IAS0031 Modeling and Identification

Lecture 9: Global Optimization. Symbolic Regression

Aleksei Tepljakov, Ph.D.

Optimum

What is an optimum?

Global optimization is a branch of applied mathematics and numerical analysis that focuses on optimization.

The goal of global optimization is to find the best possible elements $x^\star \in \mathbb{X}$ according to a set of criteria $F = \{f_1(x), f_2(x), .., f_n(x)\}$. These criteria are expressed as mathematical functions, the so-called objective functions.

We consider two cases:

- Single objective functions,
- Multiple objective functions.

Optimum

A **global maximum** $\hat{x}_l \in \mathbb{X}$ of one (objective) function $f : \mathbb{X} \mapsto \mathbb{R}$ is an input element with

$$f(\hat{x}_l) \geqslant f(x) \, \forall x \in \mathbb{X}. \tag{1}$$

A **global minimum** $\check{x}_l \in \mathbb{X}$ of one (objective) function $f : \mathbb{X} \mapsto \mathbb{R}$ is an input element with

$$f(\check{x}_l) \leqslant f(x) \, \forall x \in \mathbb{X}. \tag{2}$$

A **global optimum** $x_l^\star \in \mathbb{X}$ of one (objective) function $f : \mathbb{X} \mapsto \mathbb{R}$ is either a global maximum or a global minimum.

Objective Functions

For a single criterion f, the optimum is its maximum or minimum.

In global optimization the optimization problems are most often defined as minimizations and if criterion f is subject to maximization, we minimize its negation $(-f)$.

The **optimal set** $\mathbf{X}^\star$ is the set that contains all optimal elements. There are normally multiple, often infinitely many optimal solutions.

The tasks of global optimization algorithms are to find solutions to the problem that are

1. as good as possible given the provided *cost*, *bounds*, and *constraints*, and
2. widely different from each other [1].

Multiple Objective Functions

Algorithms designed to optimize sets of objective functions (sets F consisting of $n = |F|$ objective functions f_i, each representing one criterion to be optimized) are usually named with the prefix Multi-Objective.

Multi-objective optimization often means to find a feasible compromise between conflicting goals.

To find the global optimum could mean to maximize one function $f_i \in F$ and to minimize another one $f_j \in F, (i \neq j)$. Hence, it makes no sense to talk about a global maximum or a global minimum in terms of multi-objective optimization. Thus notation of the set of optimal elements is $\mathbf{x}^\star \in \mathbf{X}^\star \subseteq \mathbb{X}$.

Since compromises for conflicting criteria can be defined in many ways, there exist multiple approaches to define what an optimum is. These different definitions, in turn, lead to different sets $\mathbf{X}^\star$.

Weighted Sum

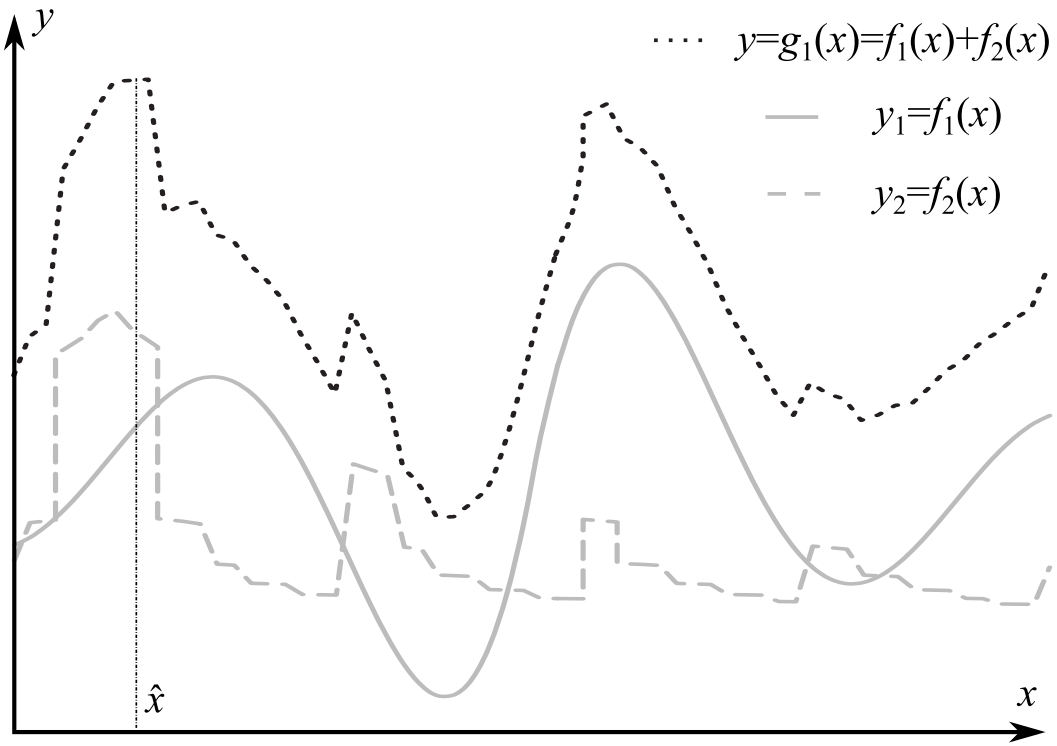
The simplest method to define what is optimal is computing a weighted sum $g(x)$ of all the functions $f_i(x) \in F$. Each objective f_i is multiplied with a weight w_i representing its importance.

$$g(x) = \sum_{i=1}^{n} w_i f_i(x) = \sum_{\forall f_i \in F} w_i f_i(x). \tag{3}$$

Using signed weights also allows us to minimize one objective and to maximize another.

Weighted Sum Method: Illustration

Multi-objective problems are reduced to single-objective ones by this method.



Domination

Pareto optimality defines the frontier of solutions that can be reached by trading-off conflicting objectives in an optimal manner. From this front, a decision maker can finally choose the configurations that, in his opinion, suit the problem best.

Domination. An element x_1 dominates (is preferred to) an element x_2 (notation: $x_1 \vdash x_2$), if x_1 is better than x_2 in at least one objective function and not worse with respect to all other objectives:

$$x_1 \vdash x_2 \Leftrightarrow \forall i : 0 < i \leqslant n \Rightarrow w_i f_i(x_1) \leqslant w_i f_i(x_2) \wedge \exists j : 0 < j \leqslant n : w_j f_j(x_1) < w_j f_j(x_2), \quad (4)$$

where

$$w_i = \begin{cases} 1, & \text{if } f_i \text{ should be minimized} \\ -1, & \text{if } f_i \text{ should be maximized.} \end{cases}$$

Contrary to the weighted sum approach, the factors w_i only carry sign information which allows us to maximize some objectives and to minimize some others.

Pareto Optimality

An element $\mathbf{x}^\star \in \mathbb{X}$ is **Pareto optimal** if it is not dominated by any other element in the problem space $\mathbb{X}$. In terms of Pareto optimization, $\mathbf{X}^\star$ is called the **Pareto set** or the **Pareto frontier**.

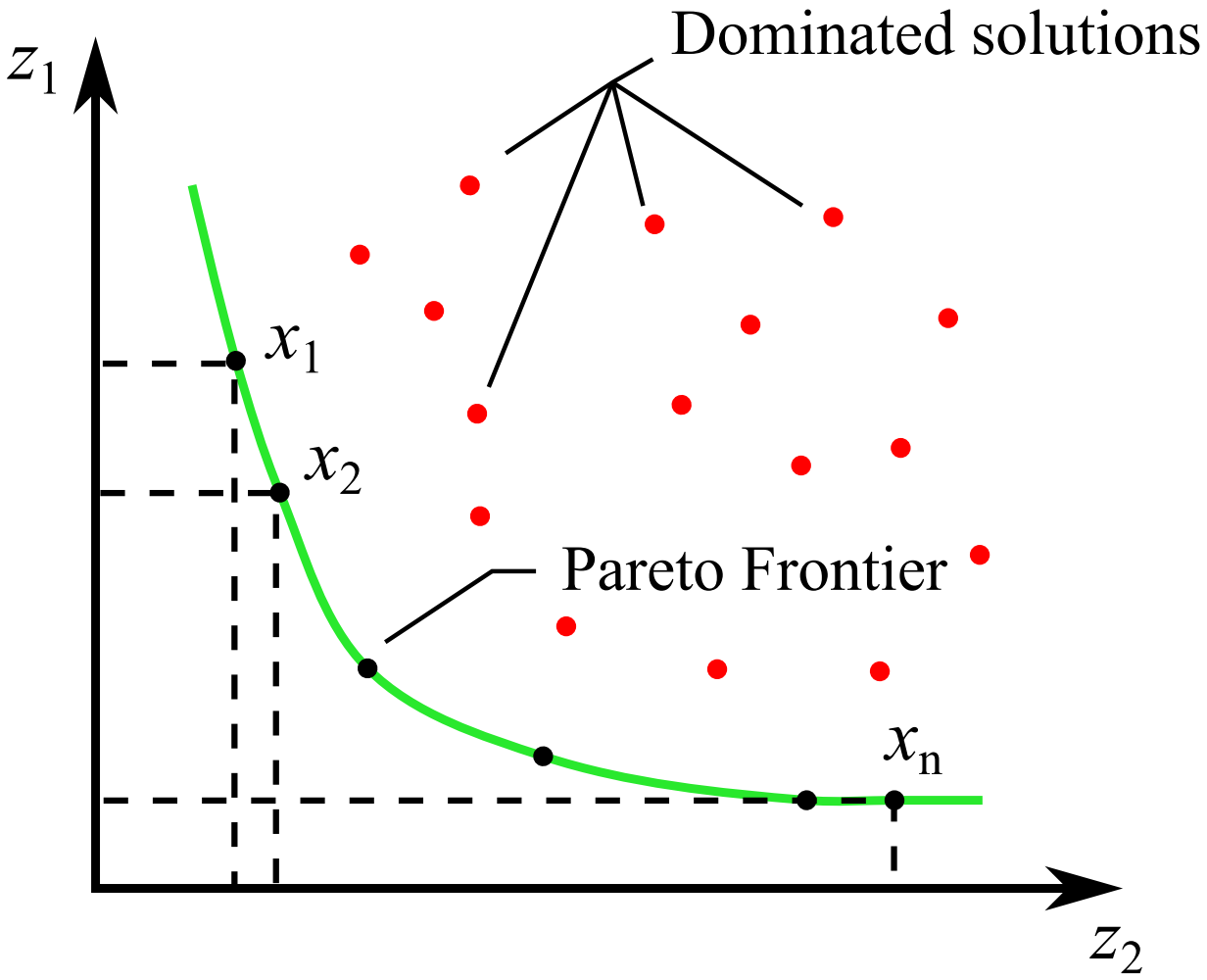
$$\mathbf{x}^\star \in \mathbf{X}^\star \Leftrightarrow \nexists \mathbf{x} \in \mathbb{X} : \mathbf{x} \vdash \mathbf{x}^\star. \tag{5}$$

Consider a minimization problem and the set of solutions that are

- dominated by some other solutions;
- not dominated by any other solution—the Pareto front.

An illustration is provided on the following slide.

Pareto Optimality (continued)



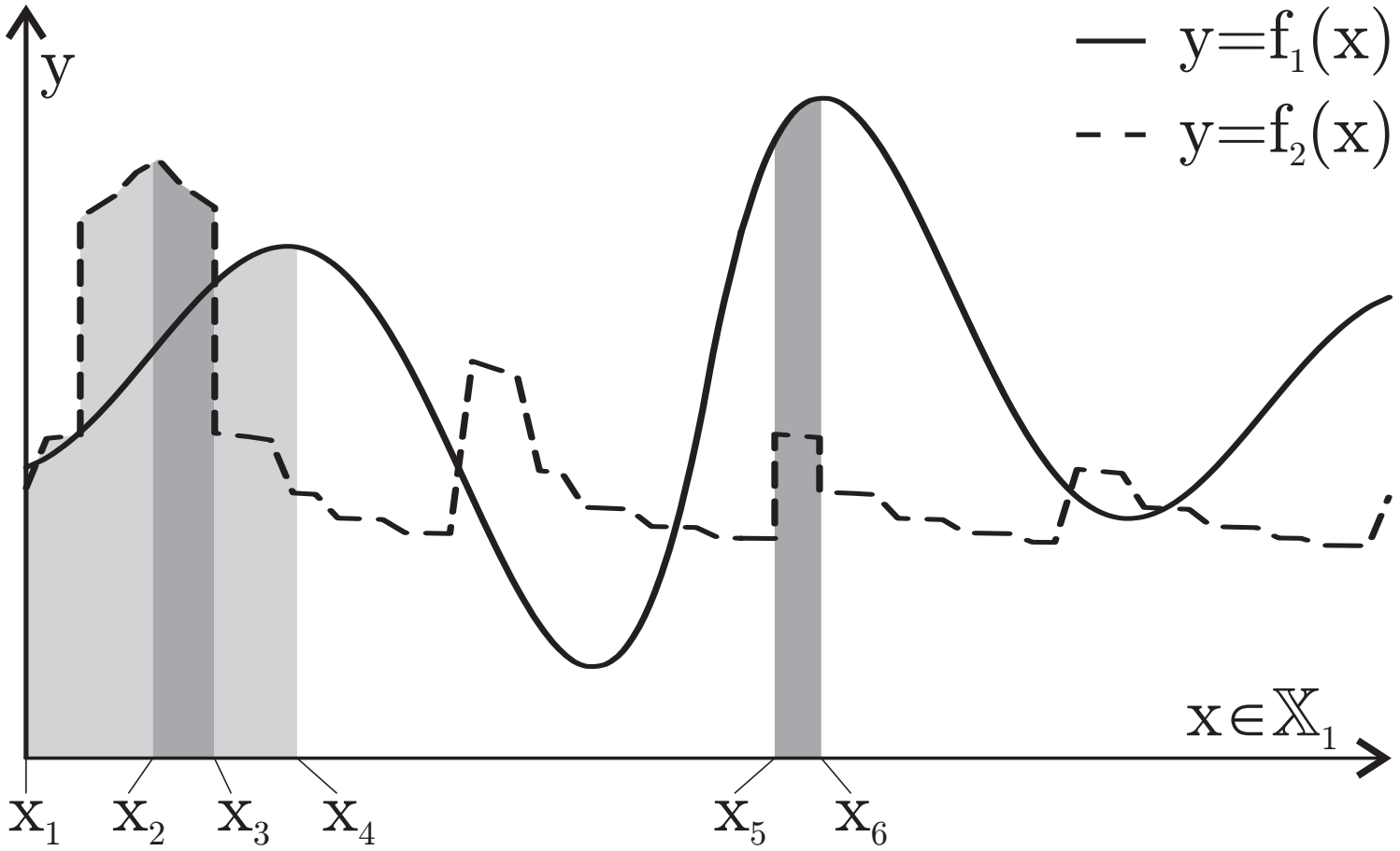
Pareto Optimality: Example

Let us now consider an example due to Weise [2]. The task is to maximize a multi-objective problem $\{\mathbf{f}_1, \mathbf{f}_2\}$. In the following image, dark gray shaded areas are pareto optimal. Light gray shaded areas are dominated solutions specifically related to the dominating ones (nonshaded areas of the plot contain dominated solutions as well).

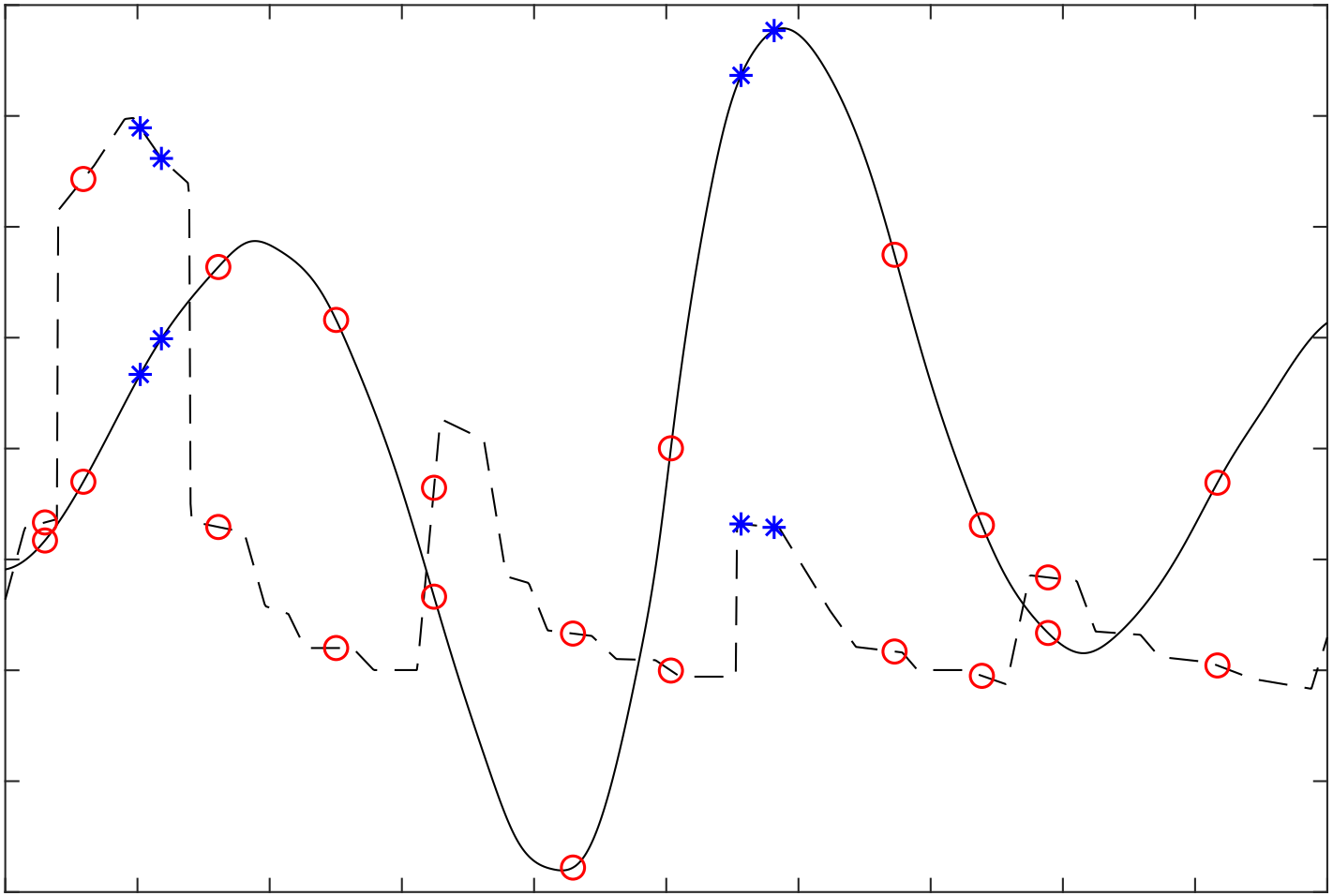
To determine the Pareto optimal solutions in this example when inspecting the $(x, \mathbf{f}_1)$ and $(x, \mathbf{f}_2)$ graphs, think about domination relations between the functions.

This approach also works for problems where there are more than three objectives. Unfortunately, higher-dimensional problems are already hard to visualize in the 2D or 3D objective space. Here, it is still possible to observe the 2D objective space thus ensuring that Pareto optimal solutions were identified correctly.

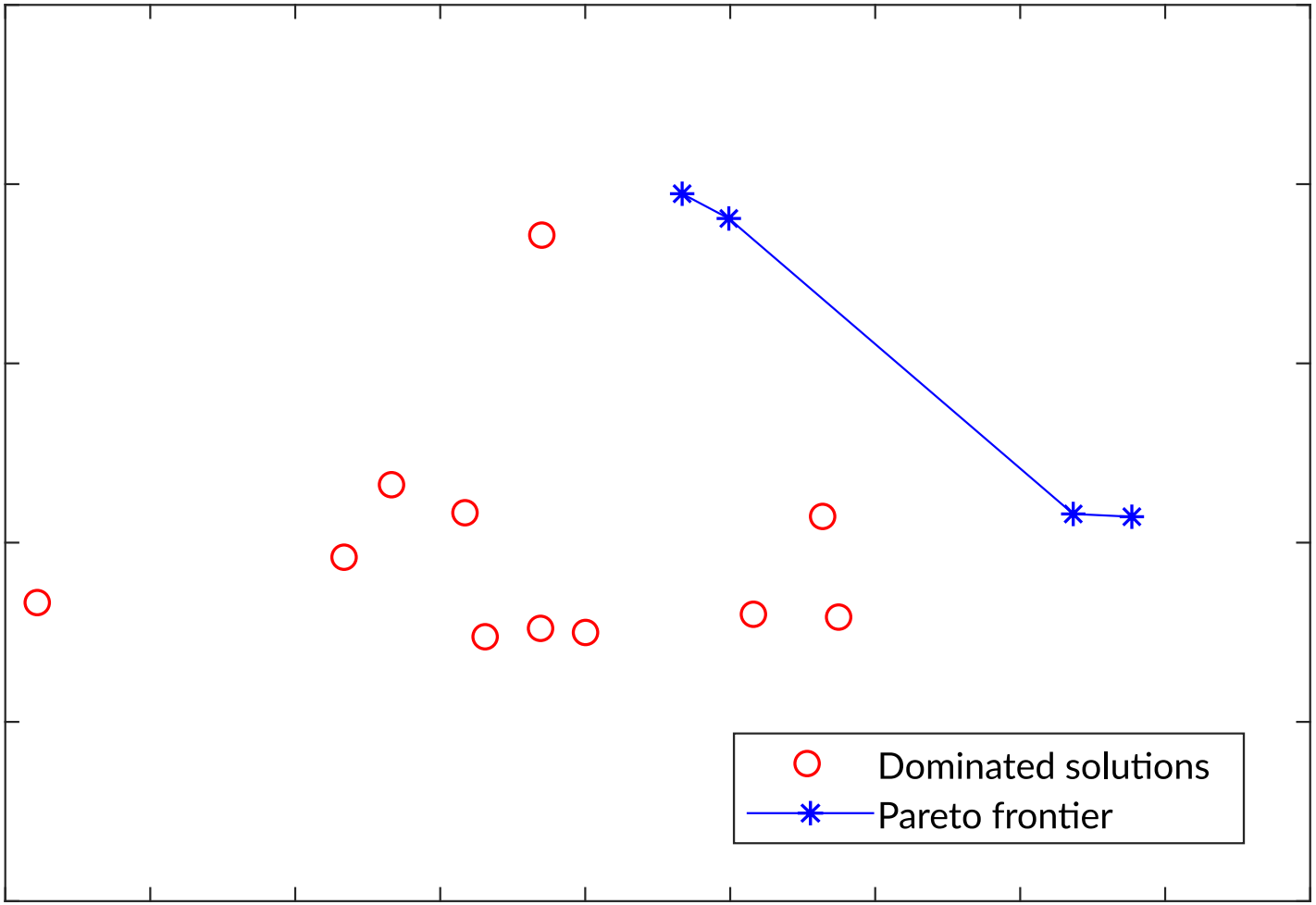
Pareto Optimality: Example (continued)



Pareto Optimality: Example (continued)



Pareto Optimality: Example (continued)



Structure of Optimization

Defenitions [2]

The **problem space** $\mathbb{X}$ (phenome) of an optimization problem is the set containing all elements x which could be its solution.

A **solution candidate** (phenotype) x is an element of the problem space $\mathbb{X}$ of a certain optimization problem.

The elements $g \in \mathbb{G}$ of the search space $\mathbb{G}$ of a given optimization problem are called the **genotypes**.

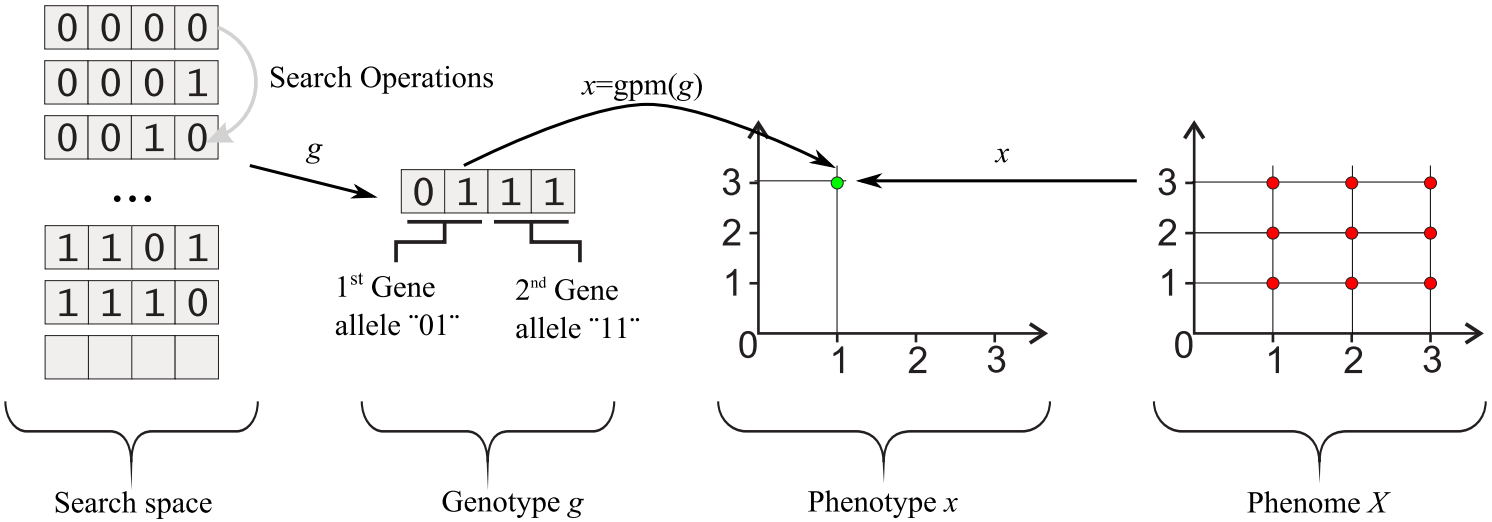
The distinguishable units of information in a genotype that encode the phenotypical properties are called **genes**.

An **allele** is a value of specific gene.

The Genotype-Phenotype Mapping (GPM) gpm : $\mathbb{G} \mapsto \mathbb{X}$ is a left-total binary relation which maps the elements of the search space $\mathbb{G}$ to elements in the problem space $\mathbb{X}$.

$$\forall g \in \mathbb{G}\ \exists x \in \mathbb{X} : \mathsf{gpm}(g) = x.$$

Genotype-Phenotype Mapping



An optimization problem is defined by a five- tuple $(\mathbb{X}, F, \mathbb{G}, \text{SearchOp}, \text{gpm})$ specifying the problem space $\mathbb{X}$, the objective functions F, the search space $\mathbb{G}$, the set of search operations *SearchOp*, and the genotype-phenotype mapping *gpm*.

Optimization Algorithm

The **fitness** value $fit(x) \in \mathbb{V}$ of an element x of the problem space $\mathbb{X}$ corresponds to its utility as solution or its priority in the subsequent steps of the optimization process. The space spanned by all possible fitness values $\mathbb{V}$ is normally a subset of the positive real numbers $\mathbb{V} \subseteq \mathbb{R}^+$.

An optimization algorithm is characterized by

1. The way it assigns fitness to the individuals.
2. Ways it selects them for further investigation.
3. The way it applies the search operations.
4. The way it builds and treats its state information.

Global Optimization Algorithm

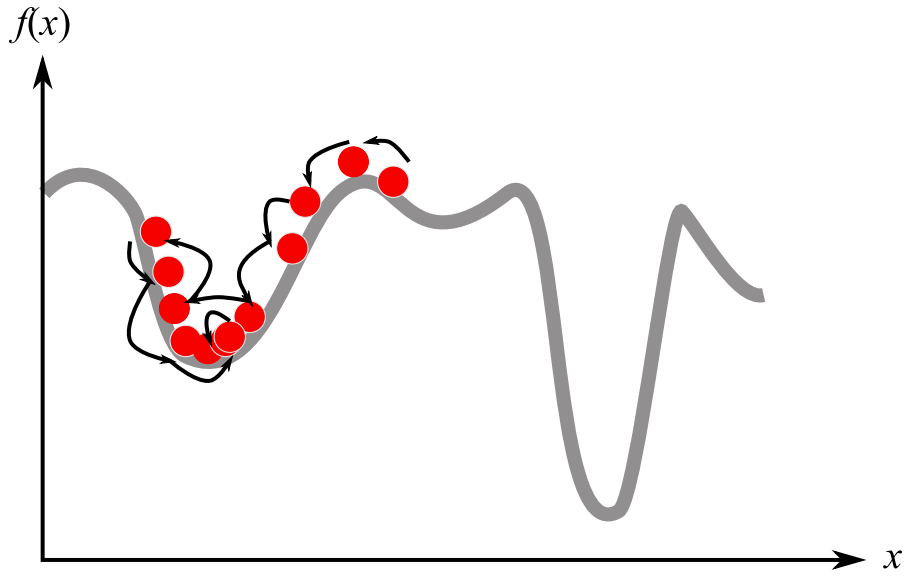
Global optimization algorithms are optimization algorithms that employ measures that prevent convergence to local optima and increase the probability of finding a global optimum.

1. Optimization algorithms discover good solutions with higher probability than solutions with bad characteristics.
2. The success of optimization highly depends on the way the search is conducted.
3. It also depends on the time (or the number of iterations) the optimizer allows to use.
4. Hill climbing algorithms are **not** global optimization algorithms since they have no means of preventing getting stuck at local optima.

Problems in Optimization

Premature Convergence

We say that an optimization process has *prematurely converged* to a local optimum if it is no longer able to explore other parts of the search space than the area currently being examined and there exists another region that contains a superior solution.



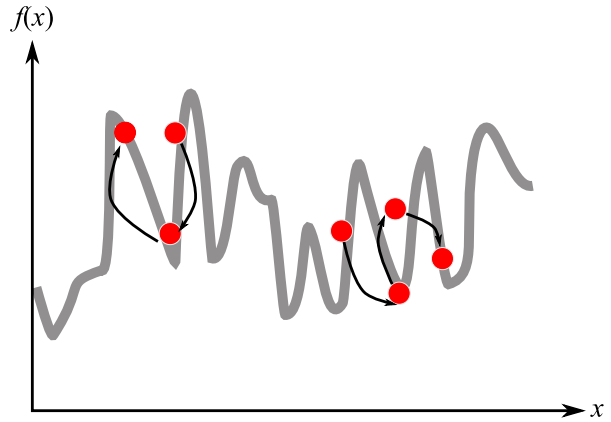
Premature Convergence (continued)

There is no general approach which can prevent premature convergence. Possible solutions:

1. Sometimes an effective measure is to restart the optimization process at randomly chosen points in time (clearing intermediate variables).
2. Increasing the proportion of exploration operations may also reduce the chance of premature convergence.
3. Introducing the capability of self-adaptation, allowing the optimization algorithm to change its strategies or to modify its parameters depending on its current state.

Ruggedness

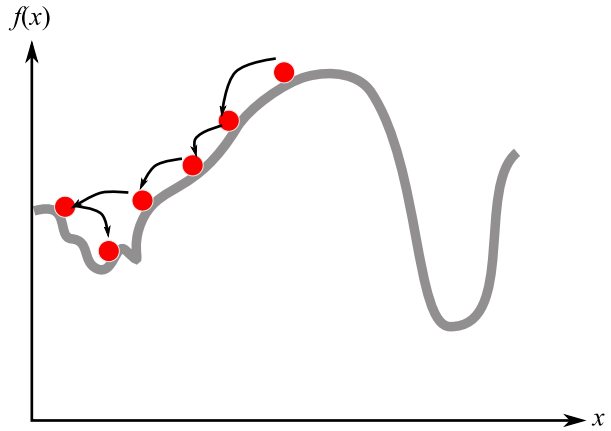
If the objective functions are unsteady or fluctuating, i.e., going up and down, it becomes more complicated for the optimization process to find the right directions to proceed to.



Countermeasures: no viable method which can directly mitigate the effects of rugged fitness landscapes exists. In population-based approaches, using large population sizes and applying methods to increase the diversity can reduce the influence of ruggedness, but only up to a certain degree.

Deceptiveness

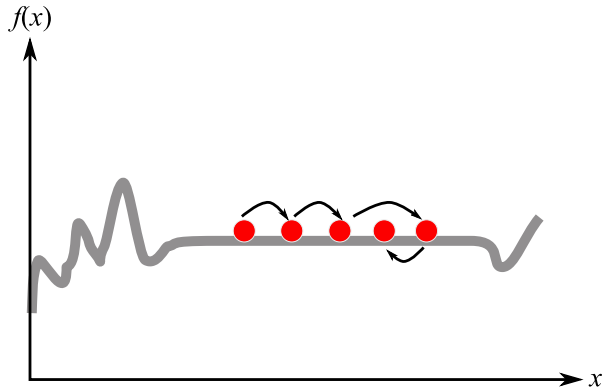
The gradient of deceptive objective functions leads the optimizer away from the optima.



Using large population sizes, maintaining a very high diversity, and utilizing linkage learning are the only approaches which can provide at least a small chance of finding good solutions.

Neutrality

We consider the outcome of the application of a search operation to an element of the search space as neutral if it yields no change in the objective values.



An optimizer then cannot find any gradient information and thus, no direction in which to proceed in a systematic manner.

The **evolvability** of an optimization process in its current state defines how likely the search operations will lead to solution candidates with new values of the objectives.

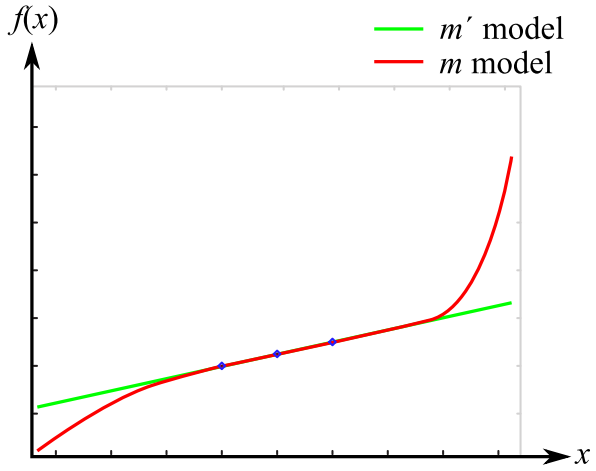
Redundancy

Redundancy in the context of global optimization is a feature of the genotype-phenotype mapping and means that multiple genotypes map to the same phenotype.

Redundancy can have a strong impact on the explorability of the problem space. When utilizing a one-to-one mapping, the translation of a slightly modified genotype will always result in a different phenotype. If there exists a many-to-one mapping between genotypes and phenotypes, the search operations can create offspring genotypes different from the parent which still translate to the same phenotype.

Overfitting

Overfitting is the emergence of an overly complicated model (solution candidate) in an optimization process resulting from the effort to provide the best results for as much of the available training data as possible and which works well **only** for the training data.



A model $m \in X$ optimized based on a finite set of training data is considered to be overfitted if a less complicated, alternative model $m' \in X$ exists which has a smaller error for producible data samples.

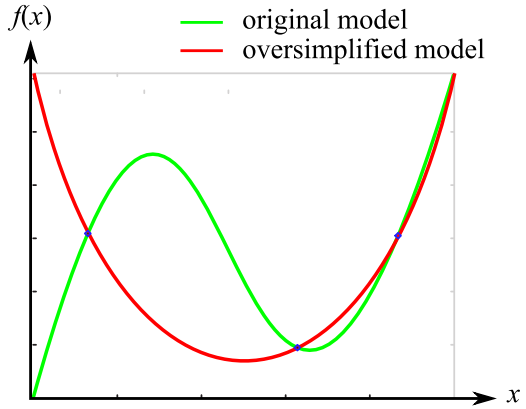
Overfitting (continued)

Possible solutions:

1. Use a new set of scenarios for each evaluation of a solution candidate. The resulting objective values may then differ largely even if the same individual is evaluated twice in a row, introducing incoherence and ruggedness into the fitness landscape.
2. At the beginning of each iteration of the optimizer, a new set of (randomized) scenarios is generated which is used for all individual evaluations during that iteration. This method leads to objective values which can be compared without bias. They can be made even more comparable if the objective functions are always normalized into some fixed interval, say [0, 1].
3. Limit the runtime of the optimizers.

Oversimplification

The training sets only represent a fraction of the set of possible inputs. Be aware of such an incomplete coverage which may fail to represent some of the dependencies and characteristics of the data, which then may lead to oversimplified solutions.



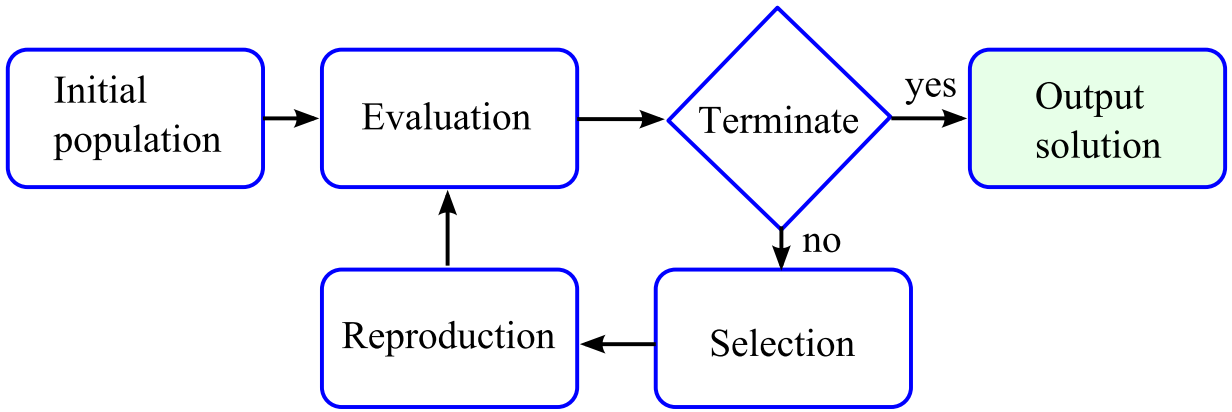
Another reason for oversimplification is that ruggedness, deceptiveness, too much neutrality, may lead to premature convergence. Yet another possible cause is that a particular problem space could have been chosen such that does not include the correct solution.

Evolutionary Algorithm

EA

Evolutionary algorithms (EAs) are population-based metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively.

Basic steps of EA:



Family of EA

1. Genetic Algorithms (GA) subsume all evolutionary algorithms which have bit strings as search space $\mathbb{G}$.
2. The set of evolutionary algorithms which explore the space of real vectors $\mathbb{X} \subseteq \mathbb{R}^n$ is called Evolution Strategies (ES).
3. On one hand, Genetic Programming (GP) includes all evolutionary algorithms that grow programs, algorithms, and these alike. On the other hand, also all EAs that evolve tree-shaped individuals are instances of GP.
4. Learning Classifier Systems (LCS) are online learning approaches that assign output values to given input values. They internally use a GA to find new rules for this mapping.
5. Evolutionary programming (EP) is an evolutionary approach that treats the instances of the genome as different species rather than as individuals. Over the decades, it has more or less merged into Genetic Programming and the other evolutionary algorithms.

Populations

There exist various way in which an evolutionary algorithm can process its population.

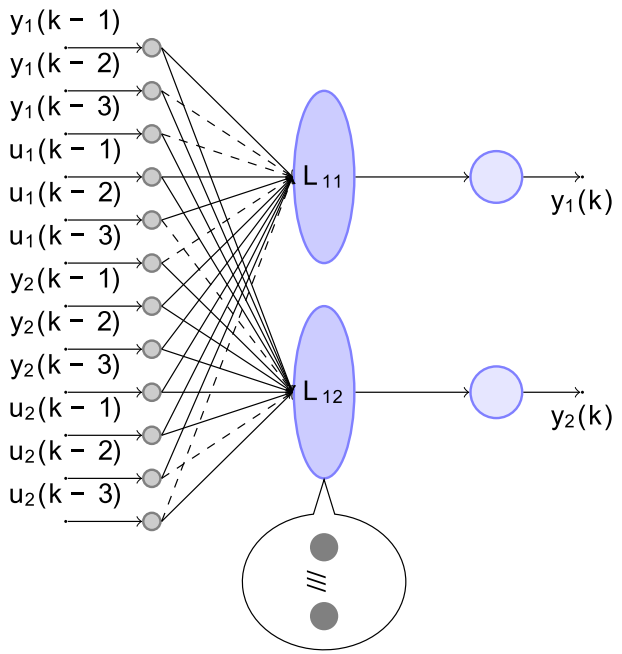
In evolutionary algorithms that are **generational** the next generation will only contain the offspring of the current one and no parent individuals will be preserved.

An **elitist** evolutionary algorithm ensures that at least one copy of the best individual(s) of the current generation is propagated on to the next generation.

The main advantage of elitism is that its convergence is guaranteed, meaning that once the global optimum has been discovered, the evolutionary algorithm converges to that optimum. On the other hand, the risk of converging to a local optimum is also higher.

Genomes

A string chromosome can either be a fixed-length tuple or a variable-length list. String chromosomes are normally bit strings, vectors of integer numbers, or vectors of real numbers. Example (NARX neural network representation):



gene = [1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1].

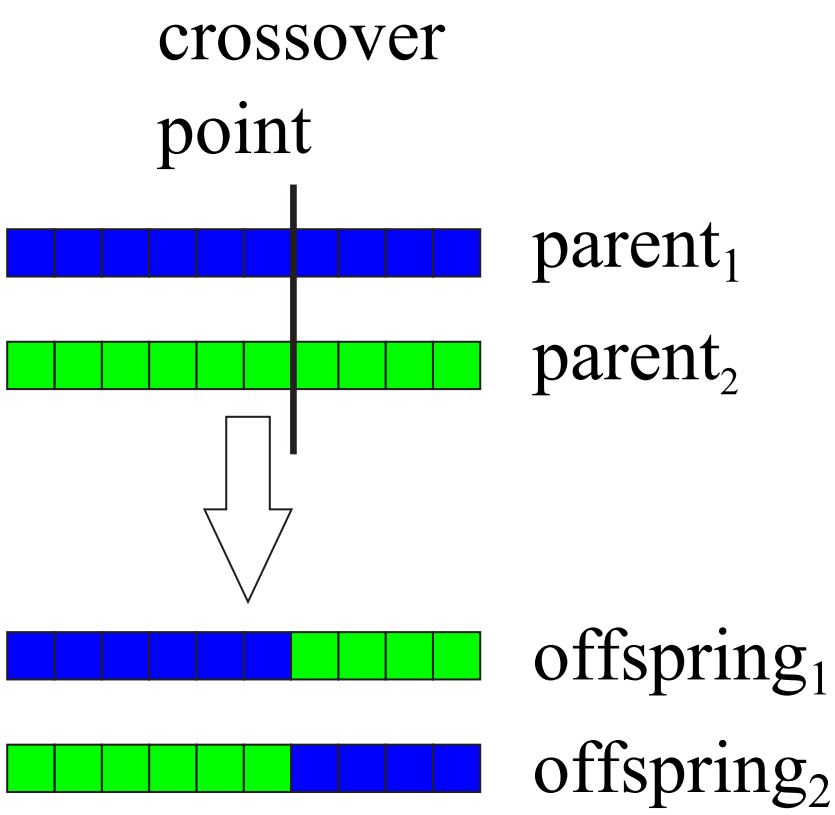
GA Operations

Mutation is an important method for preserving the diversity of the solution candidates by introducing small, random changes into them.

- In binary coded chromosomes, for example, these genes would be bits which can simply be toggled.
- For real-encoded genomes, modifying an element g_i can be done by replacing it with a number drawn from a normal distribution with expected value g_1.

Crossover is a recombination of two string chromosomes, which is performed by swapping parts of two genotypes.

Crossover



Particle Swam Optimization

When a swarm looks for food, its individuals will spread in the environment and move around independently. Each individual has a degree of freedom or randomness in its movements which enables it to find food accumulations. So, sooner or later, one of them will find something digestible and, being social, announces this to its neighbors. These can then approach the source of food, too.

With Particle Swarm Optimization (PSO), a swarm of particles (individuals) in a n-dimensional search space $\mathbb{G}$ is simulated, where each particle p has a position $p.g \in \mathbb{G} \subseteq \mathbb{R}^n$ and a velocity $p.v \in \mathbb{R}^n$. The position $p.g$ corresponds to the genotypes, and, in most cases, also to the solution candidates, i. e., $p.x = p.g$, since most often the problem space $\mathbb{X}$ is also the $\mathbb{R}^n$ and $\mathbb{X} = \mathbb{G}$. The velocity vector $p.v$ of an individual p determines in which direction the search will continue and if it has an explorative (high velocity) or an exploitive (low velocity) character.

Differential Evolution

Taking the *difference* between the two differentiation vectors is very much like taking the derivative. But as the two differentiation vectors are usually quite far apart (certainly not infinitesimally far), this "derivative" is more a global measure of how much the objective function changes *on average* over that interval.

The derivative is computed at each iteration between two new, *randomly* selected vectors, so on average, the solutions will tend to go to where the average slope is zero, and the function globally minimal. Sometimes this operation is called the **global pseudo-derivative**, and it is the key to the power of the DE algorithm.

Numerous variations on this basic algorithm exist. However, they normally improve DE's performance only marginally, and as such, they will not be mentioned here.

ASA

Traditionally, for Adaptive Simulated Annealing, individual trial solutions are called "atoms" or "particles", to reflect the method's underlying philosophy - as the temperature drops, the atoms literally "freeze" into low-energy states (low function values). But before they freeze, they have the ability to move to *higher* energy states, with a certain probability. This is what makes ASA also a global optimizer, in the sense that it is not "greedy" as to only accept lower function values, but also explores regions behind high-energy barriers.

Originally, Simulated Annealing was built around a single solution (the initial condition). However, this method is easily rewritten into a population-based method (just use N randomly generated initial conditions).

Symbolic Regression

Motivation for Symbolic Regression

- Machine learning methods such as Artificial Neural Networks produce models that cannot be analyzed symbolically. Meanwhile, Symbolic Regression produces symbolic models (mathematical expressions).
- Symbolic Regression allows to evolve models without any initial assumptions about model structure.
- A model portfolio is generated that contains models of different levels of complexity and predictive quality.
- Hidden, nonlinear relationships in the data may be discovered.

Enter GPTIPS 2: an open-source software platform for symbolic data mining implemented in MATLAB.

Symbolic Regression: Key Concepts

Symbolic regression essentially evolves computer programs based on given criteria. The programs are usually represented as **tree data structures**. Every tree is referred to as a **gene** that, in turn, is comprised of

- **functions** (symbols) and
- **terminals** (end nodes)

that for the purpose of computer processing are encoded as a **string of characters**.

The underlying principle is that of genetic programming—simulated process of evolution, where best individuals in the population (candidate symbolic expressions) are evaluated based upon their fitness (natural selection). New individuals are obtained via

- **Growth**—branches are added to trees until a terminal node is selected;
- **Mutation**—a terminal node is replaced with a random node;
- **Crossover**—a random node is replaced with a subtree from an individual.

Basic Building Blocks

Basic building blocks of a **multi-gene** symbolic regression problem are

- Function set $\mathcal{F}_{fun}$ that contains available function symbols, e.g.,

$$\mathcal{F}_{fun} = \{+, -, \times, \sqrt{\ }, \sin\}.$$

- Terminal set $\mathcal{T}_{term}$ that contains available terminals, e.g.,

$$\mathcal{T}_{term} = \{x_1, x_2, \varepsilon_C\},$$

  where ε_C is a set of so called ephemeral (random) constants.

The overall expression can be represented as a linear combination of genes as

$$y = a_0 + a_1 G_1 + a_2 G_2 + \cdots + a_n G_n,$$

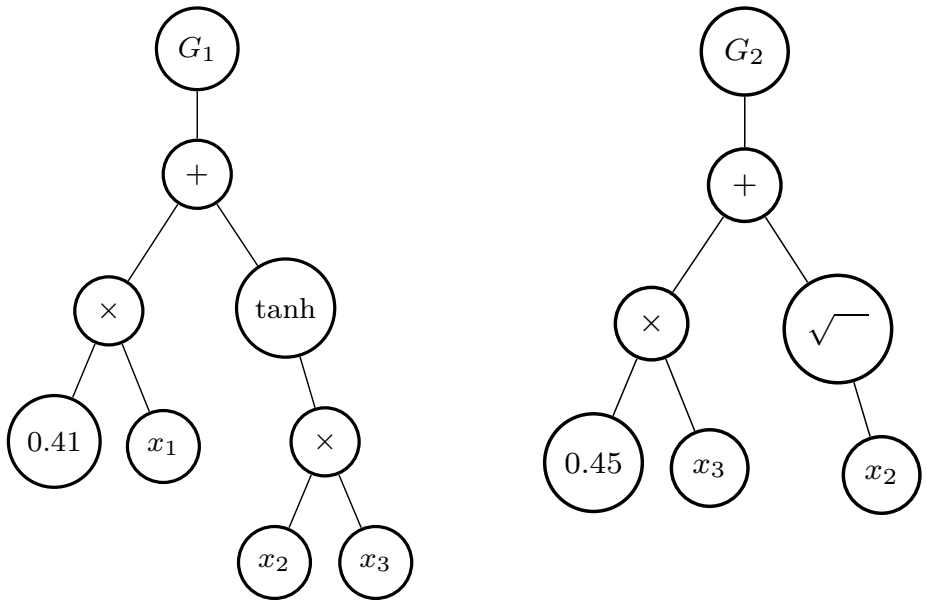
where the vector $a = \begin{bmatrix} a_0 & a_1 & \cdots & a_n \end{bmatrix}$ can be obtained on each iteration by means of least squares estimation.

Example: Multiple Gene Model

The model

$$y = d_0 + d_1 \underbrace{(0.41x_1 + \tanh(x_2x_3))}_{G_1} + d_2 \underbrace{(0.45x_3 + \sqrt{x_2})}_{G_2}$$

is represented (without the weights) by two genes as

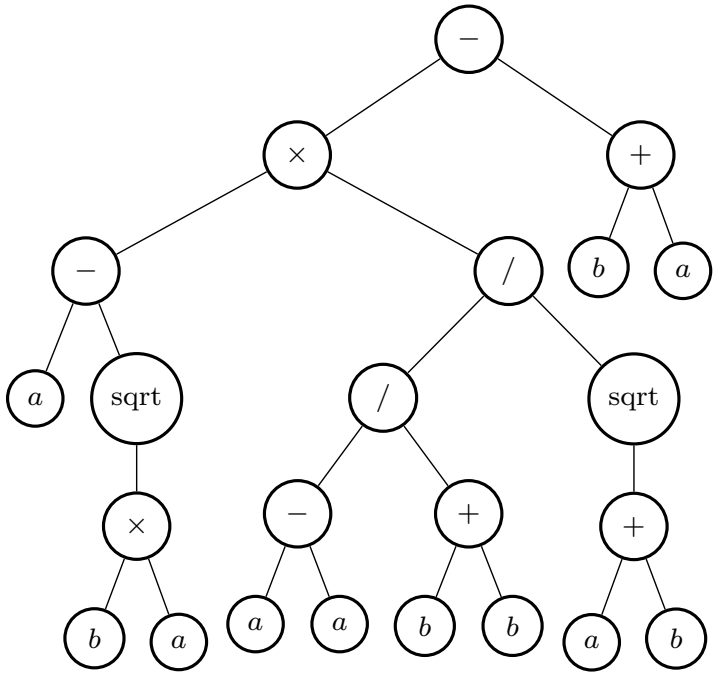


Example: Encoded String

Encoded expression:

`(-(*(-a(sqrt(*ba)))(/(/(-aa)(+bb))(sqrt(+ab))))(+ba))`

Corresponding tree:



Bibliography

[1] D. Kalyanmoy, "Evolutionary algorithms for multi-criterion optimization in engineering design," *Evolutionary Algorithms in Engineering and Computer Science*, vol. 8, pp. 135–161, 3 1999.

[2] T. Weise, *Global Optimization Algorithms–Theory and Application*, 2nd ed. Germany: it-weise.de (self-published), 2009. [Online]. Available: http://www.it-weise.de/projects/book.pdf

[3] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.

[4] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992.