large as possible.

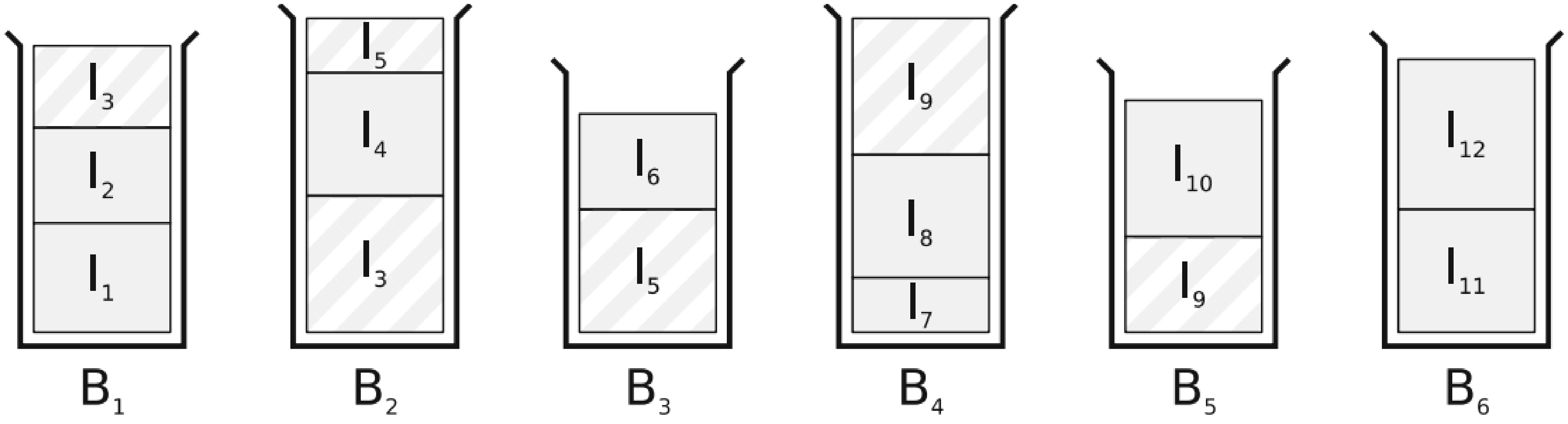
$$\begin{cases} \sum v_i x_i \to max \\ \sum w_i x_i \leq W,\ x_i \in \mathbb{N} \end{cases}$$



I. Popular Optimization Problems

- ***Bin Packing Problem***

The bin packing problem is to pack items of different sizes a finite number of bins or containers, each of a fixed given capacity, in a way that minimizes the number of bins used.



II. Mathematical Optimization

• *Continuous/discrete*

An optimization problem can be represented in the following way:

- *Given:* an objective function $f : A \to \mathbb{R}$ from some set A to the real numbers,
- Sought: an element $x_0 \in A$ such that $f(x_0) \leq f(x)$ for all $x \in A$ ("minimization") or such that $f(x_0) \geq f(x)$ for all $x \in A$ ("maximization").

If the set $A \subset \mathbb{R}^n$ then the corresponding optimization problem is called *continuous.*

For these problems we can use the different calculus techniques.

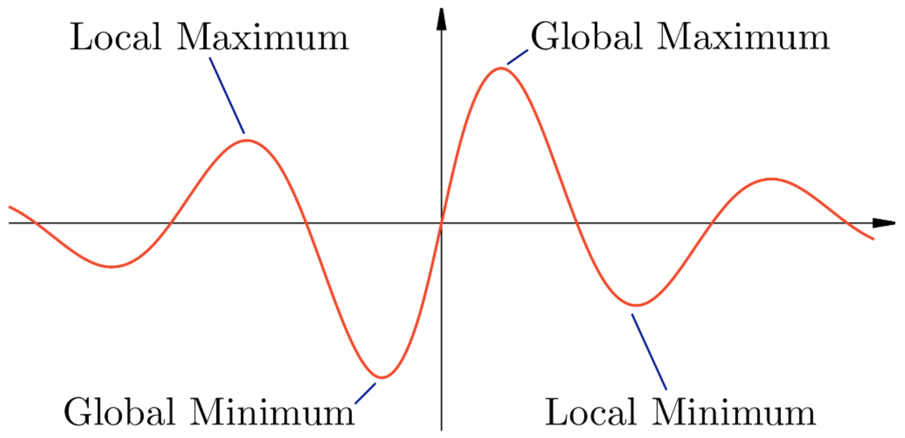
If the set $A \subset \mathbb{Z}^n$ then the corresponding optimization problem is called *discrete.*

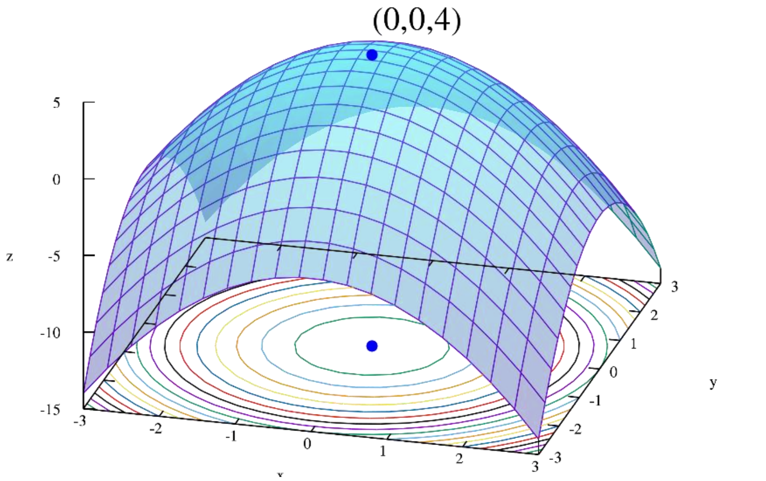
For these problems we cannot use the calculus. In general, the discrete optimization problems are much harder than the continuous problems.

II. Mathematical Optimization

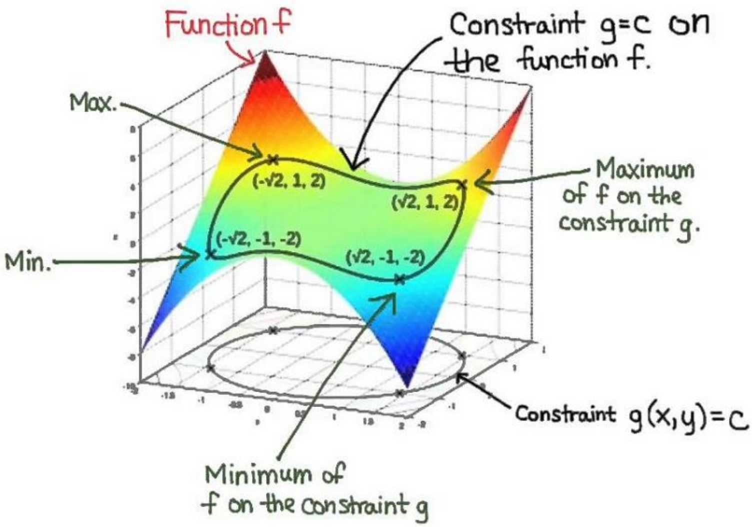
• *Unconstrained/constrained*

- If the set $A = \mathbb{R}^n$ $(\mathbb{Z}^n)$ then the corresponding optimization problem is called *unconstrained.*





- If the set $A \subsetneq \mathbb{R}^n$ $(\mathbb{Z}^n)$ then the corresponding optimization problem is called *constrained.* Typically, constraints are determined by the equalities or inequalities. For example, $\mathrm{A} = \{\mathrm{x} \in \mathbb{R}^{\mathrm{n}} : \mathrm{g}_1(\mathrm{x}) = 0, \mathrm{g}_2(\mathrm{x}) \leq 0\}$.



II. Mathematical Optimization

• *Linear/Quadratic/…/Nonlinear*

- If $A \subset \mathbb{R}^n$ and an objective function f and all constraints are linear then the optimization problem is called *linear*. The field of optimization theory studying these problems is called *Linear Programming (LP)*.

- If $A \subset \mathbb{Z}^n$ and everything else is as above then the problem is called *integer linear*. Correspondingly, the field of optimization theory studying such problems is called *Integer Linear Programming (ILP or just IP)*.

- If an objective function or constraints are nonlinear then the problem is called *nonlinear.*

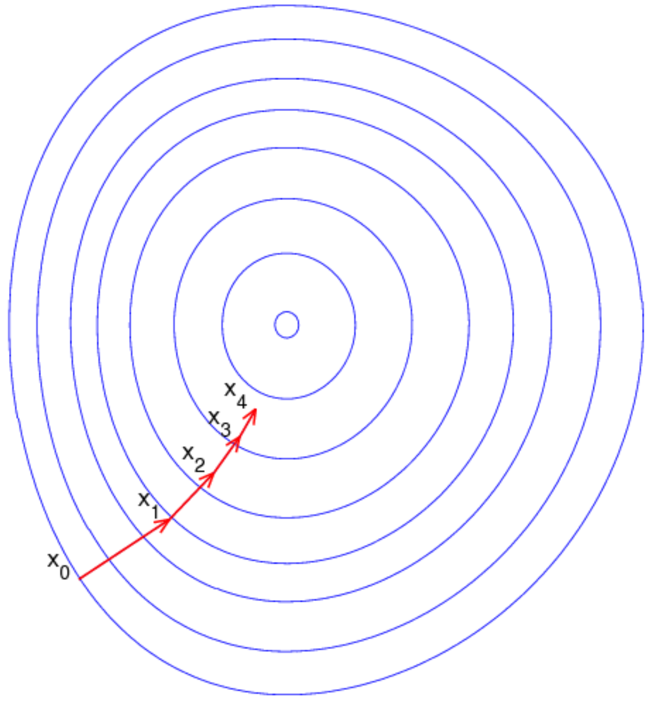
- Among the nonlinear problems the *polynomial* problems (*quadratic*, *cubic* and so on) should be distinguished.

III. Some Methods of Solving

• *Continuous Nonlinear Optimization*

Typically, continuous nonlinear unconstrained optimization problems are numerically solved using iterative algorithms like the *gradient descent*. Roughly this algorithm looks as follows

1. Take an initial point x_0;
2. Compute the gradient $\nabla f(x_0)$ of the objective function f at the point x_0;
3. Move the point x_0 to the point $x_1 := x_0 + \lambda \nabla f(x_0)$
4. Do the same with the point x_1 obtaining the point x_2;
5. And so on until we reach the position sufficiently closed to the true point of the minimum.



III. Some Methods of Solving

• *Linear Programming*

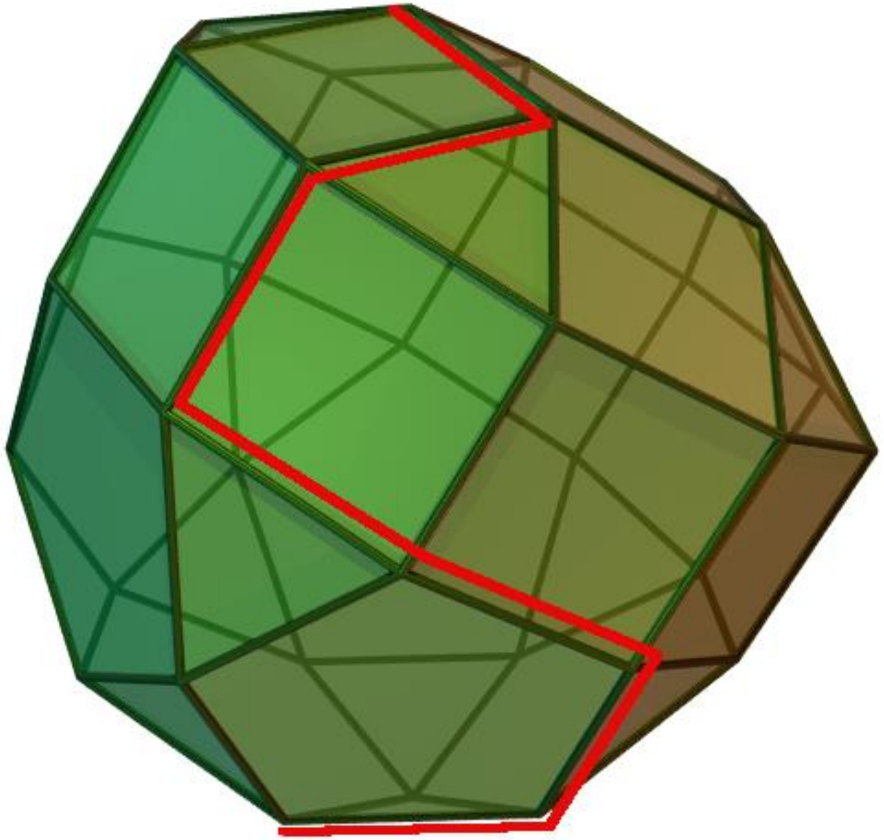
One of the popular algorithms of solving problems in Linear Programming is *Dantzig's simplex algorithm*.

Suppose we have the following linear problem

$$\sum c_i x_i \to max, \qquad \sum a_{ij} x_j \leq b_i \,, x_j \geq 0.$$

Note that these linear constraints define a *polytope*. Geometrically the method looks as follows:

1. Picking a vertex of the polytope;
2. Look at the neighbors of this vertex;
3. If there is a neighbor that gives a better solution we move to that new vertex.
4. We follow this procedure until we reach a vertex that none of its neighboring vertices will give a better solution.



III. Some Methods of Solving

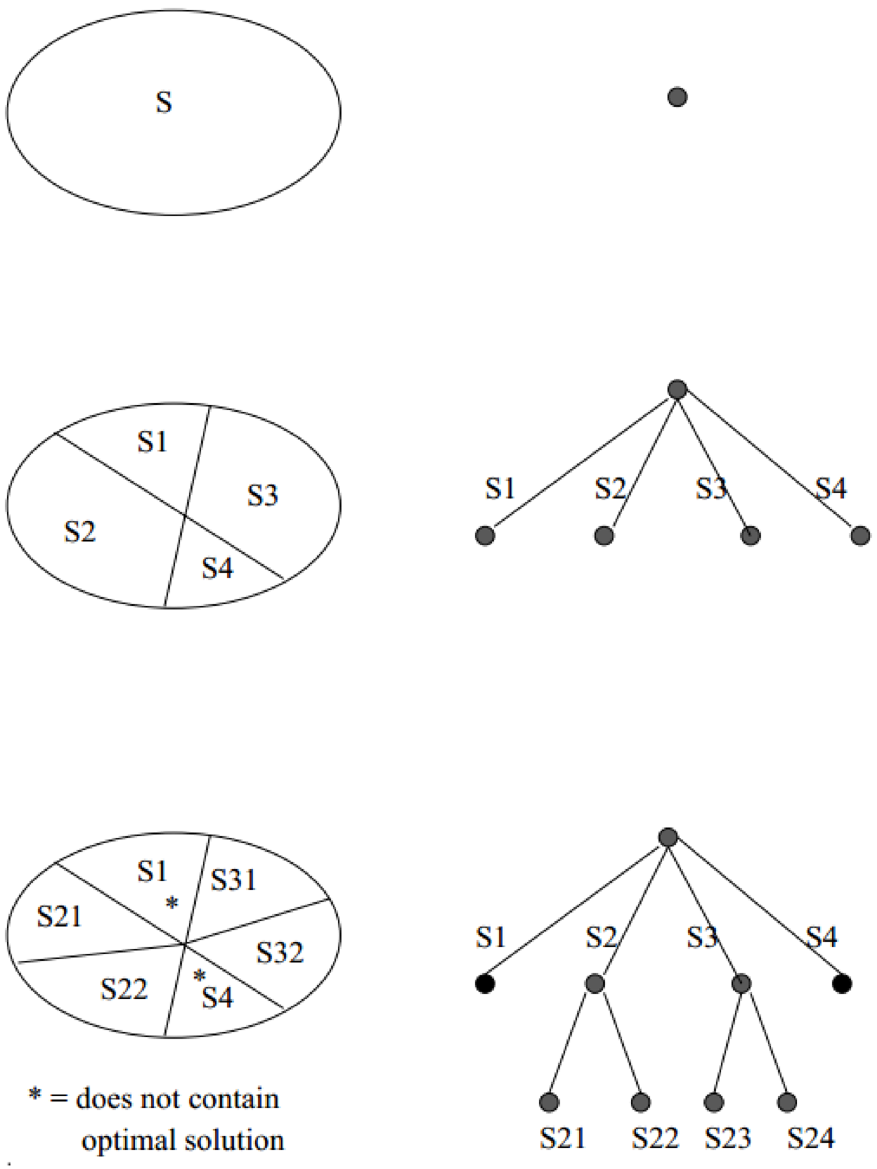
- ***Integer Programming***

Typically, the integer linear optimization problems are solved by using the *Branches and Bounds* paradigm.

This paradigm is based on

- stepwise constructing a tree T whose nodes are subsets of the set of all feasible solutions (branching);
- computing for each node of the tree T the upper/lower bound for the values of the objective function over the corresponding subset.

This paradigm is the basis of the *IP solvers* which we will study in details during this course.



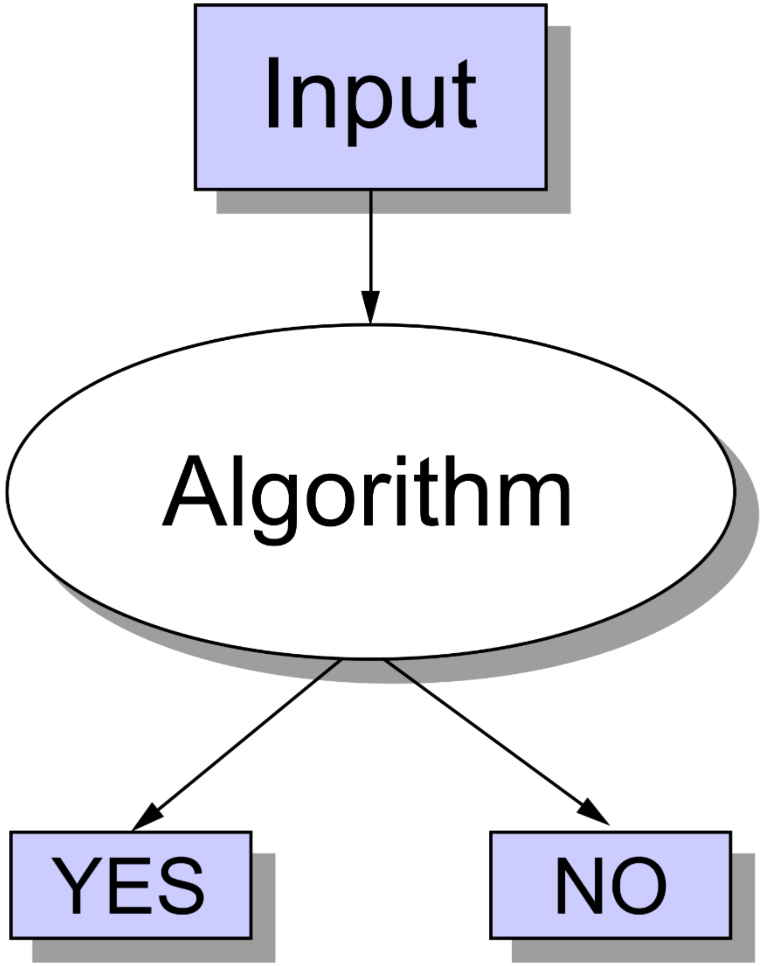
IV. Complexity Theory

- ***Decision Problems***

A *decision problem* is a problem that can be posed as a yes–no question of the input values.

Unlike decision problems, for which there is only one correct answer for each input, optimization problems are concerned with finding the best answer to a particular input.

On the other hand, there are standard techniques for transforming optimization problems into decision problems.



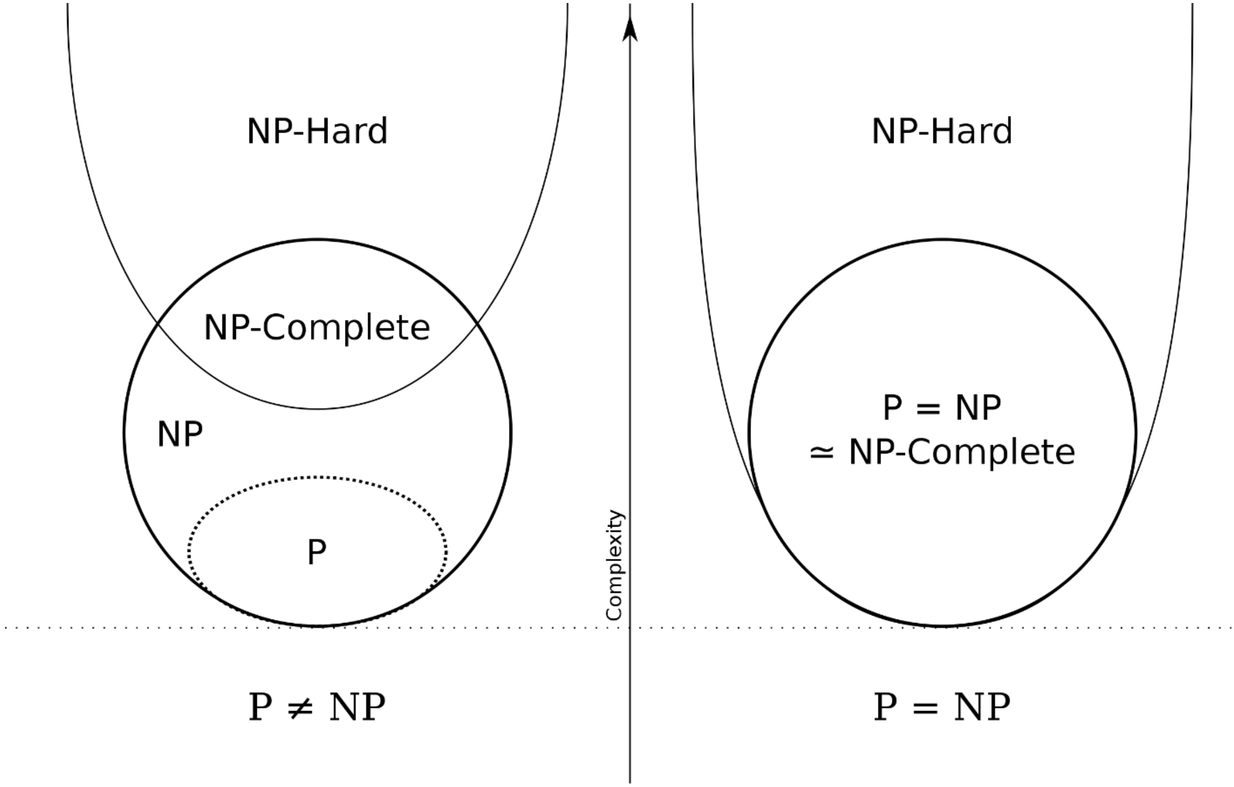
IV. Complexity Theory

• *N/NP classes*

From the class of all decision problems one can select the following important subclasses:

- *class P* which contains all decision problems that can be solved by a deterministic Turing machine using a polynomial time.

- *class NP* which contains all decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine.



Unsolved Millennium Prize Problem: to prove that $P \neq NP$ or $P = NP$.

IV. Complexity Theory

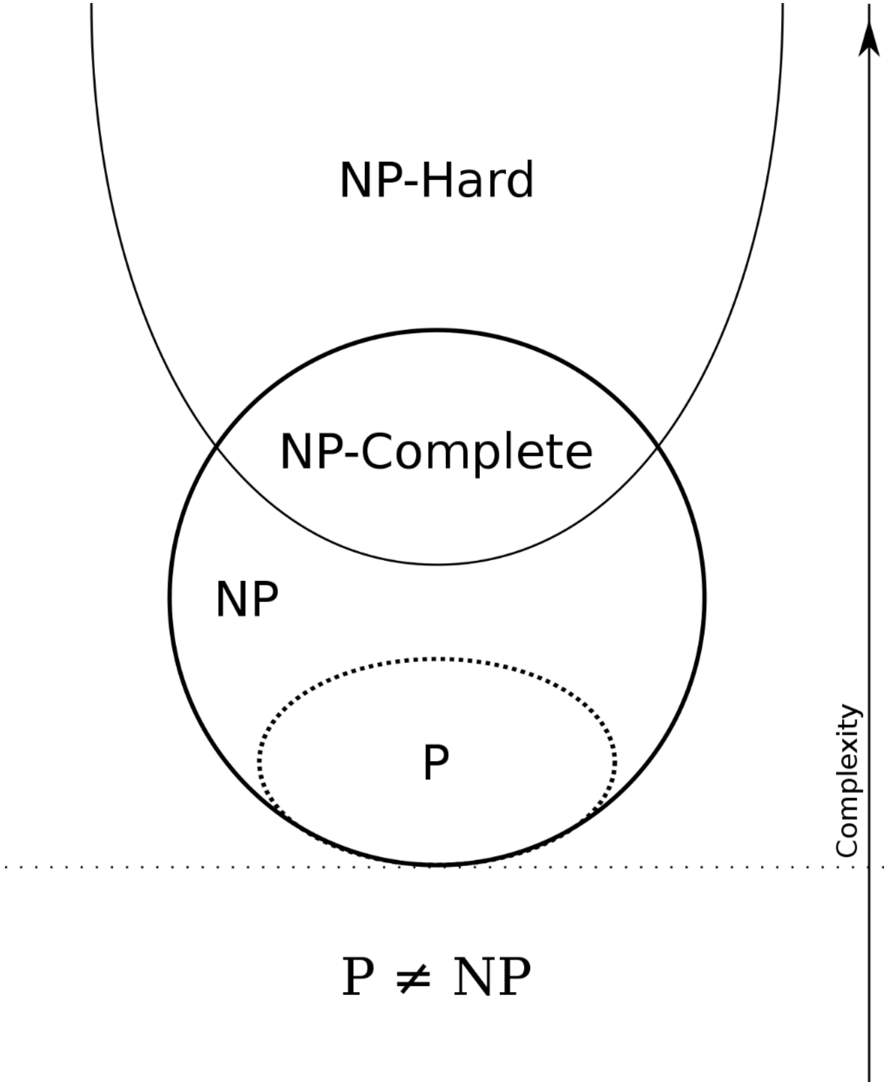
- ***NP-hard/NP-complete***

A decision problem H is called *NP-hard* when every decision problem L in NP can be reduced in polynomial time to H.

A decision problem H is called *NP-complete* when H is in NP and H is NP-hard at the same time.

The same classification is applied to the optimization problems depending on which class the corresponding decision problem belongs to.

The most of the important discrete optimization problems are NP-complete. In practice it means that there are no polynomial solutions for these problems and we need heuristic algorithms.



Thank you.

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI