Evolutionary Computation

What is evolutionary computation?

Evolutionary computation (EC) is a sub-field of computational intelligence that use ideas and get inspiration from natural evolution.

Examples

- Biological evolution of human being
- Foraging and social behavior of swarm to locate food
- Movement and pattern of ants in search of food, etc.

Need

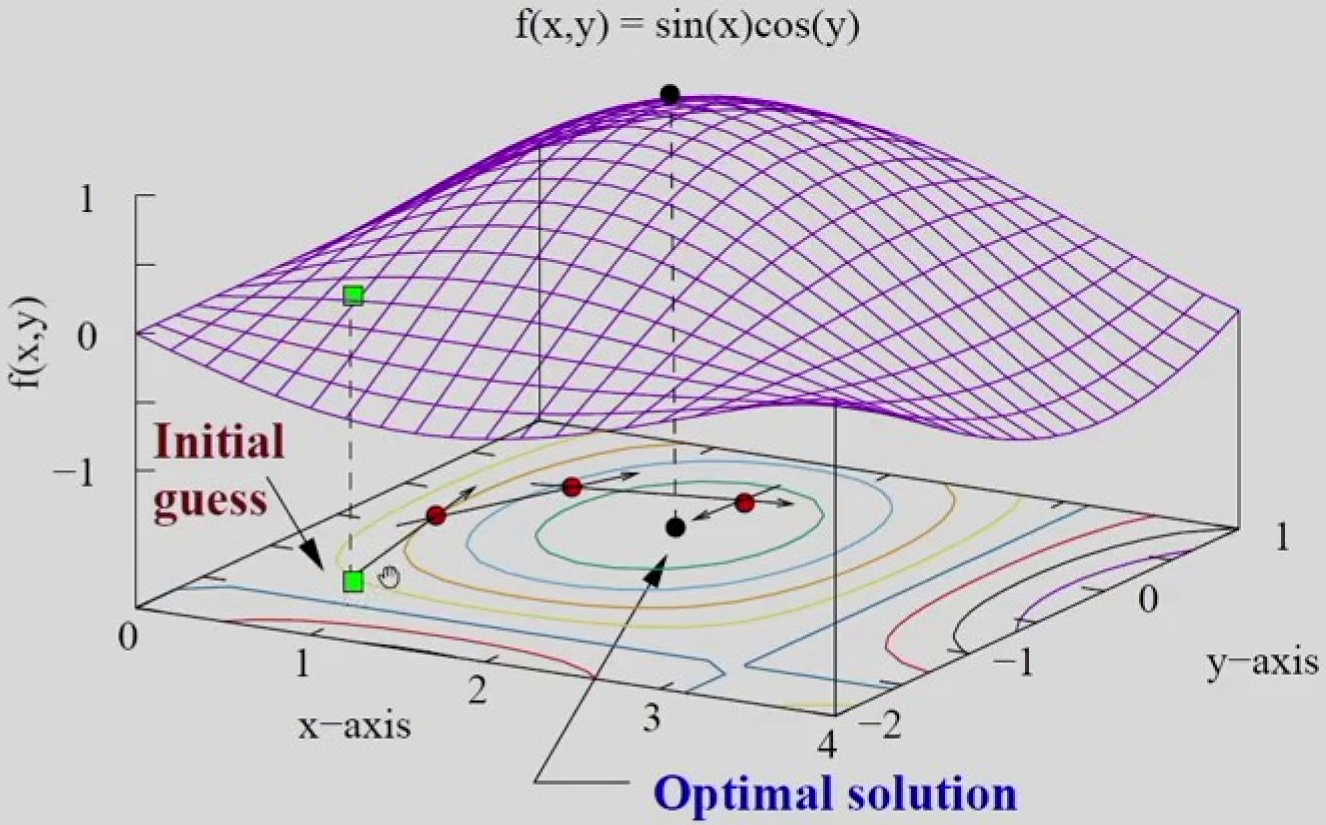
EC techniques can be used to solve complex optimization problems by generating, evaluating and modifying a population of possible solutions.

What is search and optimization?

Definition: A task of searching for a set of decision variables which would minimize or maximize objective function subjected to satisfying constraints.

Maximize: $f(x, y)$

- Decision variables: (x, y)
- Objective function: $f(x, y)$



Automotive Design

Aim: Best combination of material with best engineering to provide faster, lighter, more fuel efficient and safer vehicles.

Study: Castellani, F. and Franceschini, G., "Use of Genetic Algorithms as an Innovative Tool for Race Car Design," SAE Technical Paper 2003-01-1327, 2003.

Save: Spending years in laboratories working with polymers, wind tunnels and balsa wood shapes

Achievement: Much quicker and more efficiently by computer modeling using GA

Engineering Design

Aim: To optimize the structural and operational design of buildings, factories, machines, etc.

Applications: Design of heat exchangers, robot gripping arms, satellite booms, building trusses, flywheels, turbines, mechanisms, etc.

Advantage: Just another computer-assisted engineering design application.

Robotics

Aim: Intelligence and learning in robotics

Study: Katic D., Vukobratovic M. (2003) Genetic Algorithms in Robotics. In: Intelligent Control of Robotic Systems. International Series on Microprocessor-Based and Intelligent Systems Engineering, vol 25. Springer, Dordrecht.

Applications: Robot path planning, robot vision, robot speech, robot behavior, etc.

Evolvable Hardware

Aim: Electronic circuits created by GA computer models

Study: E. Stomeo, T. Kalganova and C. Lambert, "A Novel Genetic Algorithm for Evolvable Hardware," 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, 2006, pp. 134-141.

Application: New hardware, self-configurable electronic system

Advantage: Rapid circuit design

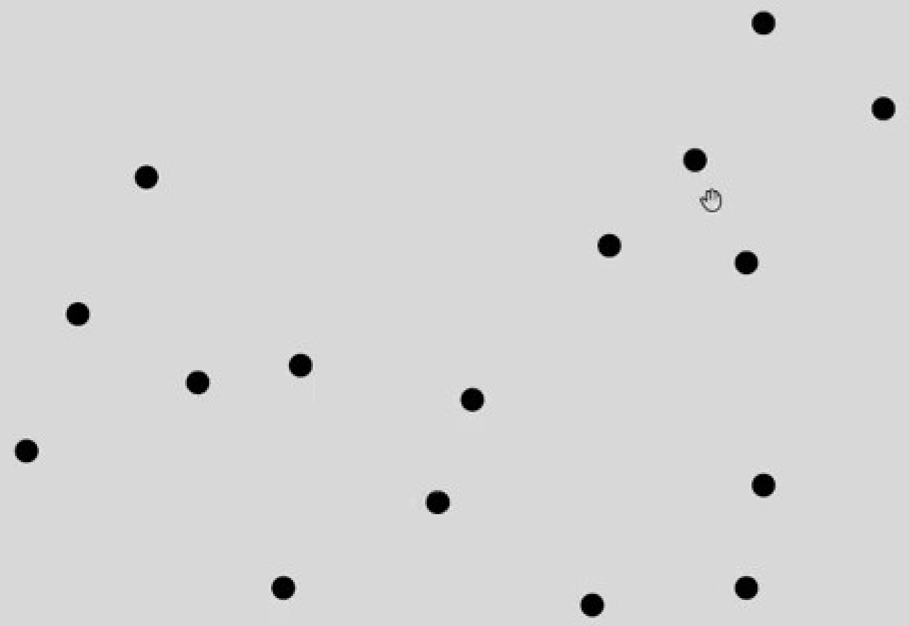
Traveling Salesman Problem (TSP)

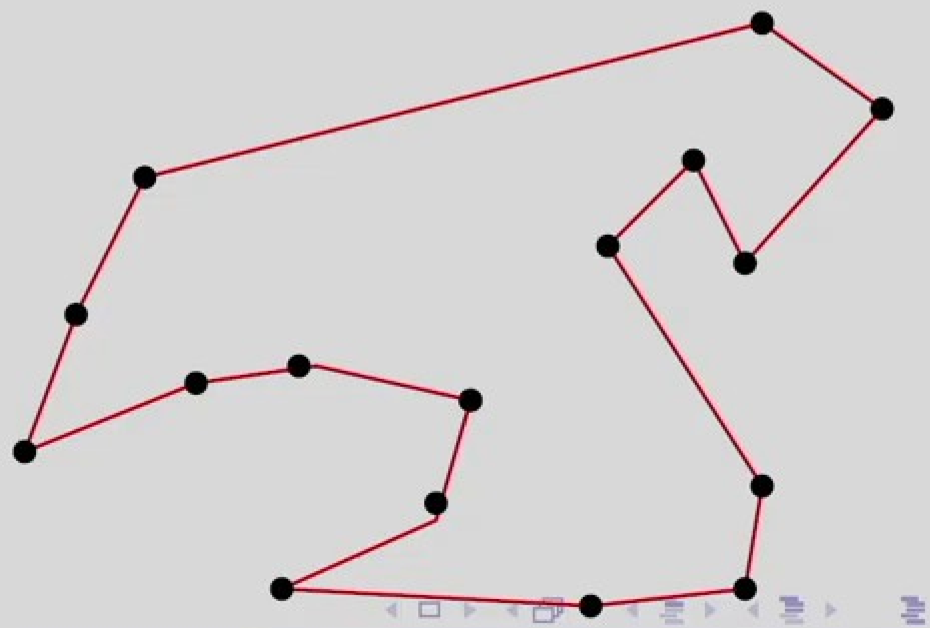
Aim: Shortest route for a salesman

Study: Potvin, J. Genetic algorithms for the traveling salesman problem. Ann Oper Res 63, 337370 (1996).

Applications: Routing and Scheduling, Transportation Problems, Travel Trip, Traffic, Shipment Routing, Production Planning, etc.

Advantages: A large size of problem can be easily solved with desired accuracy.





Optimized Telecommunications Routing

Aim: Dynamic and anticipatory routing of circuits for telecommunications networks

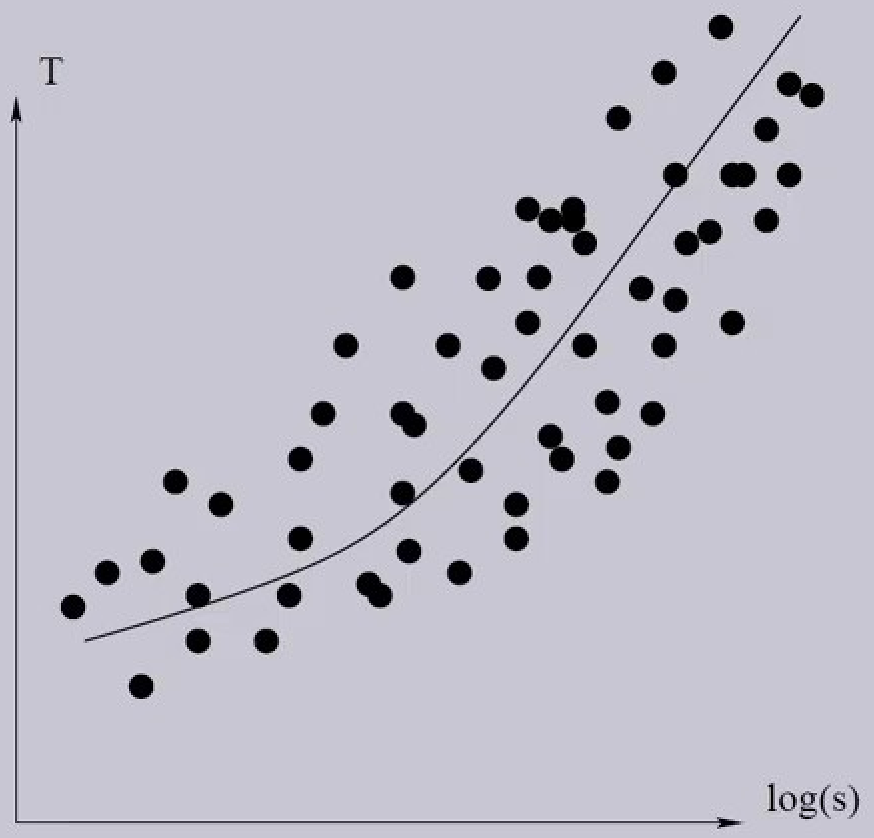
Study: Livramento S., Moura A.V., Miyazawa F.K., Harada M.M., Miranda R.A. (2004) A Genetic Algorithm for Telecommunication Network Design. In: Raidl G.R. et al. (eds) Applications of Evolutionary Computing. EvoWorkshops 2004. Lecture Notes in Computer Science, vol 3005. Springer, Berlin, Heidelberg.

Application: Optimized placement and routing of cell towers for best coverage.

Advantage: Total project cost was minimized and achieved uniform demand distribution among the various service sections.

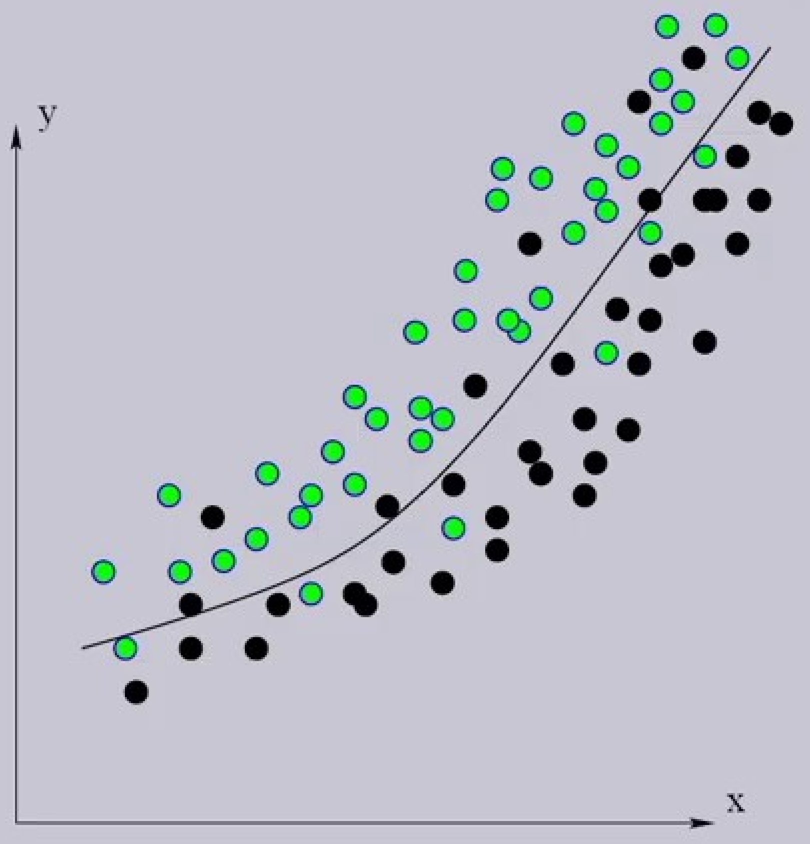
Data Mining

Regression



- Parametric optimization, prediction, etc.

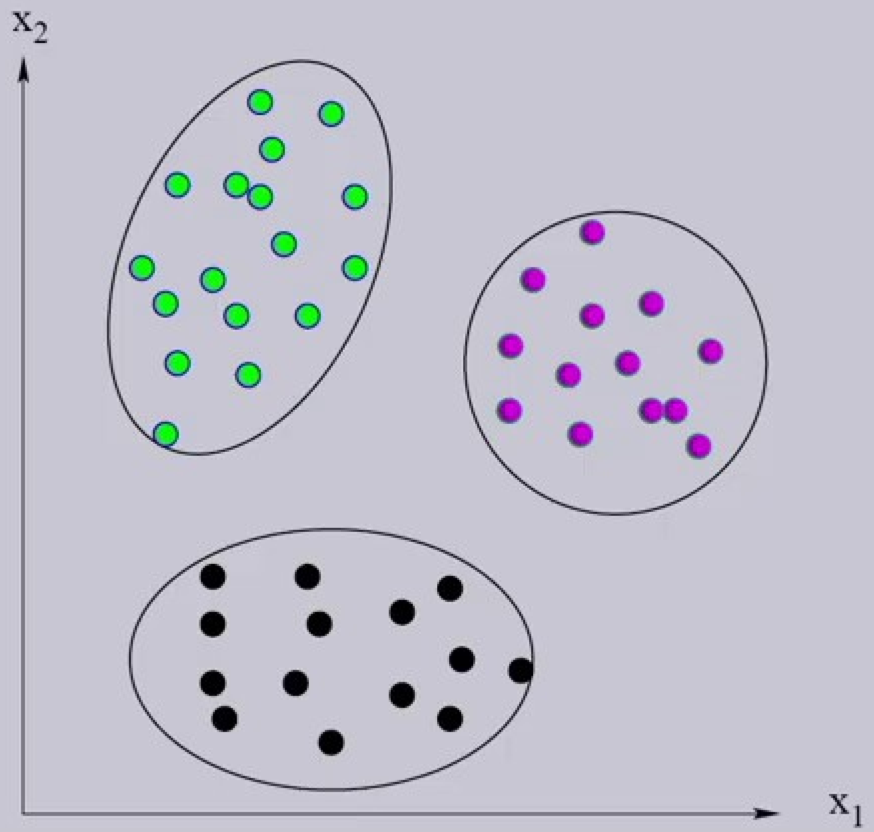
Classification



- Condition of equipment, status of diseases, etc.

Data Mining

Clustering



- Pattern recognition, image processing, etc.

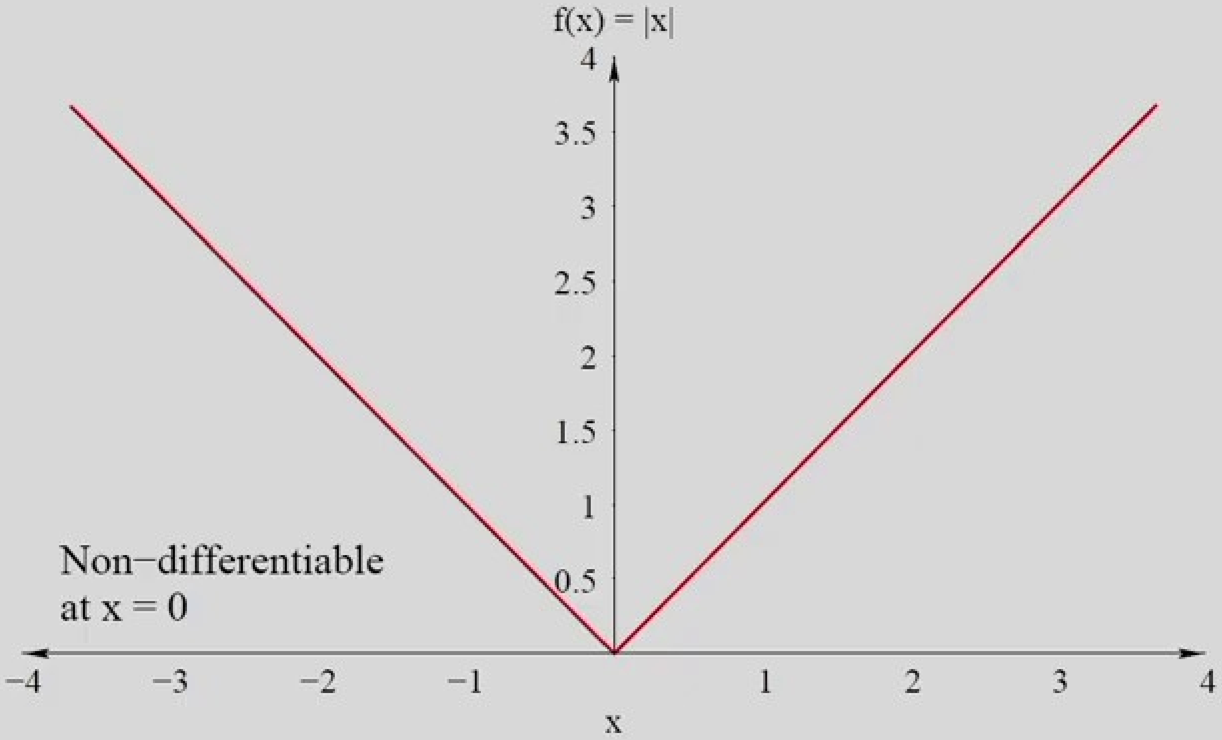
Other Applications

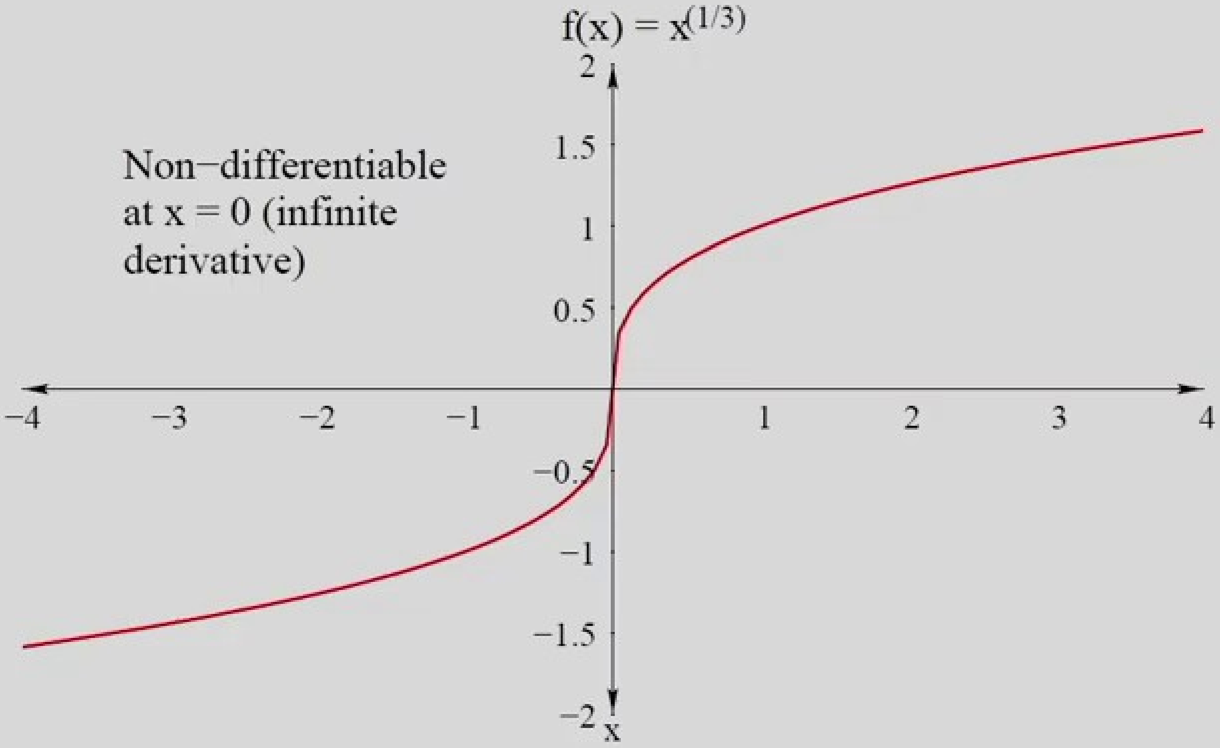
Artificial creativity, Audio watermark detection, Computer-automated design, Automated design of sophisticated trading systems in the financial sector, Bioinformatics, Biology and computational chemistry, Chemical kinetics (gas and solid phases), Code-breaking, Computer architecture, Container loading optimization, Control engineering, Design of water distribution systems, Gene expression profiling analysis, File allocation for a distributed system, Filtering and signal processing, Finding hardware bugs, Game theory equilibrium resolution, Economics, Scheduling applications, including job-shop scheduling, Molecular structure optimization (chemistry). Multidimensional systems, Plant floor layout, Pop music record producer, Protein folding and protein/ligand docking, Quality control, Cellular manufacturing systems, Timetabling problems, Airlines Revenue Management, etc.

Properties of Practical Optimization Problems

Properties of Practical Optimization Problems

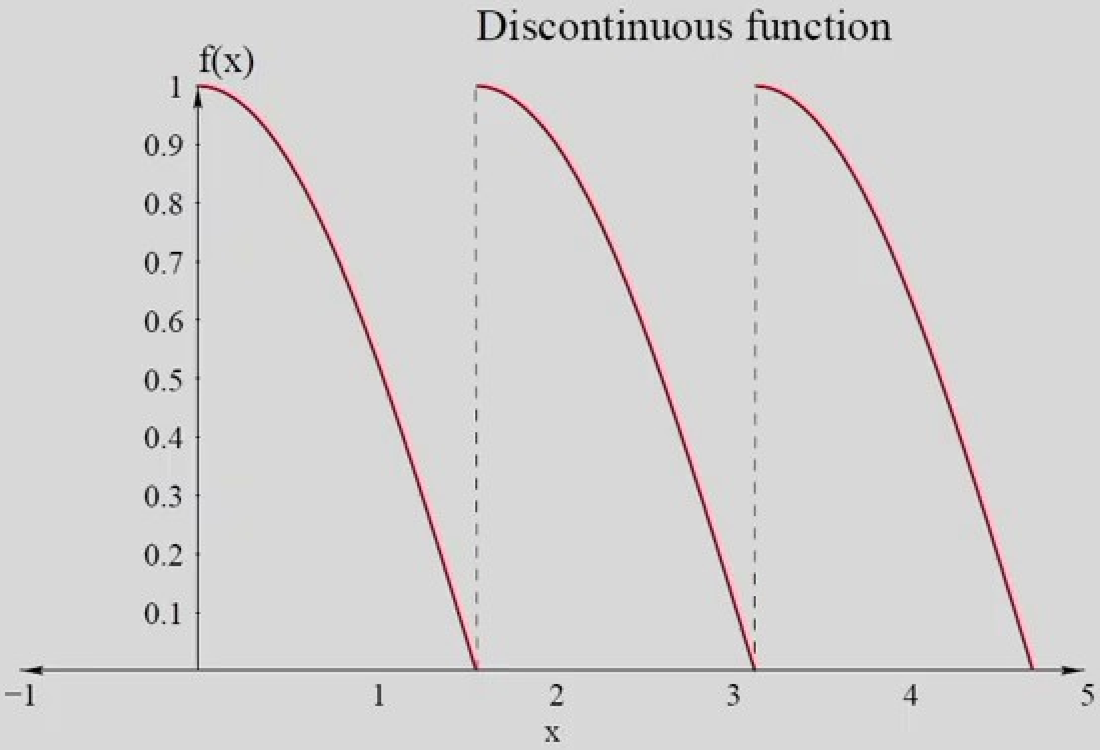
- Non-differentiable functions and constraints



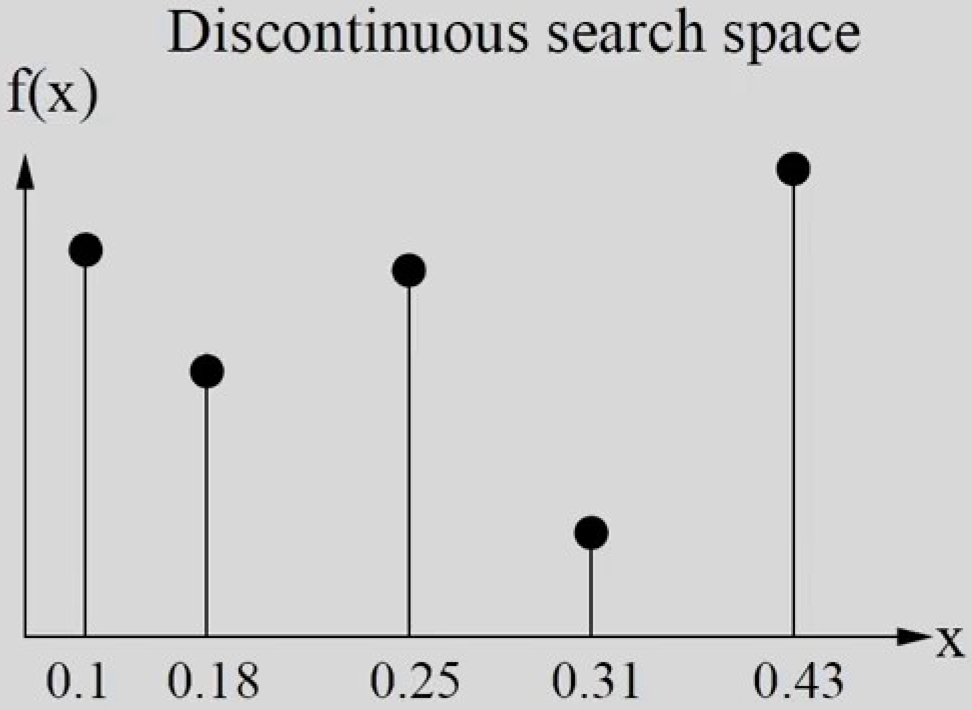


Properties of Practical Optimization Problems

- Discontinuous function

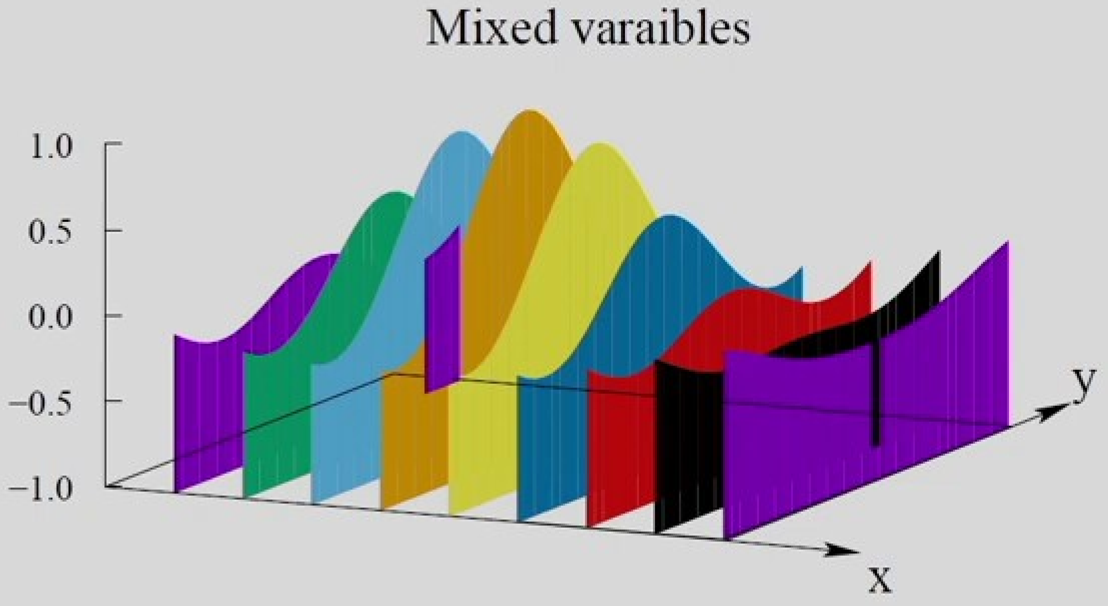


- Discrete/discontinuous search space

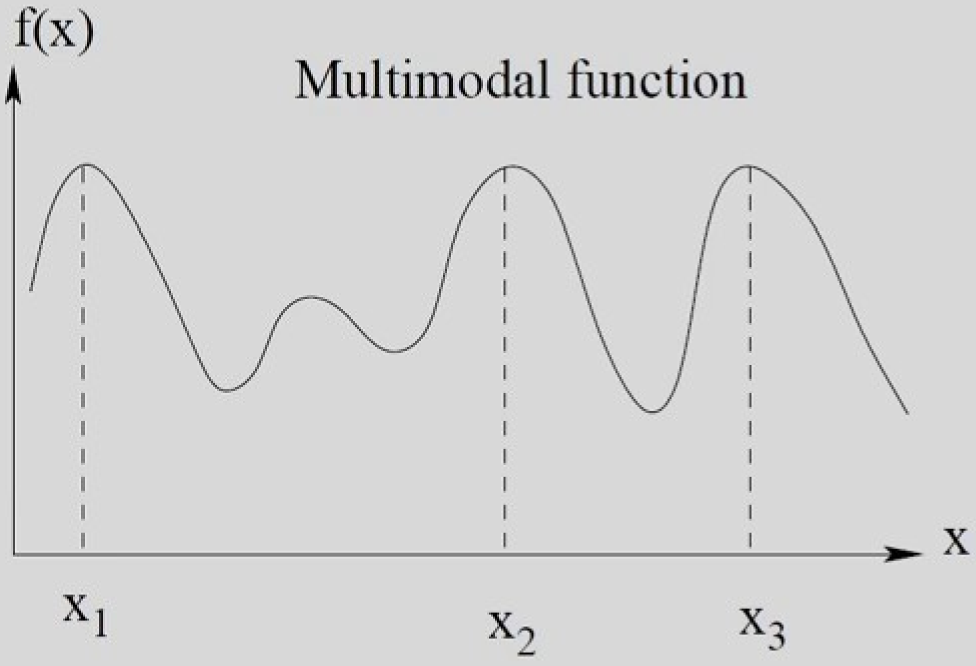


Properties of Practical Optimization Problems

- Mixed variables (discrete, continuous, permutation)

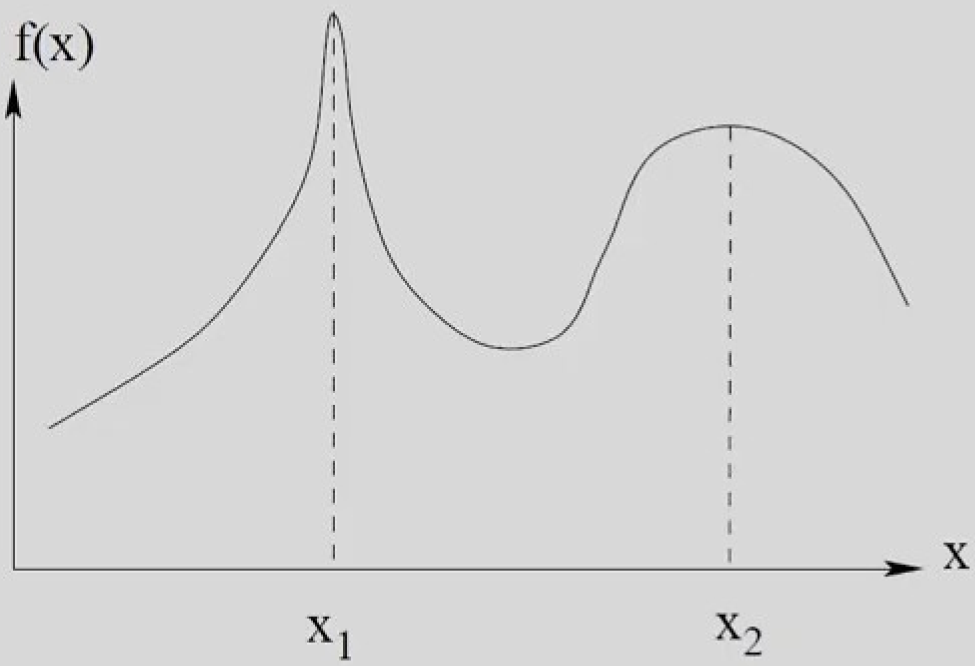


- Multi-modal function

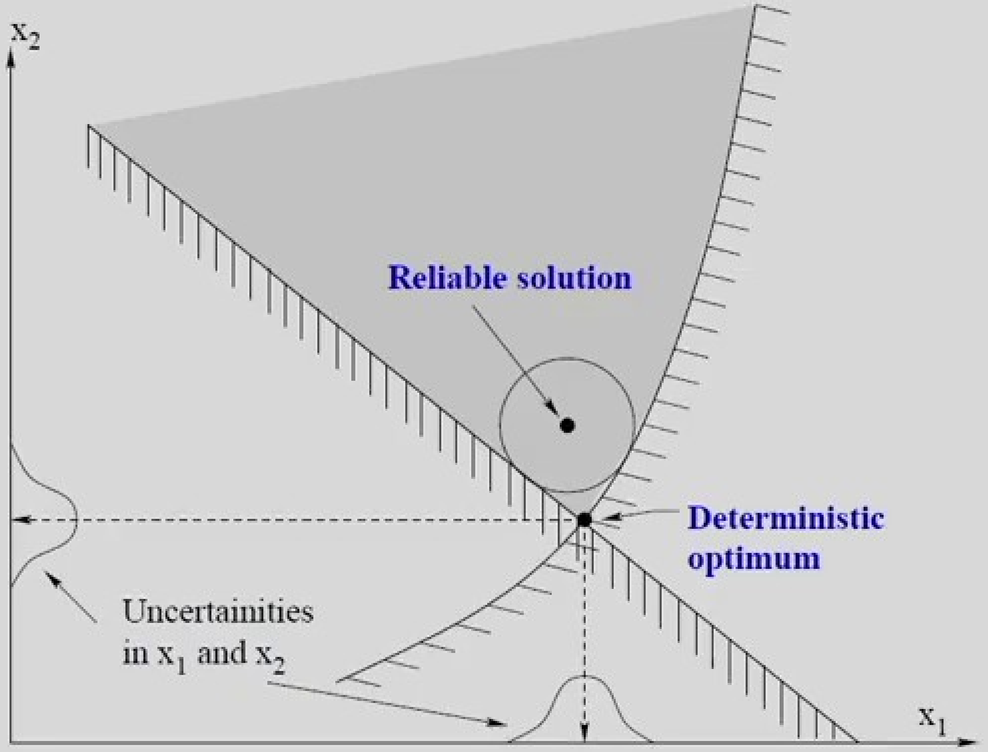


Properties of Practical Optimization Problems

- Robust solution



- Reliable solution



Properties of Practical Optimization Problems

- Non-differentiable functions and constraints
- Discontinuous function and search space
- Discrete search space
- Mixed variables (discrete, continuous, permutation)
- Large dimensional problem (variables, constraints)
- Non-linear constraints
- Multi-modal function
- Multiple objectives
- Uncertainties in variables
- Computationally expensive problems
- Multi-disciplinary optimization

Problem Formulation

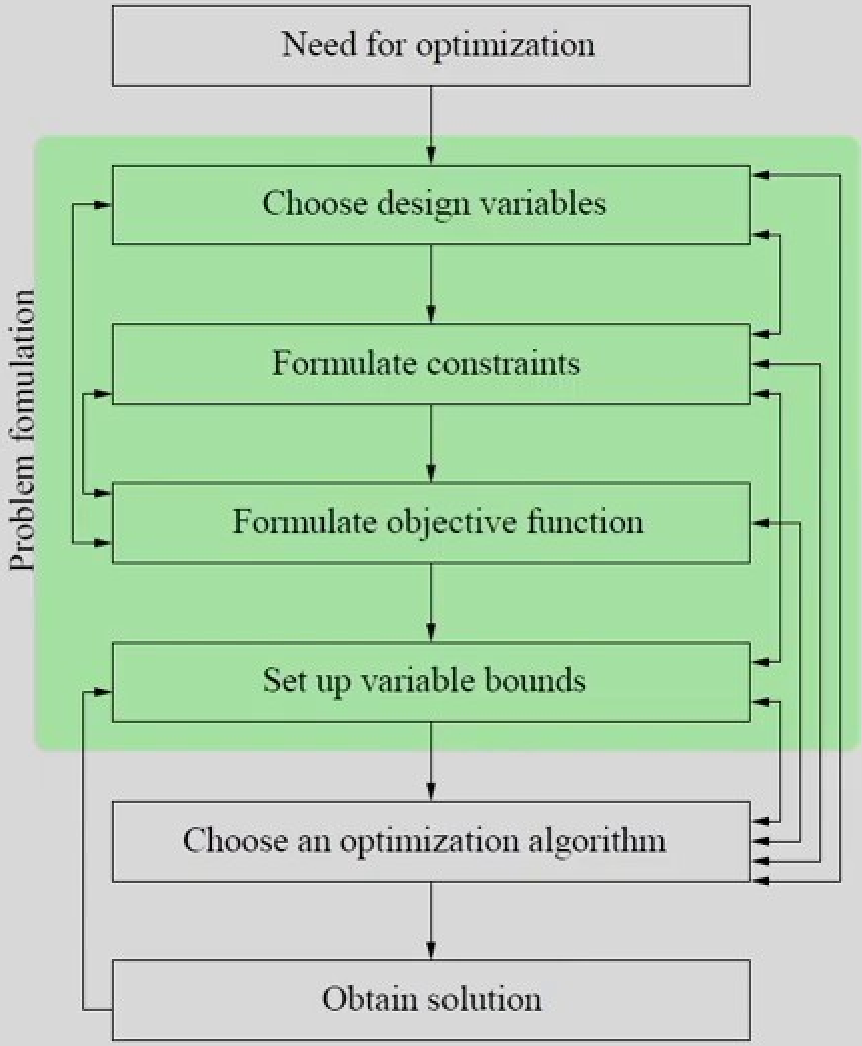
A multi-variable single-objective optimization problem is given as

$$
\begin{array}{lll}
\text{Minimize:} & f(\mathbf{x}), & \\
\text{subject to} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J, \\
 & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K, \\
 & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N.
\end{array}
$$

- $\mathbf{x} = \{x_1, \ldots, x_i, \ldots, x_N\}^T$ is the decision-variable vector.
- $f(\mathbf{x})$ is the objective function.
- $g_j(\mathbf{x})$ is the j-th inequality constraint.
- $h_k(\mathbf{x})$ is the k-th equality constraint.
- $x_i^{(L)}$ and $x_i^{(U)}$ are the lower bound and upper bound on the i-th decision variable.

Steps in an Optimization Task

- Need for optimization
- Problem formulation or modeling
  - Identify problem parameters
  - Choose design variables from parameters
  - Formulate constraints
  - Formulate objective function
  - Set up variable bounds
  - Requires 50-60% of the effort
- Choose an optimization algorithm
- Obtain solution
- Reformulation and rerun if desired



Design Variables

- Identify the underlying design/decision variables

Design variables

- A design problem usually involves many design parameters
  - List any and every parameter related to the problem
- Parameters sensitive to the given design or problem can be considered as design variables in the parlance of optimization procedure
  - Sensitivity analysis, etc.
  - Experience of the users can be used.
- Specify the type of each parameter (binary, discrete, real)
- First thumb rule of an optimization problem: Choose as few variables as possible
  - Efficiency and speed of optimization algorithm depend, to a large extent, on the number of chosen design variables.

Constraints

- Constraints represent limit on certain resource or on certain physical phenomenon, *for example*, satisfy stress limitation, current or voltage restriction, etc.
- Identify the constraints associated with the optimization problem

Types of Constraints

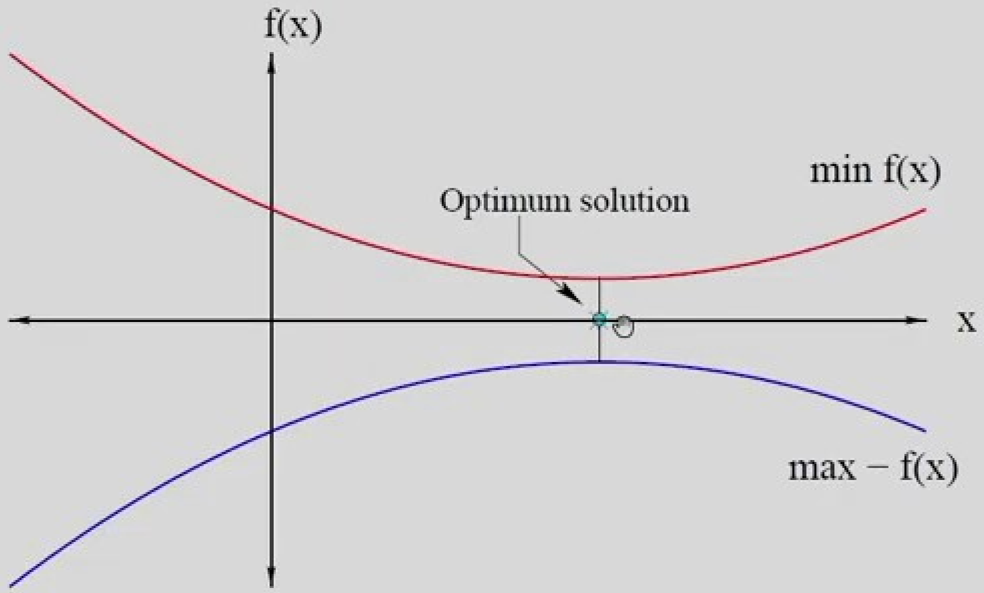
- **Inequality constraint**: $g(\mathbf{x}) \geq 0$ or $g(\mathbf{x}) \leq 0$
  - mostly encounter in engineering design problems
- **Equality constraint**: $h(\mathbf{x}) = 0$, difficult to handle
- Handling of equality constraint, *for example*, deflection of beam, say $\delta(\mathbf{x}) = 0.35$ mm, can be converted it into two inequality, say $\delta(\mathbf{x}) \geq 0.25$ and $\delta(\mathbf{x}) \leq 0.45$
- Second thumb rule in the formulation of optimization problem: The number of complex equality constraints should be kept as low as possible

Objective Function

Objective Function

- Minimize or Maximize $f(\mathbf{x})$, which is written in terms of design variables and other parameters
- Optimization algorithms usually written either for minimization or maximization

- Duality principle helps unification, $\min f(\mathbf{x}) = \max F(\mathbf{x})$, where $F(\mathbf{x}) = -f(\mathbf{x})$



Variable Bounds

Let $\mathbf{x} = (x_1, \ldots, x_i \ldots, x_N)^T$ is the vector of decision variables.

Variable bounds

- Bound on each decision variable: $x_i^{(L)} \leq x_i \leq x_i^{(U)}$ $\forall i = 1, 2, \ldots, N$
- $x_i^{(L)}$ is the lower bound of variable x_i.
- $x_i^{(U)}$ is the upper bound of variable x_i.

Optimization

Optimization Algorithm

It provides systematic and efficient ways of creating and comparing new design solutions in order to achieve an optimized design solution.

- Optimization algorithm works on a mathematical model of the optimal design problem.
- Time consuming and computationally expensive procedure because it requires comparison of a number of design solutions.

Critical Remarks on Numerical Optimization Techniques

- One method is not applicable in many optimization problems
- Constrained handling is sensitive to the penalty parameters
- Not efficient in handling discrete variables
- Local perspective for searching
- Uncertainties in decision and state variables
- Noisy/dynamic optimization problems
- Multiple objectives optimization problems

Need for an innovative and flexible optimization algorithm

Evolutionary Computation Techniques

Outline

1. **Evolutionary Computation**
   - Introduction
   - Principles of EC Techniques
2. **Generalized Framework**
   - Flowchart and Generalized Framework
   - Advantages, Limitations and Differences
3. **Typical Behavior**
   - Performance on 1-dimensional fitness landscape
   - Convergence Plot
   - View as Problem Solver
4. **No Free Lunch Theorem for Optimization**
5. **Closure**

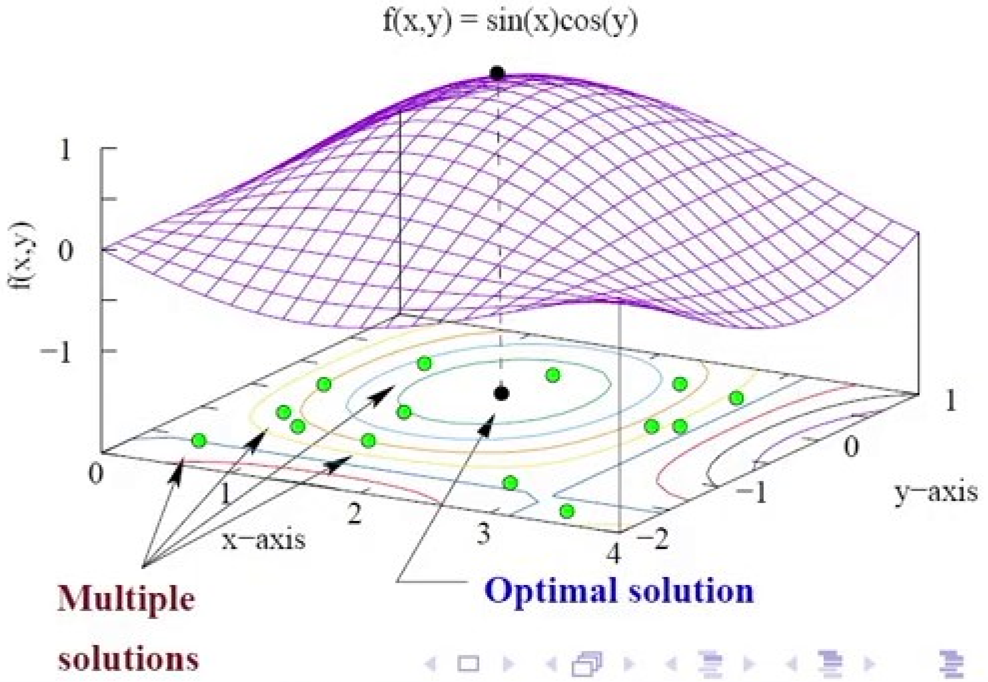
Recap

- Search and optimization
- Applications of optimization
- Properties of optimization problems in practice
- Mathematical problem formulation or modeling
- Remarks on the numerical optimization techniques

Evolutionary Computation: Introduction

- Nature inspired algorithm (<u>but not copied</u>).
  - Mimic natural or biological phenomena or process
- Based on natural evolution + genetics.
  - **Natural evolution**: Offspring (new solutions) are created by variation operators, such as crossover, mutation etc.
  - **Survival of the fittest**: Good solutions are retained and bad are deleted.
  - **Genetics**: Information is coded.

- Evolutionary computation (EC) techniques can be used in optimization, learning and design.
- EC techniques are population-based algorithms.
  - Many solutions or points



Nature as an Optimizer

- Nature as structural engineer
  - Stem, Bamboo, insect trachea, bee-hive
- Nature as a CFD solver
  - Birds, fishes
- Nature as a drag reducer
  - Penguin body

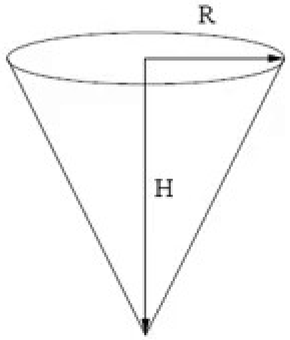
Genetics

Genotype | **Phenotype**

Cone 1

Chromosome = $\{01001\ 01010\}$

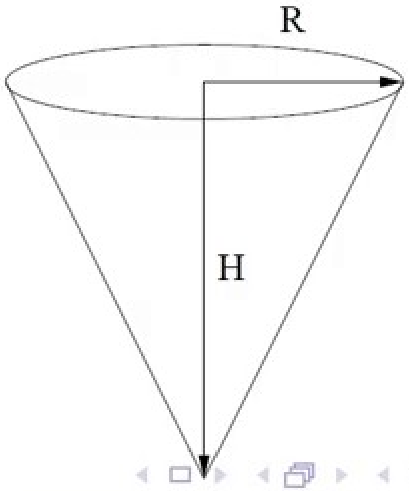
$(R, H) = (9, 10)$cm



Cone 2

Chromosome = $\{01100\ 01111\}$

$(R, H) = (12, 15)$cm



Natural Evolution

- Crossover between two parents at the random site (say at 6th site)

P1:	1 0 1 0 1 1 | 0 0 1 0
P2:	0 1 0 1 0 0 | 0 1 1 0

O1:	1 0 1 0 1 1 | 0 1 1 0
O2:	0 1 0 1 0 0 | 0 0 1 0

- Mutation at the random bit position (say at 4th position)

$$1\ 0\ 1\ \boxed{0}\ 1\ 1\ 0\ 1\ 1\ 0$$

$$\downarrow$$

$$1\ 0\ 1\ \boxed{1}\ 1\ 1\ 0\ 1\ 1\ 0$$

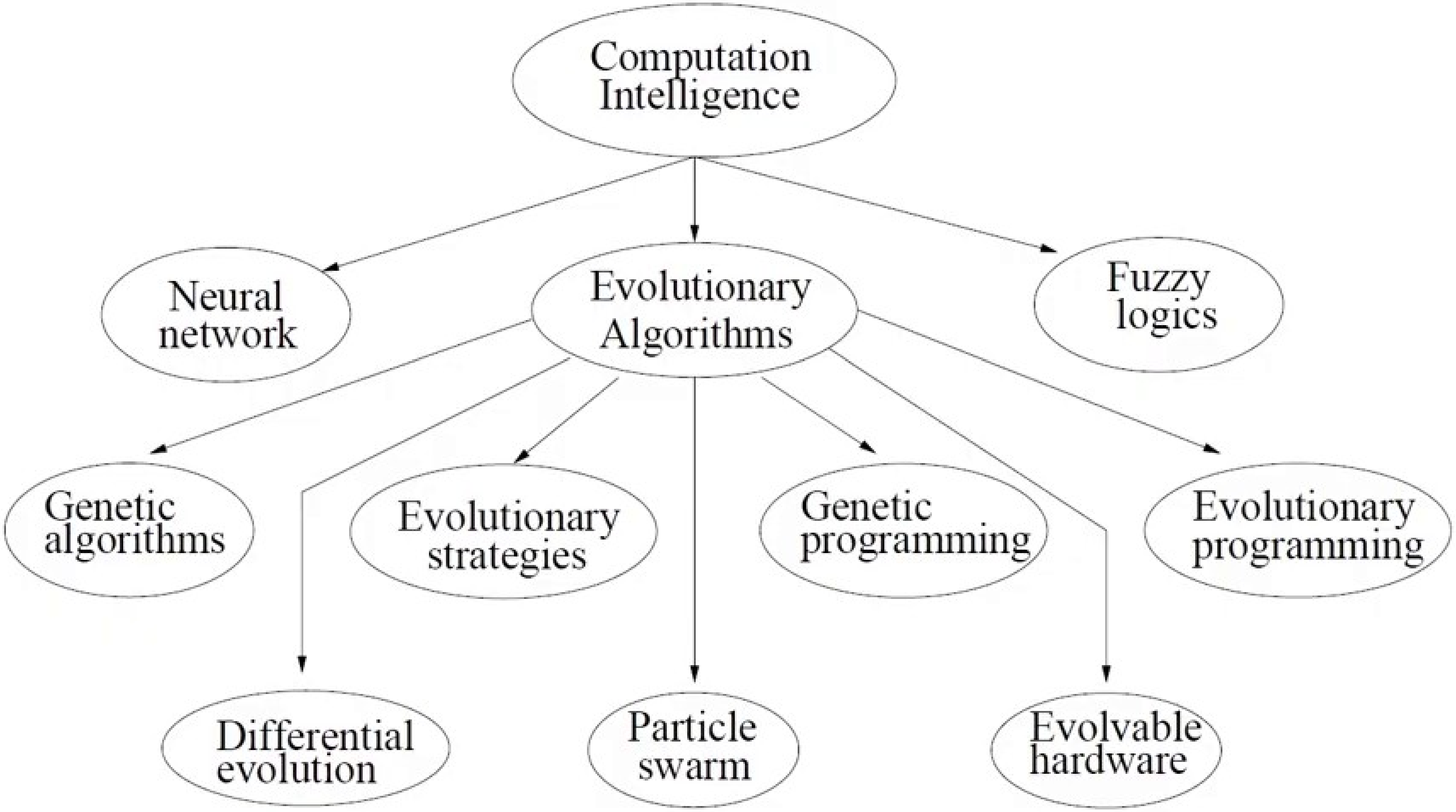
Survival of the Fittest

- Darwin's principle of evolution
  - **Survival of the fittest**: Stronger candidate has more chance of survival than weaker candidates in an environment of limited sources like food, etc.
  - Diversity: Due to crossover, offspring will be evolved from stronger parents
  - Chances are more that offspring will possess good traits of strong parents which can be further improved by mutation
  - **Diversity drives changes** in the population

Generalized Framework

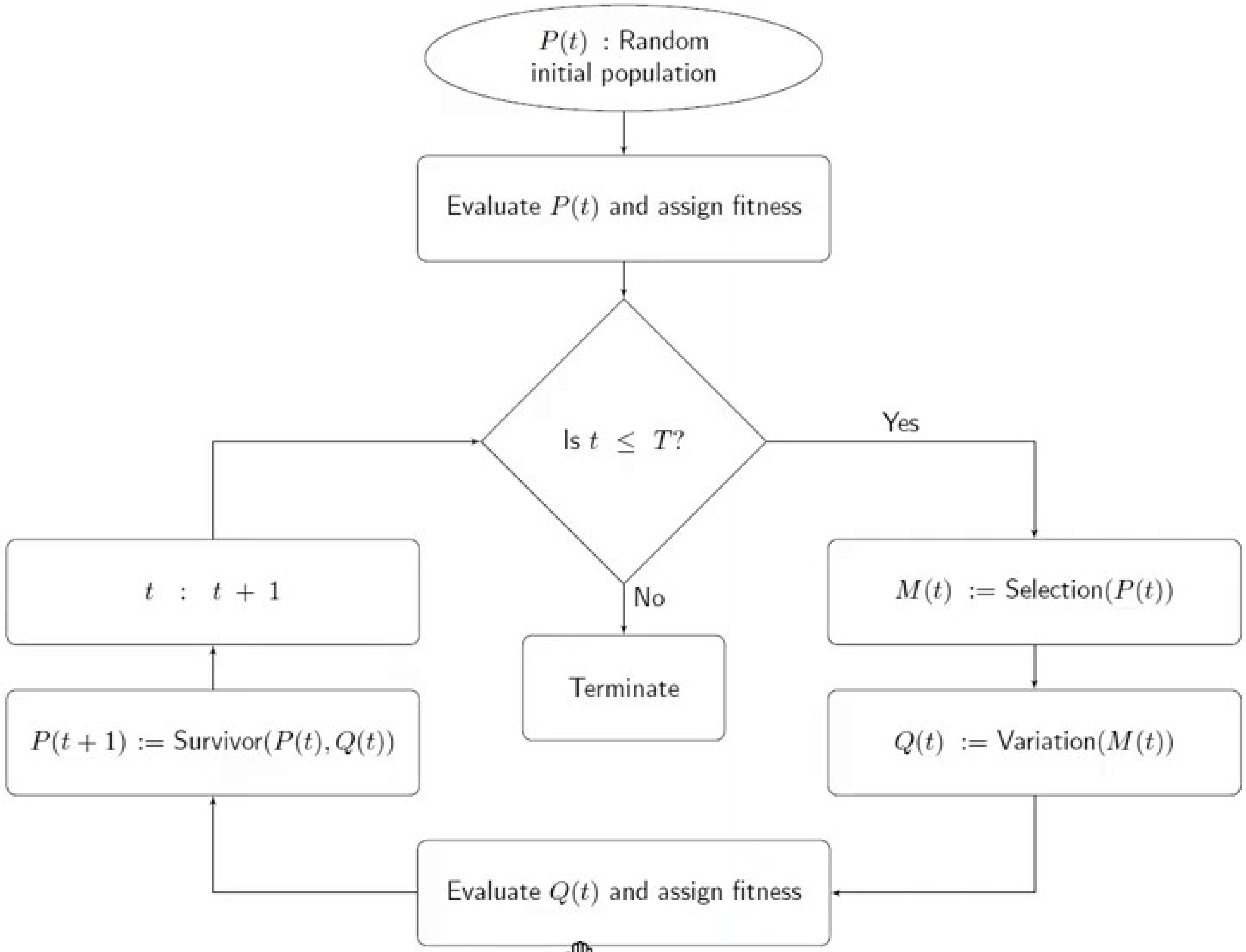
Computational Intelligence and EAs

EC techniques are also referred as evolutionary algorithms (EAs).



We treat EA as a search and optimization tool.

Flowchart of EC Techniques



Generalized Framework of EC Techniques

Algorithm 1 Generalized Framework

Solution representation %Genetics
Input: $t := 1$ (Generation counter), Maximum allowed generation $= T$
Initialize random population $(P(t))$; %Parent population
Evaluate $(P(t))$; %Evaluate objective, constraints and assign fitness
while $t \leq T$ **do**
    $M(t) :=$ Selection$(P(t))$; %Survival of the fittest
    $Q(t) :=$ Variation$(M(t))$; %Crossover and mutation
    Evaluate $Q(t)$; %Offspring population
    $P(t+1) :=$ Survivor$(P(t), Q(t))$; %Survival of the fittest
    $t := t + 1$;
end while

Advantages of EC Techniques

- Applicable in problems where no (good) method is available
  - Multi-modalities, discontinuities, non-linear problems
  - Noisy problems
  - Implicitly defined problems (simulation models)
  - Discrete variable space
- Most suitable in problems where multiple solutions are sought.
  - Multi-modal optimization problems
  - Multi-objective optimization problems
- No presumptions with respect to problem space
- Low development costs, i.e., costs to adapt to new problem spaces
- Parallel implementation is easier for computationally expensive problems

Disadvantage of EC Techniques

- No guarantee for finding optimal solutions in a finite amount of time
  - However, asymptotic convergence proofs are available
- Parameter tuning mostly by trial-and-error
  - Self-adaptation is a remedy
- Population approach may be expensive
  - Parallel implementation is a remedy

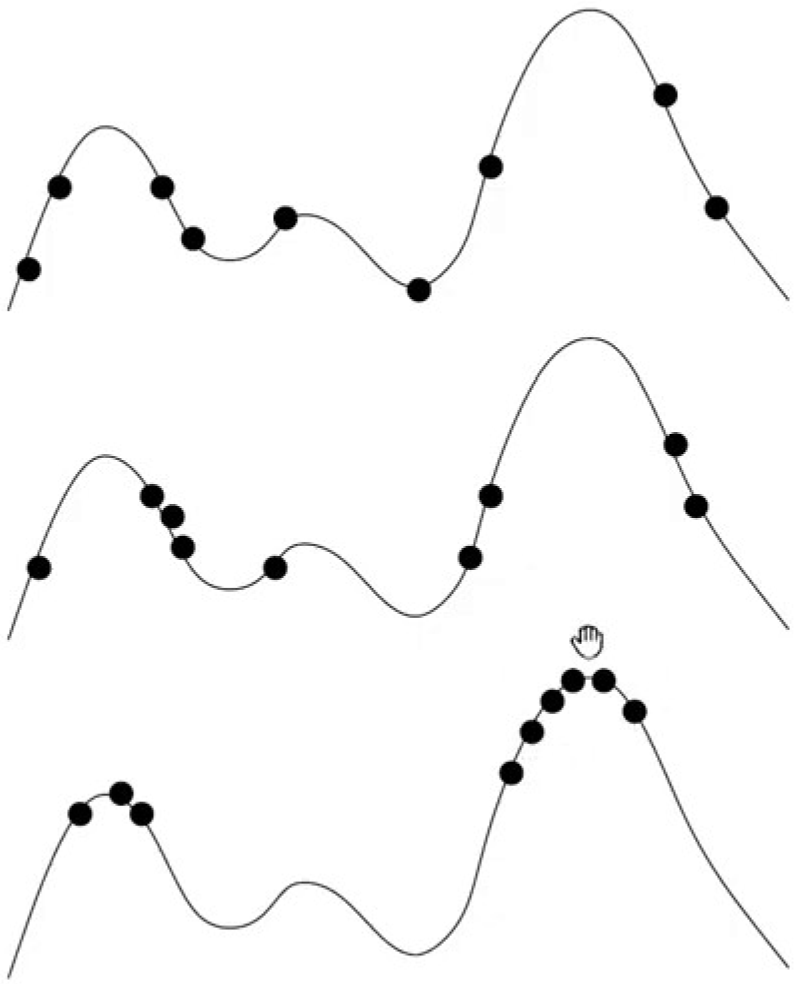
Differences With Numerical Optimization Techniques

- Different types of variables can be used with EC techniques
  - A problem requires mixed-variable can be handled.
  - Variables such as permutation, continuous, discrete can be used together.
- EC techniques are population-based metaheuristics.
  - The number of solutions can provide safety against premature convergence that can lead to a global perspective to EC techniques.
  - Implicit parallelism
- Operators of EC techniques are probabilistic in nature.
  - It reduces the chance of getting stuck at the local optima.
- EC techniques do not require any gradient information.
- EC techniques are ideal for parallel computation.
  - Evaluations can be distributed the computing processors and threads.

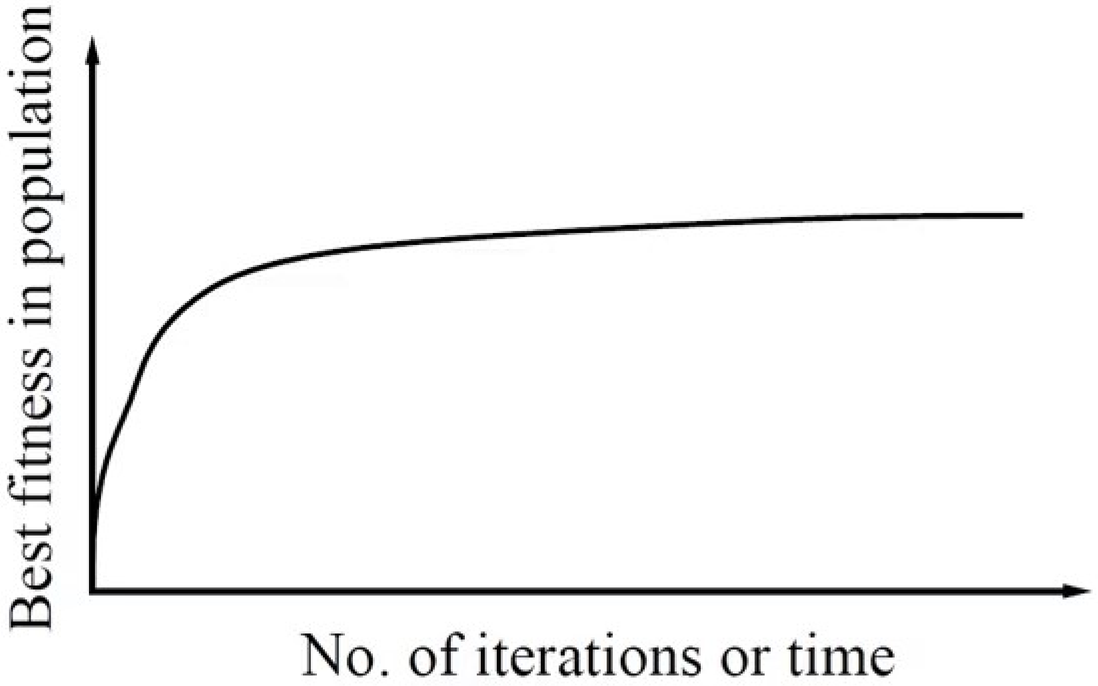
Typical behavior of an EA

Phases in optimization on a 1-dimensional fitness landscape

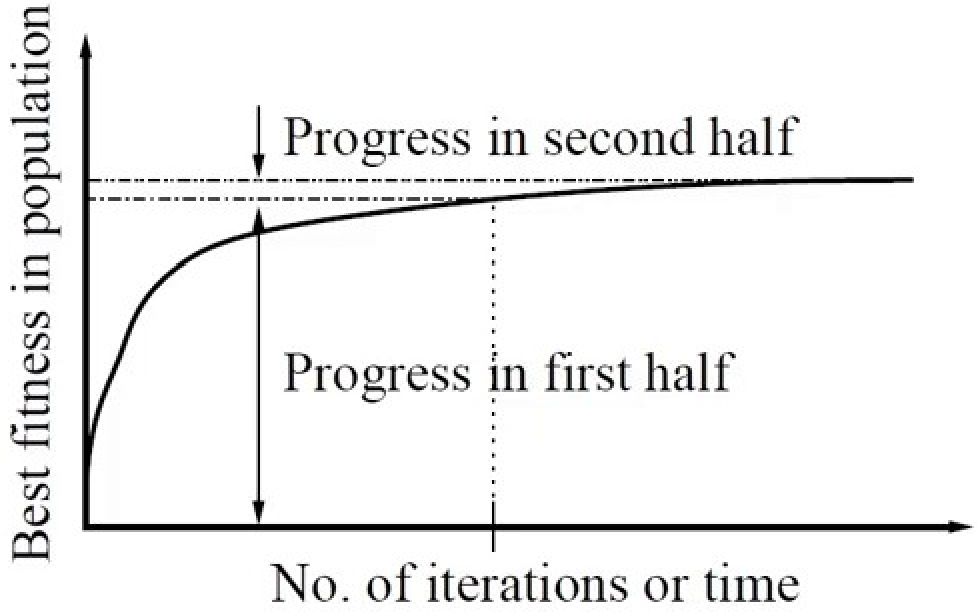
- **Early phase**: quasi-random population distribution
- **Mid-phase**: population arranged around/on hills
- **Late phase**: population concentrated on high hills



Typical Run: Convergence Plot



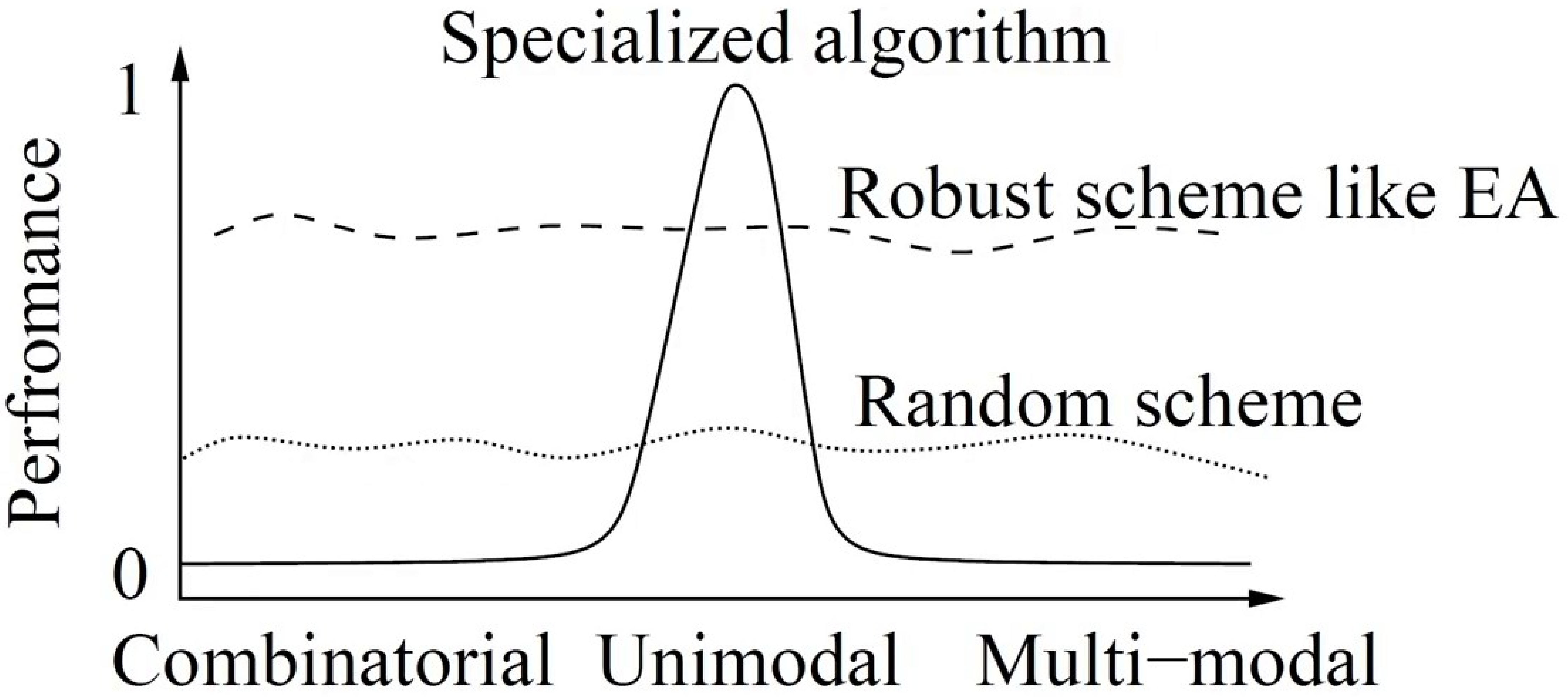
Are long runs beneficial?



Answer:

- It depends how much you want the last bit of progress.
- It may be better to do more shorter runs.

GAs as Problem Solvers: Goldberg's 1989 view



No Free Lunch (NFL) Theorem for Optimization

- In the context of optimization
  - D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, 1(1),67-82, 1997.
  - A framework is developed to explore the connection between effective optimization algorithms and the problems they are solving.
  - For any algorithm, any elevated performance over one class of problems is offset by performance over another class.
  - Algorithms $A1$ and $A2$
  - All possible problems F in one class
  - Performances $P1$ and $P2$ using $A1$ and $A2$ for a fixed number of evaluations
  - $P1 = P2$
- NFL breaks down for a narrow class of problems or algorithms

Closure of Module 1

- Search and optimization
- Applications of optimization
- Properties of optimization problems in practice
- Mathematical problem formulation or modeling
- Remarks on the numerical optimization techniques
- Introduction to evolutionary computation (EC)
- Principles of EC: Genetics, evolution and survival of the fittest
- Generalized framework
- Advantages and limitations of EC techniques
- Behavior of EC run
- No Free Lunch (NLP) Theorem in optimization