# Nature Inspired Search and Optimisation Advanced Aspects of Nature Inspired Search and Optimisation

## Lecture 8: Constraint Handling in Evolutionary Algorithms (I)

Shan He

School for Computational Science
University of Birmingham

February 6, 2020

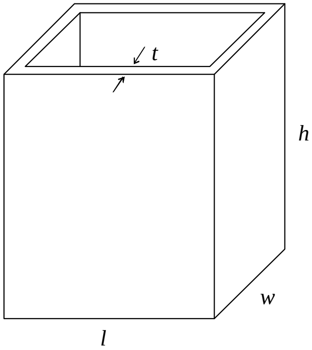
# Outline of Topics

# Example 1: Cubic Vessel Design

- **Aims**: to find an optimal values of length $l$, width $w$, high $h$, and thickness $t$, to minimize the material consumption (or equivalently, the surface area).

Engineering Opt.

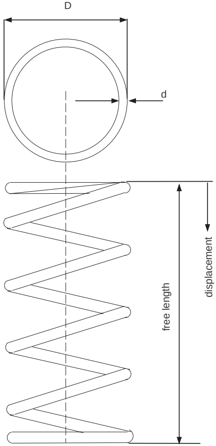
- **Constraints**: material consumption cannot be minimise infinitely since there are some requirements:
    - Quality requirements, e.g., the maximum deflection should be less than an allowable value.
    - Restrictions imposed by government and corporate regulations, e.g., shapes or the maximum capacity



# Example 2: Spring design

- **Aims** to minimize the weight of a tension/compression spring. All design variables are continuous (See my paper).
- **Four constraints**:
  - Minimum deflection
  - Shear stress
  - Surge frequency
  - Diameters
- Let the wire diameter $d = x_1$ , the mean coil diameter $D = x_2$ and the number of active coils $N = x_3$
- There are boundaries of design variables:

$$0.05 \leq x_1 \leq 2,\ 0.25 \leq x_2 \leq 1.3,\ 2 \leq x_3 \leq 15$$



# Constrained Optimisation Example: Spring design

- Minimize

$$f(X) = (x_3 + 2)x_2x_1^2 \tag{1}$$

subject to:

$$g_1(X) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \tag{2}$$

$$g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{3}$$

$$g_3(X) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0 \tag{4}$$

$$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \tag{5}$$

# Engineering Optimisation

- **Engineering optimisation (Design Optimisation):** to find the best combination of design variables that optimises designer's preference (design objective) and satisfies certain requirements (constraints).
- **Design variables**: A design variable is under the control of designer and could have an impact on the solution of the optimization problem
- Types of design variables can be:
  - Continuous
  - Integer (including binary)
  - Set of variables: designers are required to choose the design variables from a list of recommended values from design standards
- **Design objective**: represents the desires of the designers, e.g., to maximize profit or minimize cost.

# Engineering Optimisation

- **Constraints:** Designers desires cannot be optimized infinitely because of
  - Limited resources: budget or materials that can be used in product development.
  - Other restrictions such as user requirements and regulations
  - A design constraint is "rigid" or "hard" since usually it needs to be satisfied strictly
- Engineering optimisation, as well many real-world optimisation problems are **constrained optimisation problems**

# What Is Constrained Optimisation?

- The general constrained optimisation problem:

$$\min_{\mathbf{x}}\{f(\mathbf{x})\}$$

subject to

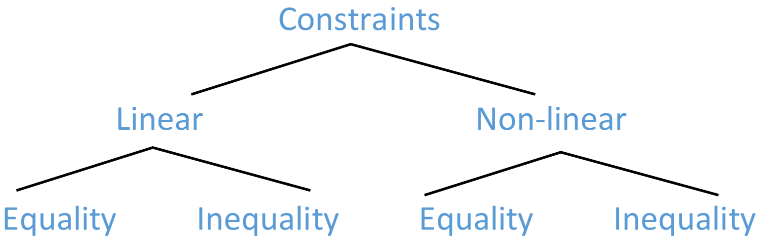
$$g_i(\mathbf{x}) \leq 0,\ i = 1, \cdots, m$$

$$h_j(\mathbf{x}) = 0,\ j = 1, \cdots, p$$

where $\mathbf{x}$ is the $n$ dimensional vector, $x = (x_1, x_2, \cdots, x_n)$; $f(\mathbf{x})$ is the objective function; $g_i(\mathbf{x})$ is the inequality constraint; and $h_j(\mathbf{x})$ is the equality constraint.

- Denote the whole search space as $\mathcal{S}$ and the feasible space as $\mathcal{F}$, $\mathcal{F} \in \mathcal{S}$.
- Note: the global optimum in $\mathcal{F}$ might not be the same as that in $\mathcal{S}$.

# Different Types of Constraints




- Linear constraints are relatively easy to deal with
- Non-linear constraints can be hard to deal with

# Constraint Handling Techniques in Evolutionary Algorithms

- **The purist approach**: rejects all infeasible solutions in search
- **The separatist approach**: considers the objective function and constraints separately.
- The **penalty function approach**: converts a constrained problem into an unconstrained one by introducing a penalty function into the objective function.
- **The repair approach**: maps (repairs) an infeasible solution into a feasible one.
- **The hybrid approach** mixes two or more different constraint handling techniques.

# Penalty Function Approach: Introduction

- New Objective Function = Original Objective Function + Penalty Coefficient * Degree Of Constraint Violation
- The general form of the penalty function approach:

$$f'(\mathbf{x}) = f(\mathbf{x}) + \left( \sum_{i=1}^{m} r_i G_i(\mathbf{x}) + \sum_{j=1}^{p} c_j H_j(\mathbf{x}) \right)$$

where $f'(\mathbf{x})$ is the new objective function to be minimised, $f(\mathbf{x})$ is the original objective function, $r_i$ and $c_j$ are penalty factors (coefficients), and $G_i$ and $H_j$ are penalty functions for inequality and equality constraints, respectively:

$$\begin{cases} 0 & \leq 0 \\ g_i(x)^\beta & > 0 \end{cases}$$

$$G_i(\mathbf{x}) = \max(0, g_i(\mathbf{x}))^\beta, \; H_j(\mathbf{x}) = \max(0, |h_j(\mathbf{x})|)^\gamma, \quad \begin{cases} 0 & = 0 \\ \pm h_j(x) & \neq 0 \end{cases}$$

where $\beta$ and $\gamma$ are usually chosen as 2.

# Penalty Function Approach: Techniques

- **Static Penalties**: The penalty function is pre-defined and fixed during evolution.
- **Dynamic Penalties**: The penalty function changes according to a pre-defined sequence, which often depends on the generation number.
- **Adaptive and Self-Adaptive Penalties**: The penalty function changes adaptively.

# Static Penalty Functions

- Static penalty functions general form:

$$f'(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} r_i (G_i(\mathbf{x}))^2$$

  where $r_i$ are predefined and fixed.
- Equality constraints can be converted into inequality ones:

$$h_j(\mathbf{x}) \Longrightarrow h_j(\mathbf{x}) - \epsilon \leq 0$$

  where $\epsilon > 0$ is a small number.
- Simple and easy to implement.
- Requires rich domain knowledge to set $r_i$.
- $r_i$ $(i = 1, \cdots, m + p)$ can be divided into a number of different levels. When to use which is determined by a set of heuristic rules, e.g., the more important a constraint $g_i$, the larger the value of $r_i$.

# Dynamic Penalty Functions

- Dynamic Penalties general form:

$$f'(\mathbf{x}) = f(\mathbf{x}) + r(t)\sum_{i=1}^{m} G_i^2(\mathbf{x}) + c(t)\sum_{j=1}^{p} H_j^2(\mathbf{x})$$

where $r(t)$ and $c(t)$ are two penalty coefficients.

- General principle: the large the generation number $t$, the larger the penalty coefficients $r(t)$ and $c(t)$.
- **Question:** why the large the generation number, the larger the penalty coefficients?

Less infeasible solutions in the final generations

# Dynamic Penalty Functions

- Common dynamic penalty coefficients:
  - **Polynomials**: $r(t) = \sum_{k=1}^{N} a_{k-1} t^{k-1}$, $c(t) = \sum_{k=1}^{N} b_{k-1} t^{k-1}$ where $a_k$ and $b_k$ are user-defined parameters.
  - **Exponentials**: $r(t) = e^{at}$, $c(t) = e^{bt}$ where $a$ and $b$ are user-defined parameters.
  - **Hybrid**: $r(t) = e^{\sum_{k=1}^{N} b_{k-1} t^{k-1}}$, $c(t) = e^{\sum_{k=1}^{N} b_{k-1} t^{k-1}}$

# Penalty function, Fitness Function and Selection

- Let static penalty function $\Phi(\mathbf{x}) = f(\mathbf{x}) + rG(\mathbf{x})$, where $G(\mathbf{x}) = \sum_{i=1}^{m} G_i(\mathbf{x})$ and $G_i(\mathbf{x}) = \max\{0, g_i(\mathbf{x})\}$
- **Question**: How does $r$ affect an Evolutionary Algorithm?

selection
fitness calc
reproduction

## Generic Evolutionary Algorithm

$\mathbf{X}_0$ := generate initial population of solutions
terminationflag := false
t := 0
Evaluate the fitness of each individual in $\mathbf{X}_0$.
while (terminationflag != true)
  **Selection**: Select parents from $\mathbf{X}_t$ based on their fitness.
  **Variation**: Breed new individuals by applying variation operators to parents
  **Fitness calculation**: Evaluate the fitness of new individuals.
  **Reproduction**: Generate population $\mathbf{X}_{t+1}$ by replacing least-fit individuals
  t := t + 1
  If a termination criterion is met: terminationflag := true
Output $x_{best}$

# Penalty function, Fitness Function and Selection

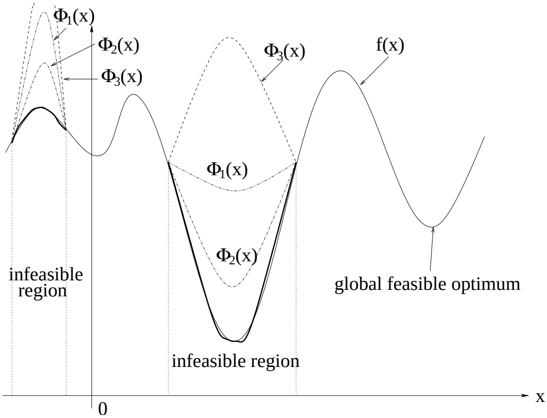
- **Minimisation** problem with a penalty function $\Phi(\mathbf{x}) = f(\mathbf{x}) + rG(\mathbf{x})$
- Given two individual $\mathbf{x_1}$ and $\mathbf{x_2}$, their fitness values are now determined by $\Phi(\mathbf{x}) \Longrightarrow$ changing fitness values
- **Fitness proportional selection**: Because fitness values are used primarily in selection: Changing fitness $\Longrightarrow$ changing selection probabilities
- **Ranking selection**: $\Phi(\mathbf{x_1}) < \Phi(\mathbf{x_2})$ means $f(\mathbf{x_1}) + rG(\mathbf{x_1}) < f(\mathbf{x_2}) + rG(\mathbf{x_2})$

  $[f(x_1) - f(x_2)] + r[G(x_1) - G(x_2)] < 0$

  - 1 $f(\mathbf{x_1}) \le f(\mathbf{x_2})$ and $G(\mathbf{x_1}) \le G(\mathbf{x_2})$: $r$ has no impact on the comparison
  - 2 $f(\mathbf{x_1}) < f(\mathbf{x_2})$ and $G(\mathbf{x_1}) > G(\mathbf{x_2})$: Increasing $r$ will eventually change the comparison
  - 3 $f(\mathbf{x_1}) > f(\mathbf{x_2})$ and $G(\mathbf{x_1}) < G(\mathbf{x_2})$: Decreasing $r$ will eventually change the comparison
- Ranking selection: Different $r$ lead to different ranking of individual in the population

# Penalties and Fitness Landscape Transformation

- Different penalty functions lead to different new objective functions.
- **Question**: For the following minimisation problem, which penalty function should we avoid? Why?



# Penalty Functions Demystified

- Penalty function essentially:
  - Transforms fitness
  - Changes rank $\rightarrow$ changes selection
- Why not change the rank directly in an EA?
- We will introduce Stochastic Ranking in our next lecture