UniTs - University of Trieste

Faculty of Scientific and Data Intensive Computing

Department of mathematics informatics and geosciences

Global and Multi-Objective Optimisation

Lecturers:
Prof. Luca Manzoni

Author:
Andrea Spinelli

May 24, 2025

github.com/Spina02 andreaspinelli2002@gmail.com

Preface

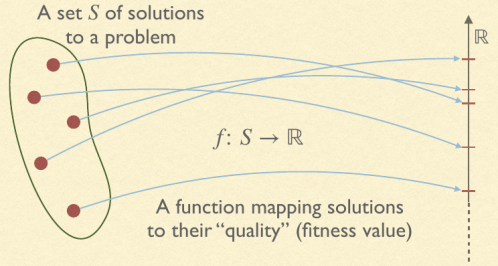
Contents

1 Introduction

1.1 Problem formulation

Given a set S of candidate solutions to an optimization problem, we seek a mapping $f : S \to \mathbb{R}$ which assigns to each solution $x \in S$ a fitness value $f(x)$. Our goal is to find:

$$\arg\max_{x \in S} f(x) \quad \text{or} \quad \arg\min_{x \in S} f(x).$$



In many practical settings it is not possible to solve this problem analytically: the search space S may be exponentially large, the function f may be a "black-box" (we have few assumptions on its smoothness or structure), and an exhaustive enumeration of all solutions can be infeasible. In such cases, we aim instead for heuristics that return solutions of acceptable quality in reasonable time.

1.1.1 A simple illustrative example: OneMax

As a motivating example, let $S = \{0,1\}^n$ and define:

$$f(x) = \text{the number of ones in } x.$$

Clearly the global maximiser is the string 1^n, with fitness n. Even this trivial problem becomes intractable for large n if approached by brute-force enumeration.

Random search

A simplest stochastic approach is random search: pick an initial $b \in S$, then repeatedly sample:

$$x \sim \text{Uniform}(S),$$

and if $f(x) \geq f(b)$ replace b by x. Terminate when a budget of evaluations is exhausted. In the worst case this explores a constant fraction of S, which is equivalent to an exhaustive search in some enumeration order, and so scales poorly in practice.

Tip: ***Random search***

Even if repeated samples are avoided, random search still requires sampling a significant fraction of the space, so it is generally unfeasible for high-dimensional or combinatorial domains.

Hill climbing

Hill climbing maintains a single incumbent solution b. At each iteration we choose a neighbour x of b (according to some neighbourhood structure) and replace b with x if $f(x) \geq f(b)$. The process stops when no improving neighbour can be found or a fixed evaluation budget is reached.

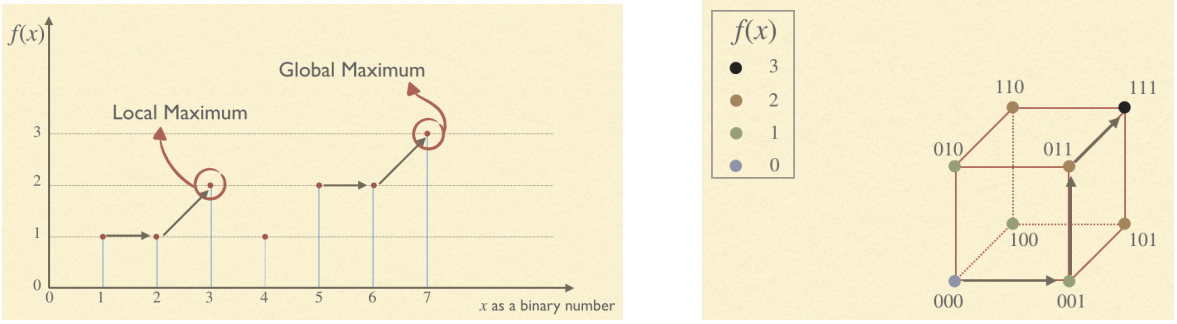


Figure 1.1: With a poor neighbourhood (left), hill climbing can get trapped in a local optimum. A richer neighbourhood (right) may eliminate local traps.

The effectiveness of hill climbing depends critically on the choice of neighbourhood. For `OneMax`, using the Hamming-1 neighbourhood (flip one bit at a time) guarantees reachability of the global optimum but may still require many steps; using only ± 1 on the integer-interpreted string can make the problem insoluble.

Simulated annealing

Simulated annealing augments hill climbing with occasional downhill moves to escape local optima. Starting from $b \in S$ and a "temperature" T, at each step we pick a neighbour x. If $f(x) \geq f(b)$ we accept it, otherwise we accept it with probability

$$\exp\big((f(x) - f(b))/T\big).$$

We then decrease T according to a cooling schedule. Proper tuning of the schedule trades off exploration against exploitation.

Tip: ***Simulated annealing***

Allowing uphill moves with probability depending on the temperature and fitness gap helps avoid entrapment in local maxima. The cooling schedule is crucial for performance.

Multiple restarts and population-based search

Both hill climbing and simulated annealing can be repeated from fresh random starts to reduce the chance of permanent stagnation. A more powerful paradigm uses a whole population of candidate solutions that "interact" (e.g. by recombination), leading naturally to evolutionary algorithms.

2 Genetic Algorithms

2.1 Introduction

Genetic Algorithms (GAs) are a class of stochastic optimization methods inspired by the principles of natural selection and genetics, first formalized by John Holland in the 1970s [1]. At their core, GAs maintain a population of candidate solutions, called ***individuals***, which are evolved over multiple generations to approximate an optimal or sufficiently good solution to a problem.

Each individual in the population is typically represented as a fixed-length string (often a binary vector) termed the ***genotype***. This genotype encodes a possible solution, whose ***phenotype*** is the actual candidate in the problem space, obtained by decoding the genotype. The quality of each individual is measured by a ***fitness function*** f, which assigns a scalar value indicating how well the individual solves the problem at hand.

The standard evolutionary cycle in a GA consists of the following steps:

1. **Selection:** Individuals are chosen probabilistically from the population based on their fitness. Fitter individuals have a higher probability of being selected as parents, implementing a form of artificial natural selection that drives evolution toward better solutions.
2. **Crossover:** Pairs of selected parents exchange genetic material to produce offspring, recombining their genotypes in various ways. This operator enables the algorithm to combine beneficial traits from different solutions and explore the search space effectively.
3. **Mutation:** Random, typically small, modifications are introduced into the offspring's genotypes to preserve genetic diversity and explore new regions of the search space. This operator helps prevent premature convergence and allows the discovery of novel solutions.
4. **Replacement:** The new generation replaces the old one, possibly retaining a fraction of the best solutions (elitism). This ensures the population maintains its best-found solutions while allowing for continuous improvement through evolution.

This process iterates for a predefined number of generations or until a stopping criterion is met. A summary of the cycle is illustrated in Figure 2.1.

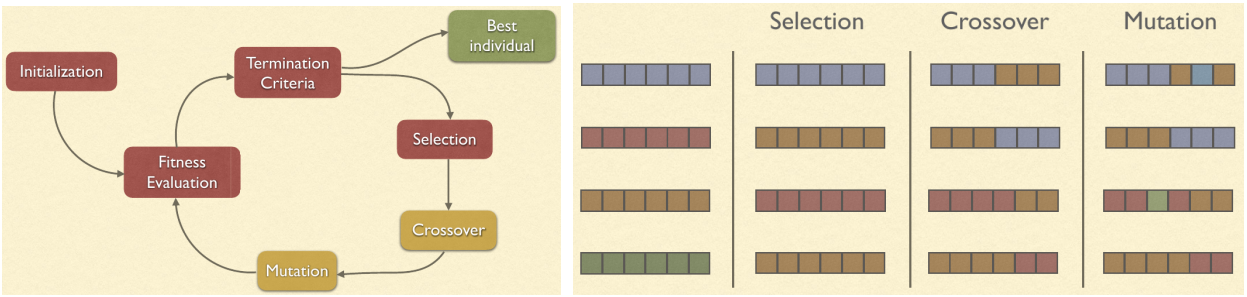


Figure 2.1: Left: Diagram showing the main components and their interactions in the evolutionary process. Right: Example of genetic operators (selection, crossover, mutation) acting on binary strings.

2.2 Core Components

Representation

The **genotype** is the encoded representation of a solution (commonly a binary string or vector over a finite alphabet) on which the genetic operators (crossover and mutation) act. The **phenotype** is the decoded, actual candidate solution evaluated by the fitness function. Selection operates at the phenotypic level, favoring those solutions that yield higher fitness.

Observation: ***Genotype vs. Phenotype***

The distinction between the two allow GAs to operate flexibly: operators modify representations (genotypes) while selection is based on problem-specific performance (phenotypes).

Key Parameters

The performance and behavior of a genetic algorithm depend on several critical parameters:

- **Population size** N: Number of individuals maintained at each generation (tipically 100-200). Larger populations increase diversity and exploration but require more computational resources.
- **Number of generations** G: How many iterations the algorithm will perform (often determined empirically). This affects the total computational budget and exploration time.
- **Selection method**: The algorithm used to select parents (tournament, roulette wheel, ...). Different methods vary the selection pressures that affect diversity and convergence speed.
- **Crossover operator**: The method for recombining genotypes. Should be chosen based on problem representation and known/assumed relationships between genes.
- **Crossover probability** p_{cross}: Probability with which crossover is applied. Higher values (typically 0.6-0.9) promote more exploration through recombination.
- **Mutation operator**: The method for introducing random changes. Must be appropriate for the chosen representation while maintaining solution feasibility.
- **Mutation probability** p_{mut}: Probability of mutating each gene. Usually set to $1/n$ for length-n genotypes to maintain a balance between exploration and stability.
- **Elitism** e: Number or percentage of best individuals preserved into the next generation. Small values (1-5%) help maintain good solutions while allowing population turnover.

2.2.1 Selection Methods and Genetic Operators

Selection

Several strategies exist for parent selection, each with different characteristics in terms of selection pressure and diversity preservation:

- ***Roulette Wheel Selection:***
  Each individual's probability of being selected is proportional to its fitness:

$$P_{x,P} = \frac{f(x)}{\sum_{y \in P} f(y)}$$

  This method is simple to implement, but can reduce diversity if a single individual dominates.
- ***Ranked Selection:***
  Individuals are ranked by fitness. Selection probabilities depend only on rank, not raw fitness, which helps control selection pressure and maintain diversity.

- ***Tournament Selection:***
  t individuals are sampled (with replacement) from the population, and the fittest among them is selected. The tournament size t directly tunes *selection pressure*: higher t increases the probability that the best individuals are chosen.

Tip: ***Selection Pressure***

Tournament selection is widely used due to its simplicity and ease of adjusting selection pressure by changing the tournament size.

Crossover

Crossover operators create new individuals by combining genetic material from two parents:

- ***One-Point Crossover:***
  A single crossover point k is randomly chosen between genes. The offspring inherit genes from parent A up to position k, and from parent B beyond k. This preserves contiguous gene sequences that may represent important building blocks:

  Parent A: `[1 1 1 | 1 1 1]`
  Parent B: `[0 0 0 | 0 0 0]`
  Offspring: `[1 1 1 | 0 0 0]`

- ***Multi-Point Crossover:***
  Multiple crossover points are chosen, and genetic material is alternately swapped between parents. This allows more flexible recombination while still preserving some gene linkage:

  Parent A: `[1 1 | 1 1 | 1 1]`
  Parent B: `[0 0 | 0 0 | 0 0]`
  Offspring: `[1 1 | 0 0 | 1 1]`

- ***Uniform Crossover:***
  For each gene position, the gene is swapped between parents with a fixed probability (commonly $1/2$). This provides maximum mixing potential and is especially useful when there is little/no linkage between adjacent genes:

  Parent A: `[1 1 1 1 1 1]`
  Parent B: `[0 0 0 0 0 0]`
  Offspring: `[1 0 1 0 0 1]`

The choice of crossover operator and its probability p_{cross} influences the balance between ***exploration*** and ***exploitation*** in the search process.

Tip: ***Crossover Design***

Select the crossover operator based on the structure of the problem representation. For representations where tightly coupled genes are adjacent, one-point crossover is often effective. For others, uniform crossover can better promote exploration.

Mutation

Mutation introduces random changes to individuals, preserving genetic diversity and enabling the exploration of new areas in the search space. The most common operator for binary representations is the ***bit-flip mutation***: for each gene, flip its value with probability p_{mut} (typically $1/n$ for length-n genotypes, so that on average one bit per individual mutates per generation).

2.2.2 Common Variants

Several variations of the basic GA have been developed:

- **Elitism**
  Strategies that preserve the best individual(s) unchanged into the next generation, guaranteeing solution quality does not degrade. Variants include retaining the single best solution, top k solutions, or best $p\%$.
- **Steady-State GA**
  Instead of replacing the entire population each generation, only a subset of individuals is replaced. The choice of which individuals to replace impacts algorithm dynamics.
- **Hybrid (Memetic) Algorithms**
  These incorporate local search techniques to further improve individuals after genetic operations. Also called *Lamarckian algorithms* or *Baldwin effect algorithms*, they require tuning of local search frequency and intensity.

2.3 Representation

While we have focused on binary representations so far, genetic algorithms can be generalized to work with symbols from any finite alphabet Σ instead of just binary values. When using a larger alphabet, mutation needs to be adapted - rather than simply flipping bits, it selects uniformly between the $|\Sigma| - 1$ alternative symbols. For ordered alphabets like $\{0, 1, 2, 3\}$, mutation can also be implemented to increment or decrement values, maintaining the ordering relationship.

2.3.1 Real-Valued GA

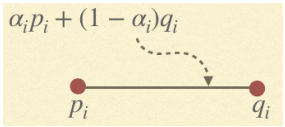
Traditional binary GAs can encode floating-point numbers using 32 or 64 binary genes, where different bit positions have varying impacts on the final value. However, real-valued GAs take a more direct approach by using floating-point genes directly and adapting the genetic operators accordingly.

Crossover

Real-valued GAs employ two main specialized crossover operators:

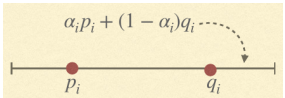
- **Intermediate recombination:** Creates offspring by taking weighted averages of the parents' genes. Given parents x_1 and x_2:

$$y_i = \alpha_i p_i + (1 - \alpha_i) q_i, \quad \alpha_i \sim \text{Uniform}(0, 1)$$



- **Line recombination:** Extends intermediate recombination by allowing exploration beyond parents:

$$y_i = \alpha_i p_i + (1 - \alpha_i) q_i, \quad \alpha_i \sim \text{Uniform}(-k, 1 + k)$$



Mutation

For real-valued representations, mutation operates by adding small perturbations to each coordinate.

$$p \leftarrow p + \varepsilon$$

These perturbations ε can be drawn from either a *uniform distribution* within specified bounds or a *Gaussian distribution* centered at the current value. The choice between these distributions affects how mutation explores the search space.

2.3.2 Permutation-Based GA

Many optimization problems involve finding optimal permutations of elements $\{1, \ldots, n\}$. These problems require specialized genetic operators that preserve the permutation constraints. While mutation typically operates by simply swapping two positions, crossover requires more sophisticated approaches.

Partially Mapped Crossover (PMX)

PMX is a sophisticated crossover operator that preserves permutation validity through a four-step process:

1. Two crossover points are selected in the parent permutations
2. A mapping is constructed between the segments defined by these points
3. The segments are exchanged between parents
4. Any conflicts are resolved by iteratively applying the mapping until a valid permutation is obtained

Cycle Crossover

Cycle crossover provides an alternative approach that preserves absolute positions from the parents. It operates by identifying and preserving cycles in the permutations: starting at a position i, it copies the value from the first parent, then finds that same value in the second parent, continuing until a cycle is completed. The remaining values are then copied from the second parent.

2.3.3 Graph Representation

Graphs can be represented in genetic algorithms using either direct or indirect encoding approaches.

Direct encoding methods include:

- **Adjacency Matrix:** A binary matrix indicates the presence or absence of edges between vertices
- **Edge List:** Explicitly evolve sets of vertices V and edges E

With edge lists, mutations can add or remove both edges and vertices (along with their associated edges).

Tip: ***Graph Crossover***

Graph representations pose particular challenges for crossover operations. In many cases, it may be more effective to rely solely on mutation operators rather than attempting to implement complex crossover schemes.

Indirect encoding offers an alternative approach using production rules that map non-terminal symbols to sequences or matrices of terminals and non-terminals. These rules are repeatedly applied until only terminal symbols remain, enabling compact representations of complex graph structures. This approach can be particularly effective for problems where graphs exhibit regular or hierarchical patterns.

Bibliography

[1] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, Apr. 1992. ISBN: 9780262275552. DOI: 10.7551/mitpress/1090.001.0001. URL: http://dx.doi.org/10.7551/mitpress/1090.001.0001.