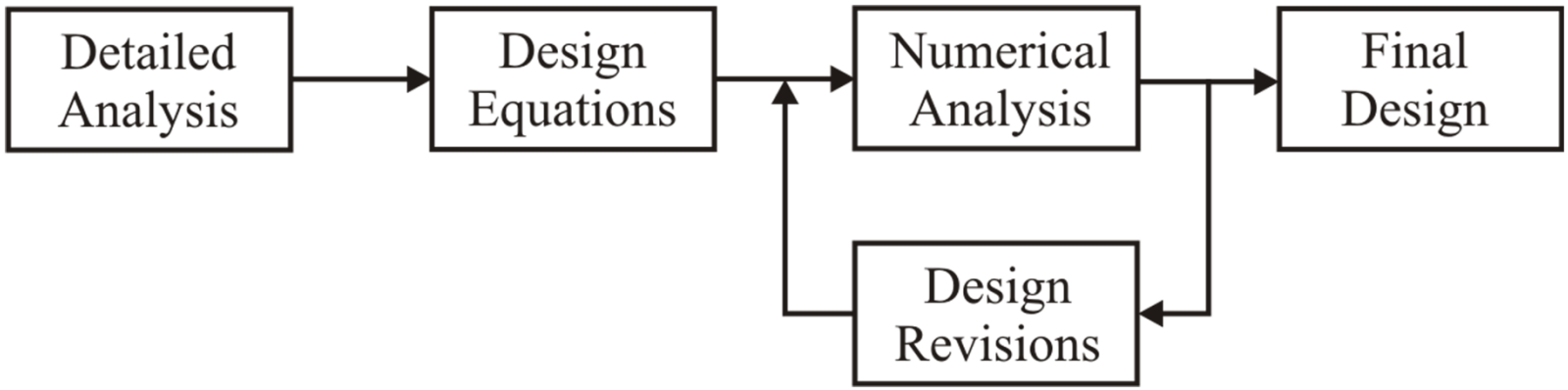
Power Magnetic Devices: A Multi-Objective Design Approach

Chapter 1: Optimization Based Design

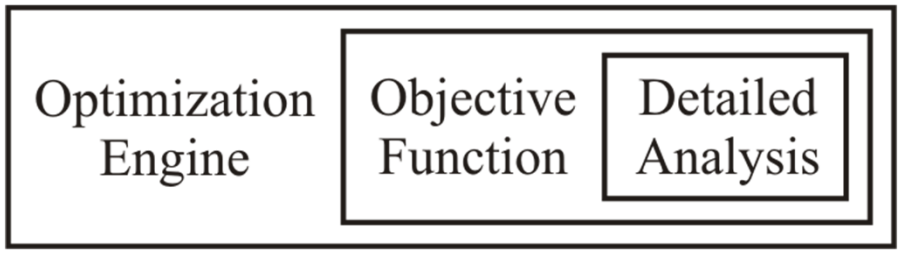
1.1 Design Approach

- Manual design process



1.1 Design Approach

- Optimization based design process



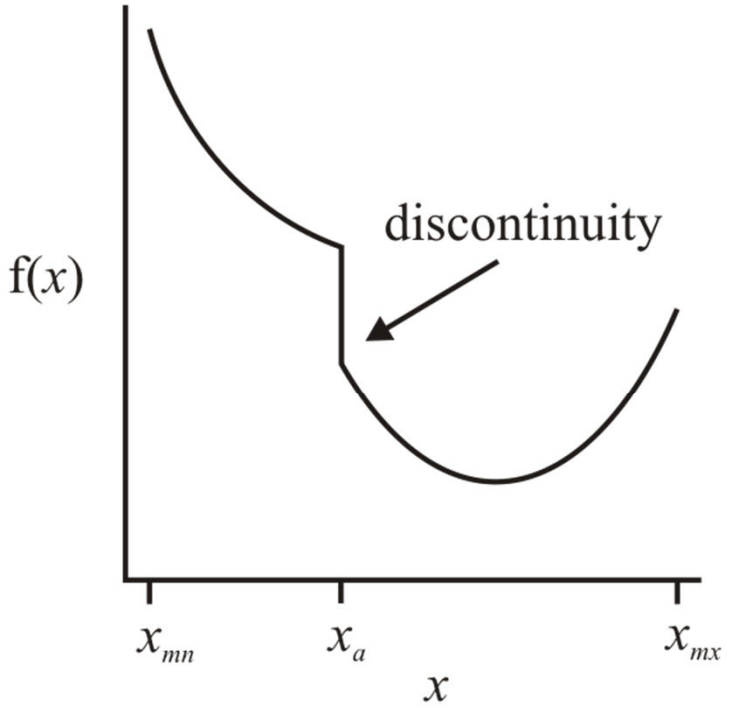
1.2 Mathematical Properties of Obj. Functions

- Definition of a global minimizer

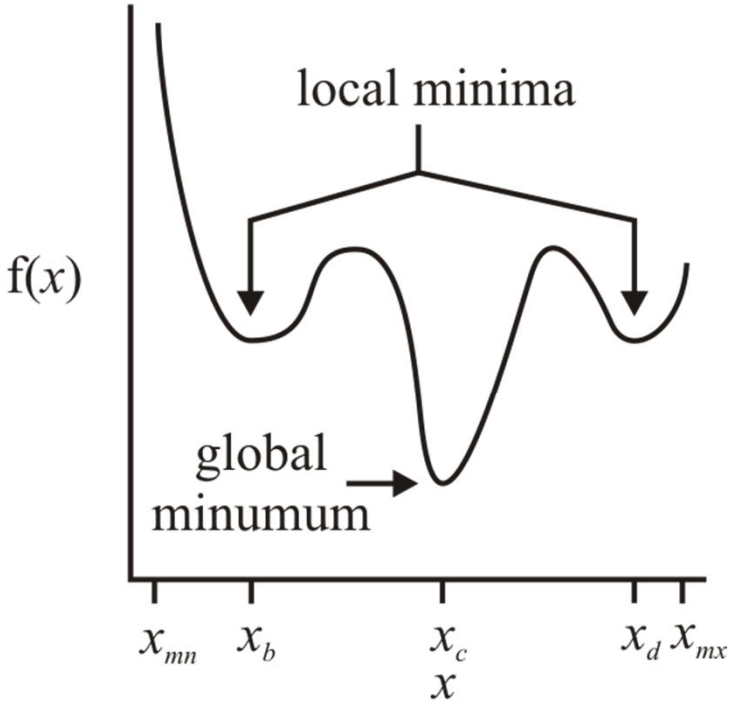
$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall x \in \Omega \setminus \{x^*\}$$

1.2 Mathematical Properties of Obj. Functions

- Discontinuities and local extrema



(a) discontinuity

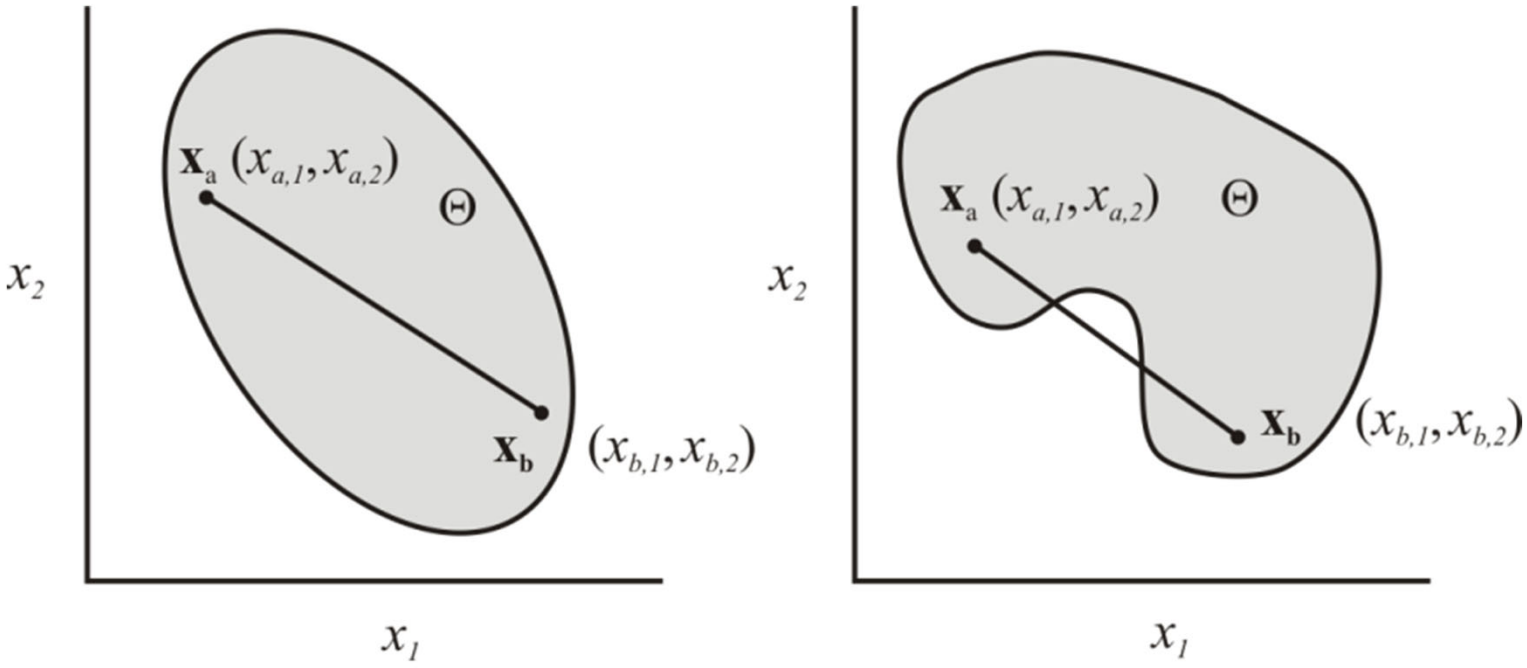


(b) local extremum

1.2 Mathematical Properties of Obj. Functions

- Definition of a convex set

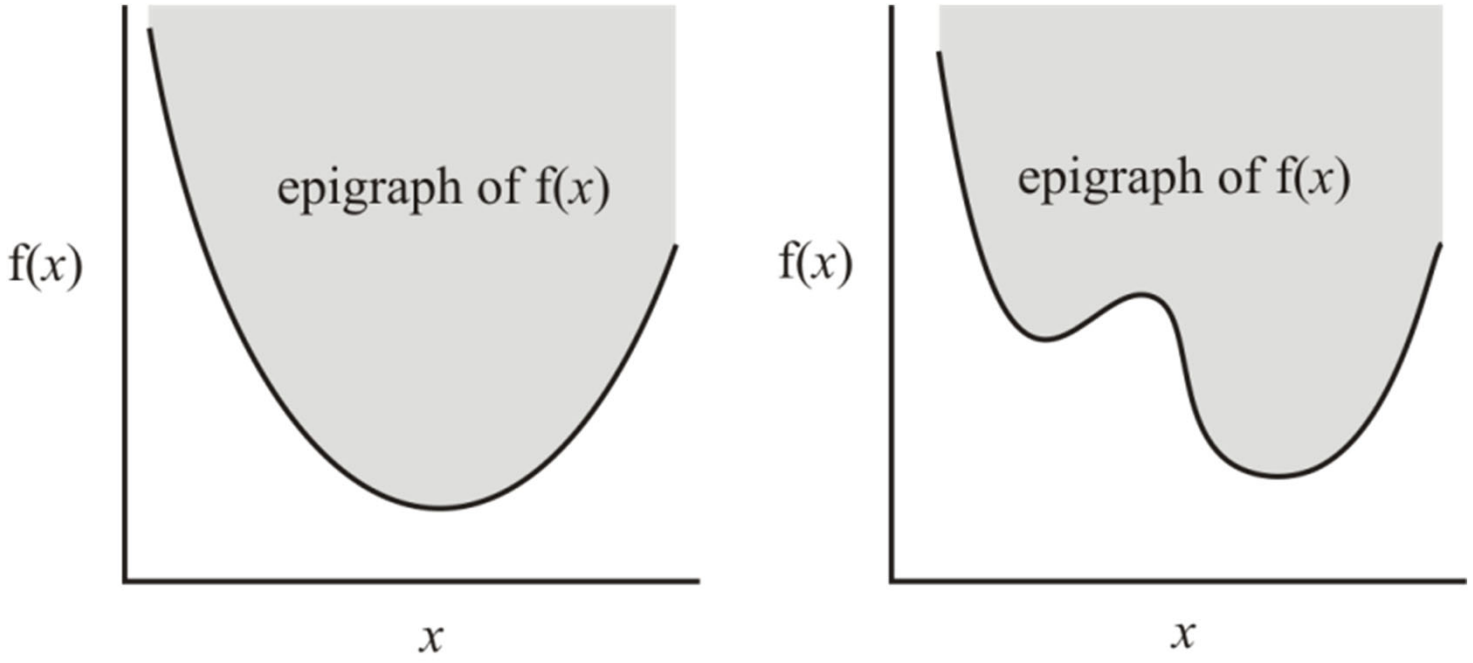
$$\alpha \mathbf{x}_a + (1-\alpha)\mathbf{x}_b \in \Theta \quad \forall \alpha \in [0,1]$$



(a) convex set (b) non-convex set

1.2 Mathematical Properties of Obj. Functions

- Definition of a convex function: the epigraph is a convex set



(a) convex function (b) non-convex function

1.2 Mathematical Properties of Obj. Functions

- Importance of convexity

1.3 Single Obj. Optim. Using Newton's Method

- Let us find the extrema of f(x)
- Assume that

$$f(\mathbf{x}) \in \mathbb{R} \qquad \mathbf{x} \in \mathbb{R}^n$$

- The first derivate or gradient is denoted

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right]^T$$

1.3 Single Obj. Optim. Using Newton's Method

- The second derivative or Hessian is denoted

$$\mathrm{F}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial^2 x_1} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_2 \partial x_1} & \mathrm{B} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_n \partial x_1} \\ \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_1 \partial x_2} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial^2 x_2} & \mathrm{B} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_n \partial x_2} \\ \mathrm{C} & \mathrm{C} & \mathrm{E} & \mathrm{C} \\ \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_1 \partial x_n} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial x_2 \partial x_n} & \mathrm{B} & \dfrac{\partial^2 \mathrm{f}(\mathbf{x})}{\partial^2 x_n} \end{bmatrix}$$

1.3 Single Obj. Optim. Using Newton's Method

- If x^* is a local minimizer of f, then

$$\nabla f(\mathbf{x}^*) = \mathbf{0}$$

$$F(\mathbf{x}^*) \geq 0$$

- These conditions are necessary but not sufficient
- If f is convex, these conditions are necessary and sufficient to know that x^* is a global minimizer of f

1.3 Single Obj. Optim. Using Newton's Method

- Newton's Method for find a function extremum
  - This is an iterative method in which an estimate is used to find a better estimate
  - Let **x**[k] denote the k'th estimate. Using Newton's Method

$$\mathbf{x}[k+1] = \mathbf{x}[k] - \mathrm{F}(\mathbf{x}[k])^{-1} \nabla \mathrm{f}(\mathbf{x}[k])$$

1.3 Single Obj. Optim. Using Newton's Method

- Derivation

1.3 Single Obj. Optim. Using Newton's Method

- Derivation (continued)

1.3 Single Obj. Optim. Using Newton's Method

- Example 1.3A: Suppose we wish to minimize

$$f(x) = 2(x_1 - 2)^4 + 3(e^{x_2} - x_1)^2 + 8$$

- First we find the gradient:

1.3 Single Obj. Optim. Using Newton's Method

- Next we find the Hessian

1.3 Single Obj. Optim. Using Newton's Method

- Iteratively applying our update expression

Table 1.3A-1 Newton's method results.

k	$\mathbf{x}[k]$	$f(\mathbf{x}[k])$	$\nabla f(\mathbf{x}[k])$	$F(\mathbf{x}[k])$
1	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	43	$\begin{bmatrix} -70 \\ 6 \end{bmatrix}$	$\begin{bmatrix} 102 & -6 \\ -6 & 12 \end{bmatrix}$
2	$\begin{bmatrix} 0.677 \\ -0.162 \end{bmatrix}$	14.2	$\begin{bmatrix} -19.6 \\ 0.888 \end{bmatrix}$	$\begin{bmatrix} 48.0 & -5.10 \\ -5.10 & 5.23 \end{bmatrix}$
3	$\begin{bmatrix} 1.11 \\ 0.0938 \end{bmatrix}$	9.25	$\begin{bmatrix} -5.53 \\ -0.0938 \end{bmatrix}$	$\begin{bmatrix} 24.9 & -6.58 \\ -6.58 & 7.13 \end{bmatrix}$
10	$\begin{bmatrix} 1.95 \\ 0.666 \end{bmatrix}$	8.00	$\begin{bmatrix} -2.23 \cdot 10^{-3} \\ 2.00 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} 6.07 & -11.7 \\ -11.7 & 22.7 \end{bmatrix}$

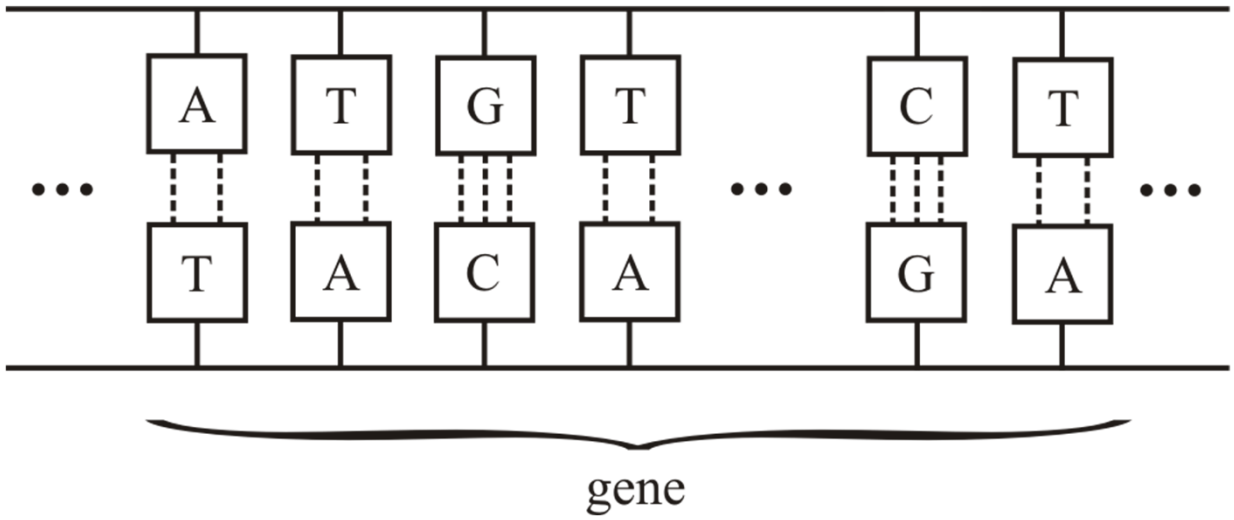
1.3 Single Obj. Optim. Using Newton's Method

- Problems with Newton's Method

- Population based optimization methods

1.4 GAs: Review of Biological Genetics

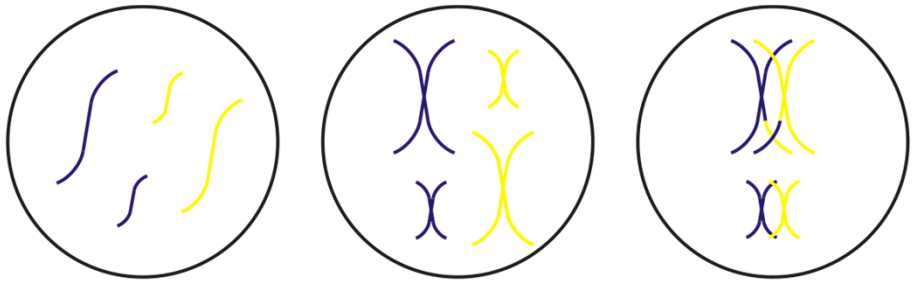
- Deoxyribonucleic acid



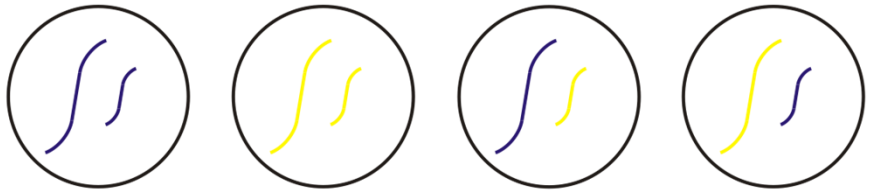
- Humans have 22 pairs of chromosomes plus two sex chromosomes

1.4 GAs: Review of Biological Genetics

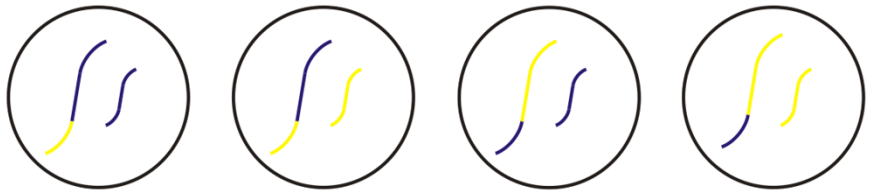
- Meiosis (formation of gametes)



(a) lengthening (b) replication (c) pairing



(d) possible chromosome distributions in gametes without crossover



(e) additional possible chromosome distributions in gametes with crossover

1.4 GAs: Review of Biological Genetics

- Example: with sexual reproduction in a creature with two chromosomes and no-crossover, how many genotypes could the children of the same two parents take on?

1.5 The Canonical Genetic Algorithm

- Salient features

1.5 The Canonical Genetic Algorithm

- Definitions
  - Let **P**[k] denote the k'th generation of a population of individuals
  - **P**[k] is organized as

$$\mathbf{P}[k] = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \text{B} \ \boldsymbol{\theta}^{N_p}\}$$

1.5 The Canonical Genetic Algorithm

- Definitions (continued)
- An individual is organized as

$$\boldsymbol{\theta}^i = \begin{bmatrix} \text{chromosome } 1 & \begin{Bmatrix} \boldsymbol{\theta}_1^i \\ \boldsymbol{\theta}_2^i \end{Bmatrix} \\ \text{chromosome } 2 & \begin{Bmatrix} \boldsymbol{\theta}_3^i \\ \boldsymbol{\theta}_4^i \\ \boldsymbol{\theta}_5^i \end{Bmatrix} \\ & \vdots \\ \text{chromosome } N_c & \begin{Bmatrix} \boldsymbol{\theta}_{N_g}^i \end{Bmatrix} \end{bmatrix}$$

1.5 The Canonical Genetic Algorithm

- A decoding function converts a genetic code to parameter values

$$\mathbf{x}^i = \mathrm{d}(\boldsymbol{\theta}^i)$$

where

$$\mathbf{x}^i = \begin{bmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_{N_g}^i \end{bmatrix}$$

1.5 The Canonical Genetic Algorithm

- The objective (fitness) function describes the goodness of the individual

$$f^i = \mathrm{f}(\mathbf{x}^i)$$

1.5 The Canonical Genetic Algorithm

- Example 1.5A. Suppose individual 13 has the genetic code

$$\boldsymbol{\theta}^{13} = \left[\underbrace{0\ 1\ 0}_{\text{gene 1}}\ \underbrace{1\ 1\ 0}_{\text{gene 2}}\ \underbrace{1\ 0\ 1}_{\text{gene 3}}\right]^T$$

The decoding function is given by

$$x_i = x_{mn,i} + \frac{x_{mx,i} - x_{mn,i}}{2^l - 1} \sum_{m=1}^{l} b_m 2^{m-1}$$

where x_{mn1}=5, x_{mx1}=10, x_{mn2}=-2, x_{mx2}=0, x_{mn3}=0, and x_{mx3}=1

1.5 The Canonical Genetic Algorithm

Finally the fitness function is given by

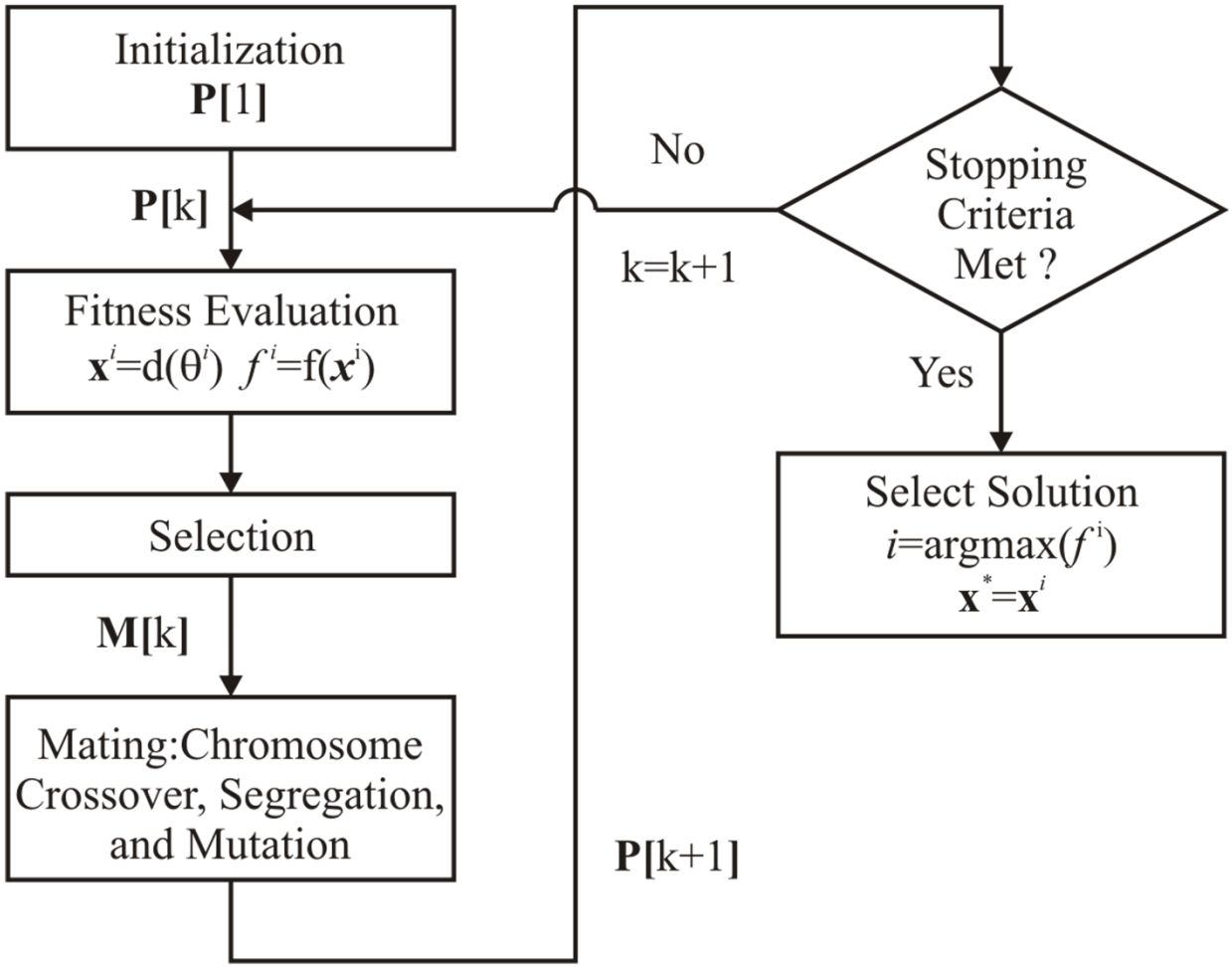
$$f(\mathbf{x}) = \frac{1}{(x_1 - 1)^2 + (x_2 + 2)^2 + x_3^2 + 1}$$

Determine the fitness of population member 13.

1.5 The Canonical Genetic Algorithm

1.5 The Canonical Genetic Algorithm

- Overview of the Canonical Genetic Algorithm



1.5 The Canonical Genetic Algorithm

- Roulette Wheel Selection. The probability of the *i*'th individual getting into the mating pool is given by

$$p^i = \frac{f^i}{\sum_{i=1}^{N_p} f^i}$$

1.5 The Canonical Genetic Algorithm

- N-Way Tournament Selection

1.5 The Canonical Genetic Algorithm

- Mating, Crossover, Segregation, Mutation

for i =1 to $N_p/2$

compute element indices

$i_1 = 2i - 1$

$i_2 = 2i$

get genetic codes of parents

$\boldsymbol{\theta}^{p1}$ = i_1'th individual in $\mathbf{M}[k]$

$\boldsymbol{\theta}^{p2}$ = i_2'th individual in $\mathbf{M}[k]$

apply genetic operators

apply crossover to $\{\boldsymbol{\theta}^{p1}, \boldsymbol{\theta}^{p2}\}$ yielding $\{\boldsymbol{\theta}^{a1}, \boldsymbol{\theta}^{a2}\}$

segregate chromosomes of $\{\boldsymbol{\theta}^{a1}, \boldsymbol{\theta}^{a2}\}$ yielding $\{\boldsymbol{\theta}^{b1}, \boldsymbol{\theta}^{b2}\}$

apply mutation to $\{\boldsymbol{\theta}^{b1}, \boldsymbol{\theta}^{b2}\}$ yielding $\{\boldsymbol{\theta}^{c1}, \boldsymbol{\theta}^{c2}\}$

place children into next population

the i_1'th individual of $\mathbf{P}[k+1]$ becomes $\boldsymbol{\theta}^{c1}$

the i_2'th individual of $\mathbf{P}[k+1]$ becomes $\boldsymbol{\theta}^{c2}$

end

1.5 The Canonical Genetic Algorithm

- Example 1.5B

$$
\begin{array}{cccccccc}
-+\boldsymbol{\theta}_{\cdot}^{p1}+/ & -+\boldsymbol{\theta}_{\cdot}^{p2}+/ & & -+\boldsymbol{\theta}_{\cdot}^{a1}+/ & -+\boldsymbol{\theta}_{\cdot}^{a2}+/ & & -+\boldsymbol{\theta}_{\cdot}^{b1}+/ & -+\boldsymbol{\theta}_{\cdot}^{b2}+/ \\
\begin{matrix} c_1\{g_1\{0 \\ 1 \end{matrix} & \begin{matrix} c_1\{g_1\{1 \\ 1 \end{matrix} & & \begin{matrix} c_1\{g_1\{0 \\ \mathbf{1} \end{matrix} & \begin{matrix} c_1\{g_1\{1 \\ \mathbf{1} \end{matrix} & & \begin{matrix} c_1\{g_1\{\mathbf{1} \\ \mathbf{1} \end{matrix} & \begin{matrix} c_1\{g_1\{\mathbf{0} \\ \mathbf{1} \end{matrix} \\
\begin{matrix} g_2\{1 \\ 1 \end{matrix} & \begin{matrix} g_2\{0 \\ 1 \end{matrix} & & \begin{matrix} g_2\{\mathbf{0} \\ \mathbf{1} \end{matrix} & \begin{matrix} g_2\{\mathbf{1} \\ \mathbf{1} \end{matrix} & & \begin{matrix} g_2\{\mathbf{1} \\ \mathbf{1} \end{matrix} & \begin{matrix} g_2\{\mathbf{0} \\ \mathbf{1} \end{matrix} \\
\begin{matrix} c_2\{g_3\{1 \\ 0 \end{matrix} & \begin{matrix} c_2\{g_3\{0 \\ 0 \end{matrix} & & \begin{matrix} c_2\{g_3\{1 \\ 0 \end{matrix} & \begin{matrix} c_2\{g_3\{0 \\ 0 \end{matrix} & & \begin{matrix} c_2\{g_3\{\mathbf{1} \\ \mathbf{0} \end{matrix} & \begin{matrix} c_2\{g_3\{\mathbf{0} \\ \mathbf{0} \end{matrix} \\
\begin{matrix} g_4\{1 \\ 1 \end{matrix} & \begin{matrix} g_4\{1 \\ 0 \end{matrix} & & \begin{matrix} g_4\{1 \\ 1 \end{matrix} & \begin{matrix} g_4\{1 \\ 0 \end{matrix} & & \begin{matrix} g_4\{\mathbf{1} \\ \mathbf{1} \end{matrix} & \begin{matrix} g_4\{\mathbf{1} \\ \mathbf{0} \end{matrix}
\end{array}
\rightarrow \qquad \rightarrow \qquad \rightarrow
$$

$$
\begin{array}{cc}
-+\boldsymbol{\theta}_{\cdot}^{c1}+/ & -+\boldsymbol{\theta}_{\cdot}^{c2}+/ \\
\begin{matrix} c_1\{g_1\{1 \\ 1 \end{matrix} & \begin{matrix} c_1\{g_1\{0 \\ 1 \end{matrix} \\
\begin{matrix} g_2\{1 \\ 1 \end{matrix} & \begin{matrix} g_2\{0 \\ 1 \end{matrix} \\
\begin{matrix} c_2\{g_3\{1 \\ 0 \end{matrix} & \begin{matrix} c_2\{g_3\{\mathbf{1} \\ 0 \end{matrix} \\
\begin{matrix} g_4\{1 \\ 1 \end{matrix} & \begin{matrix} g_4\{1 \\ 0 \end{matrix}
\end{array}
$$

1.5 The Canonical Genetic Algorithm

- Problems with the Canonical Genetic Algorithm

1.6 Real Coded Genetic Algorithms

- The main idea:

1.6 Real Coded Genetic Algorithms

- Decoding for the *j*'th gene

$$x = \mathrm{d}_j(\theta)$$

- For linearly encoded data

$$\mathrm{d}_j(\theta) = x_{mn,j} + \left(x_{mx,j} - x_{mn,j}\right)\theta$$

- Integer Encoding – same as linear except

$$\theta \in \left\{ \frac{0}{x_{mx,j} - x_{mn,j}}, \frac{1}{x_{mx,j} - x_{mn,j}}, \mathrm{B}\ , \frac{x_{mx,j} - x_{mn,j}}{x_{mx,j} - x_{mn,j}} \right\}$$

1.6 Real Coded Genetic Algorithms

- For logarithmic encoded data

$$\mathrm{d}_j(\theta) = x_{mn,j}\left(\frac{x_{mx,j}}{x_{mn,j}}\right)^{\theta}$$

1.6 Real Coded Genetic Algorithms

- Crossover algorithms
  - Single-Point Crossover
  - Simple-Blend Crossover
  - Simulated Binary Crossover
  - Single-Point Simple-Blend

1.6 Real Coded Genetic Algorithms

- Single-Point Crossover

$$\boldsymbol{\theta}^{p1} = \begin{bmatrix} 0.21 & 0.34 & 0.89 & 0.26 & \mathbf{0.01} & \mathbf{0.44} \end{bmatrix}^T$$

W crossover point

$$\boldsymbol{\theta}^{p2} = \begin{bmatrix} 0.92 & 0.45 & 0.02 & 0.92 & \mathbf{0.84} & \mathbf{0.82} \end{bmatrix}^T$$

$$\Downarrow$$

$$\boldsymbol{\theta}^{a1} = \begin{bmatrix} 0.21 & 0.34 & 0.89 & 0.26 & \mathbf{0.84} & \mathbf{0.82} \end{bmatrix}^T$$

$$\boldsymbol{\theta}^{a2} = \begin{bmatrix} 0.92 & 0.45 & 0.02 & 0.92 & \mathbf{0.01} & \mathbf{0.44} \end{bmatrix}^T$$

- Problems with Single-Point Crossover

1.6 Real Coded Genetic Algorithms

- Simple Blend Crossover
  - Consider parents θ^{p1} and θ^{p2}
  - Determine a uniformly distributed random number in range [0,1] denoted υ. Genes of the child are determined as

$$\boldsymbol{\theta}_j^{a1} = \frac{1}{2}(\boldsymbol{\theta}_j^{p1} + \boldsymbol{\theta}_j^{p2}) + \alpha(\boldsymbol{\theta}_j^{p2} - \boldsymbol{\theta}_j^{p1})\upsilon$$

$$\boldsymbol{\theta}_j^{a2} = \frac{1}{2}(\boldsymbol{\theta}_j^{p1} + \boldsymbol{\theta}_j^{p2}) - \alpha(\boldsymbol{\theta}_j^{p2} - \boldsymbol{\theta}_j^{p1})\upsilon$$

where α is a algorithm constant (say 0.5)

1.6 Real Coded Genetic Algorithms

- Simple Blend Crossover (Continued)
  - Vector Simple Blend Crossover
  - Scalar Simple Blend Crossover
  - Observation on gene values
  - Gene repair

1.6 Real Coded Genetic Algorithms

- Simulated Binary Crossover
  - Objective

  - Algorithm. Start with random number υ in [0,1)
  - Define

$$\beta = \begin{cases} (2\upsilon)^{\frac{1}{\eta_c+1}} & \upsilon \le \frac{1}{2} \\ \left(\frac{1}{2(1-\upsilon)}\right)^{\frac{1}{\eta_c+1}} & \upsilon > \frac{1}{2} \end{cases}$$

1.6 Real Coded Genetic Algorithms

- Simulated Binary Crossover (Continued)
  - Next

$$\boldsymbol{\theta}_j^{a1} = \frac{1}{2}\left[(1+\beta)\boldsymbol{\theta}_j^{p1} + (1-\beta)\boldsymbol{\theta}_j^{p2}\right]$$

$$\boldsymbol{\theta}_j^{a2} = \frac{1}{2}\left[(1-\beta)\boldsymbol{\theta}_j^{p1} + (1+\beta)\boldsymbol{\theta}_j^{p2}\right]$$

  - Vector Simulated Binary Crossover
  - Scalar Simulated Binary Crossover
  - Observation on gene values
  - Gene repair

1.6 Real Coded Genetic Algorithms

- Single-Point Simple-Blend Crossover

$$\boldsymbol{\theta}^{p1} = \begin{bmatrix} 0.21 & 0.34 & 0.89 & 0.26 & 0.01 & 0.44 \end{bmatrix}^T$$

$$\boldsymbol{\theta}^{p2} = \begin{bmatrix} 0.92 & 0.45 & 0.02 & 0.92 & 0.84 & 0.82 \end{bmatrix}^T$$

W crossover point

$$\boldsymbol{\theta}^{a1} = \begin{bmatrix} 0.21 & 0.34 & 0.10 & 0.92 & 0.84 & 0.82 \end{bmatrix}^T$$

$$\boldsymbol{\theta}^{a2} = \begin{bmatrix} 0.92 & 0.45 & 0.81 & 0.26 & 0.01 & 0.44 \end{bmatrix}^T$$

1.6 Real Coded Genetic Algorithms

- Methods for Gene Repair
  - Hard Limit Gene Repair
  - Ring Mapped Gene Repair

1.6 Real Coded Genetic Algorithms

- Mutation methods
  - Total Mutation

$$\boldsymbol{\theta}^a = \begin{bmatrix} 0.21 & 0.34 & \mathbf{0.89} & 0.26 & 0.84 & 0.82 \end{bmatrix}^T$$

$$\Downarrow \text{mutation}$$

$$\boldsymbol{\theta}^b = \begin{bmatrix} 0.92 & 0.45 & \mathbf{0.25} & 0.25 & 0.84 & 0.82 \end{bmatrix}^T$$

  - Partial Absolute Mutation

$$\boldsymbol{\theta}_j^b = \boldsymbol{\theta}_j^a + \sigma \, \mathrm{N}(\backslash)$$

1.6 Real Coded Genetic Algorithms

- Partial Relative Mutation

$$\boldsymbol{\theta}_j^b = \boldsymbol{\theta}_j^a (1 + \sigma \, \mathrm{N}(\backslash))$$

- Absolute Vector Mutation

$$\boldsymbol{\theta}^b = \boldsymbol{\theta}^a + \sigma \, \mathrm{N}(\backslash) \mathbf{V}(\backslash)$$

- Relative Vector Mutation

$$\boldsymbol{\theta}_j^b = \boldsymbol{\theta}_j^a (1 + \sigma \, \mathrm{N}(\backslash) \mathbf{V}_j(\backslash)) \quad \forall j \in [1, 2, \mathtt{A} \ , N_g]$$

- Integer Mutation

1.6 Real Coded Genetic Algorithms

- Gene repair
- Uniform Mutation
- Non-Uniform Mutation

1.6 Real Coded Genetic Algorithms

- Example 1.6A. Let us use a GA to find the maximizer of

$$f(\mathbf{x}) = \frac{1}{(x_1 x_2 - 6)^2 + 4(x_2 - 3)^2 + 1}$$

  - We will use linear coding with $x_{mn,1} = x_{mn,2} = 0$ and $x_{mx,1} = x_{mx,2} = 5$
  - We will use 2-way tournament selection
  - We will use simple blend crossover with α=0.5 and a probability of crossover of 0.5

1.6 Real Coded Genetic Algorithms

- Example 1.6A (continued)
  - We will use total mutation with the probability of a gene mutation of 0.1
  - We will consider 3 generations with a population size of 8

1.6 Real Coded Genetic Algorithms

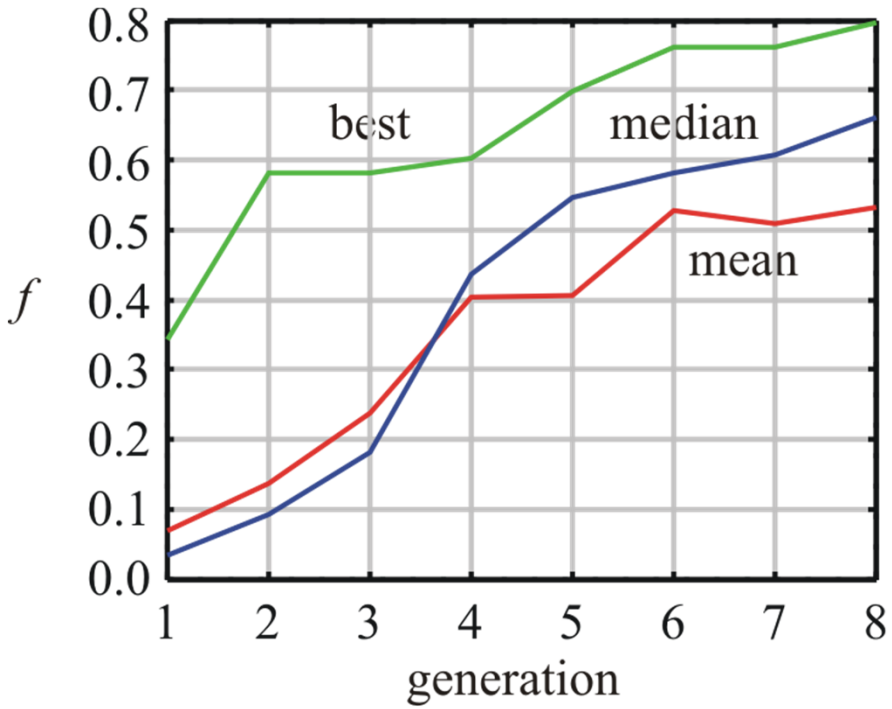
- Example 1.6A results

Table 1.6A-1 Example of real-coded genetic algorithm evolution.

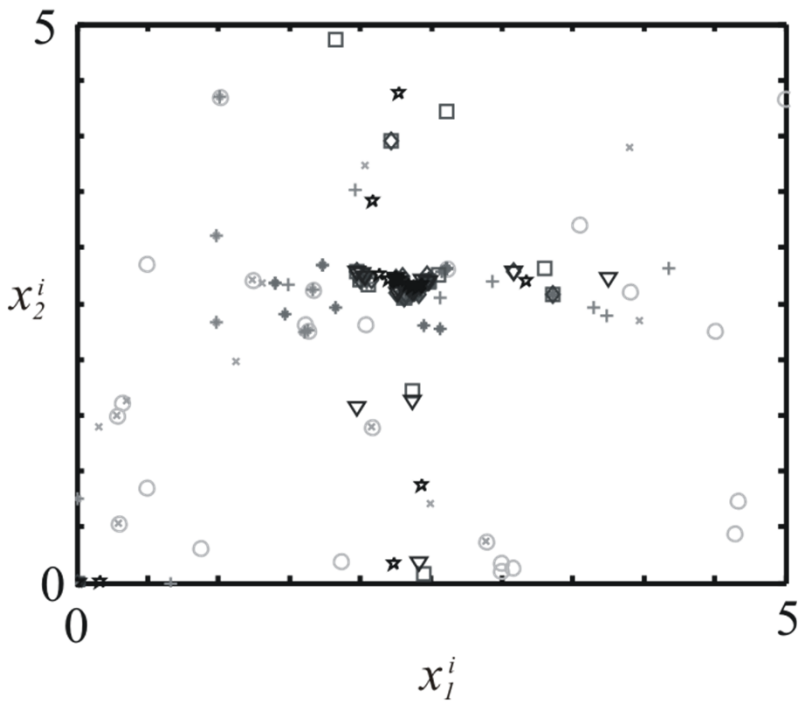
				Generation 1				
$\mathbf{P}[1]$	0.3210	0.8222	0.5718	0.6991	0.4416	0.4657	0.6754	0.9085
	0.8296	0.5707	0.2860	0.7963	0.4462	0.2790	0.9037	0.7472
$\mathbf{x}^i$	1.6051	4.1109	2.8591	3.4957	2.2079	2.3283	3.3769	4.5426
	4.1478	2.8534	1.4301	3.9813	2.2311	1.3952	4.5183	3.7360
f^i	0.1492	0.0295	0.0689	0.0148	0.2213	0.0530	0.0104	0.0081
$\mathbf{M}[1]$	0.5718	0.4657	0.8222	0.3210	0.4657	0.8222	0.4416	0.3210
	0.2860	0.2790	0.5707	0.8296	0.2790	0.5707	0.4462	0.8296
				Generation 2				
$\mathbf{P}[2]$	0.5718	0.1355	0.8975	0.6534	0.4657	0.5279	0.3627	0.8467
	0.2860	0.3321	0.7424	0.6579	0.2790	0.0321	0.6971	0.5786
$\mathbf{x}^i$	2.8591	0.6774	4.4874	3.2668	2.3283	2.6394	1.8133	4.2336
	1.4301	1.6606	3.7118	3.2895	1.3952	0.1604	3.4857	2.8932
f^i	0.0689	0.0313	0.0086	0.0419	0.0530	0.0155	0.4886	0.0249
$\mathbf{M}[2]$	0.1355	0.5718	0.3627	0.5718	0.8467	0.8467	0.5718	0.6534
	0.3321	0.2860	0.6971	0.2860	0.5786	0.5786	0.2860	0.6579
				Generation 3				
$\mathbf{P}[3]$	0.1355	0.5718	0.5462	0.3883	0.8467	0.8467	0.6235	0.6017
	0.3321	0.2860	0.3365	0.6467	0.5786	0.5786	0.5216	0.4223
$\mathbf{x}^i$	0.6774	2.8591	2.7308	1.9417	4.2336	4.2336	3.1174	3.0085
	1.6606	1.4301	1.6824	3.2334	2.8932	2.8932	2.6082	2.1114
f^i	0.0313	0.0689	0.1008	0.7719	0.0249	0.0249	0.1625	0.2335

1.6 Real Coded Genetic Algorithms

- Example 1.6A results (continued)



(a) fitness



(b) gene values

1.6 Real Coded Genetic Algorithms

- Additional operators
  - Scaling (We'll discuss this shortly)
  - Diversity Control
  - Elitism
  - Migration
  - Death

1.6 Real Coded Genetic Algorithms

- Additional operators (continued)
  - Local Search
  - Deterministic Search

1.6 Real Coded Genetic Algorithms

- Scaling
  - Used with Roulette Wheel Selection
  - Early in an evolution
  - Late in an evolution
  - Solution is scaling

1.6 Real Coded Genetic Algorithms

- Several scaling laws take the form $f' = \max(0, af + b)$ where

Table 1.6-1 Linear scaling methods.

Method	a	b	Comments
Offset Scaling	1	$-f_{min}$	Ensures positive fitness
Standard Linear Scaling	$\frac{(k-1)f_{avg}}{f_{max}-f_{avg}}$	$f_{avg}(1-a)$	Most fit individual k times more likely to be in mating pool than average
Modified Linear Scaling	$\frac{(k-1)f_{med}}{f_{max}-f_{med}}$	$f_{med}(1-a)$	Most fit individual k times more likely to be in mating pool than median
Mapped Linear Scaling	$\frac{(k-1)f_{avg}}{f_{max}-f_{\min}}$	$-af_{min}+1$	Minimum fitness mapped to 1; maximum fitness to k
Sigma Truncation	1	$-(f_{avg}-kf_{std})$	Average fitness maps to kf_{std}

1.6 Real Coded Genetic Algorithms

- Consider Standard Linear Scaling

1.6 Real Coded Genetic Algorithms

- Quadratic Scaling

$$f' = af^2 + bf + c$$

where

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} f_{max}^2 & f_{max} & 1 \\ f_{avg}^2 & f_{avg} & 1 \\ f_{min}^2 & f_{min} & 1 \end{bmatrix}^{-1} \begin{bmatrix} k_{max} \\ 1 \\ k_{min} \end{bmatrix}$$

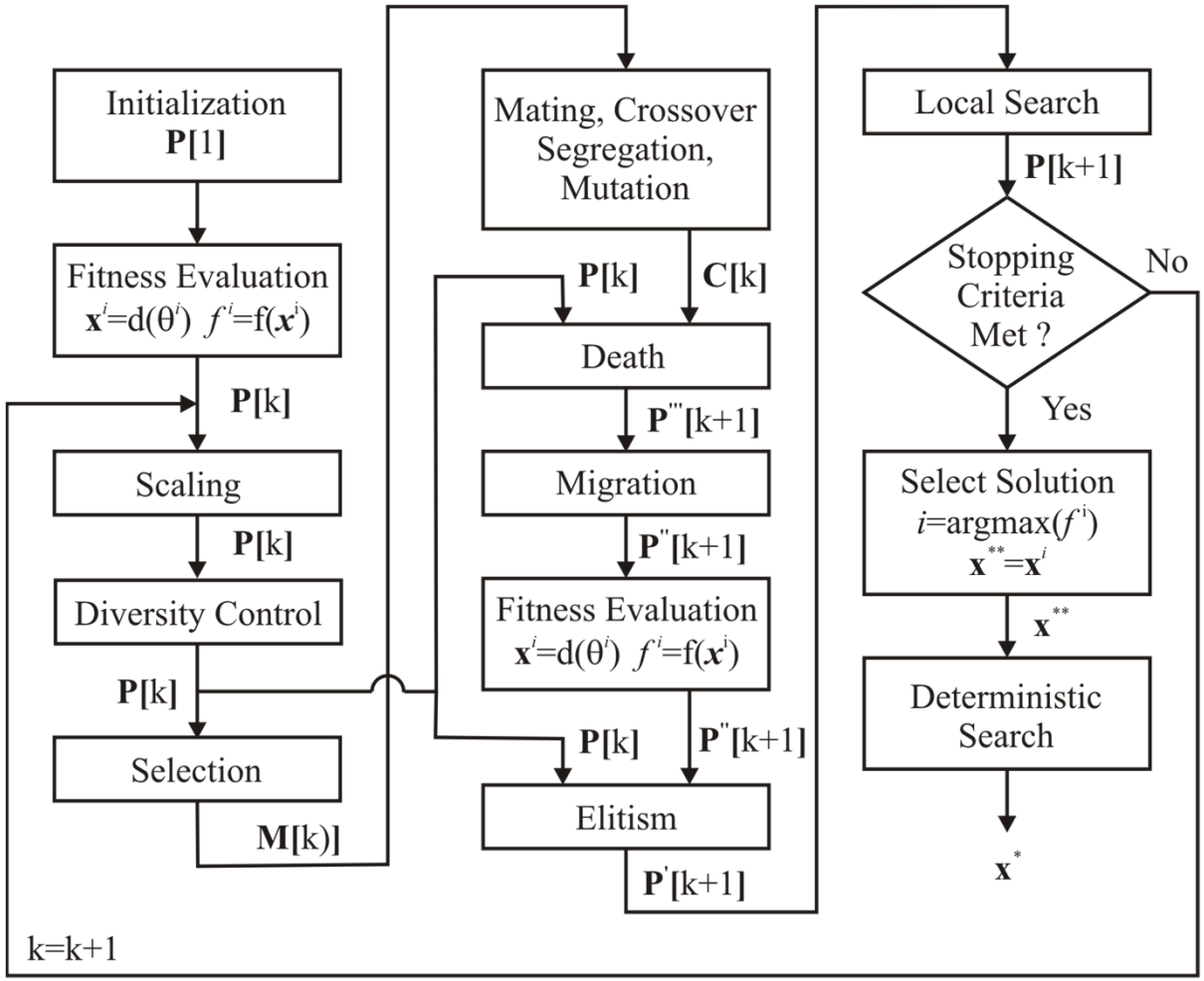
- An individual with maximum fitness is k_{max} times as likely to be in the mating pool as an average fitness individual
- An individual with minimum fitness is k_{min} times as likely to be in the mating pool as the average fitness individual

1.6 Real Coded Genetic Algorithms

- Quadratic Scaling

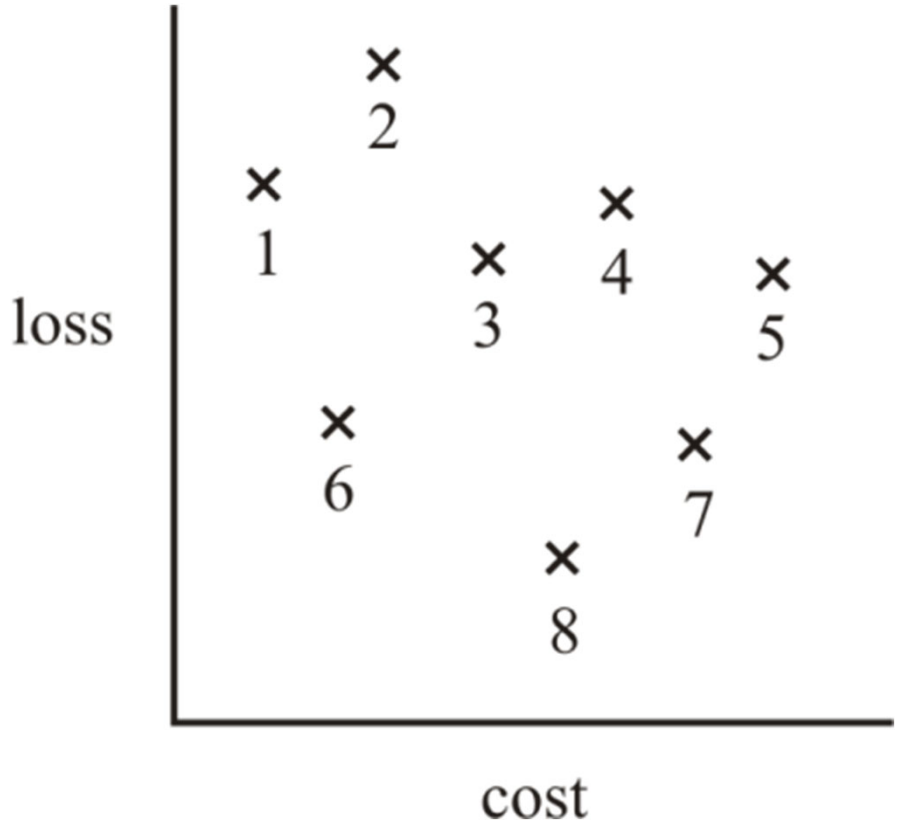
1.6 Real Coded Genetic Algorithms

- Enhanced Real-Coded Genetic Algorithm



1.7 Multi-Obj. Opt. and the P.O. Front

- Consider a set of motors. Suppose we want to minimize coss and loss. Which is best?



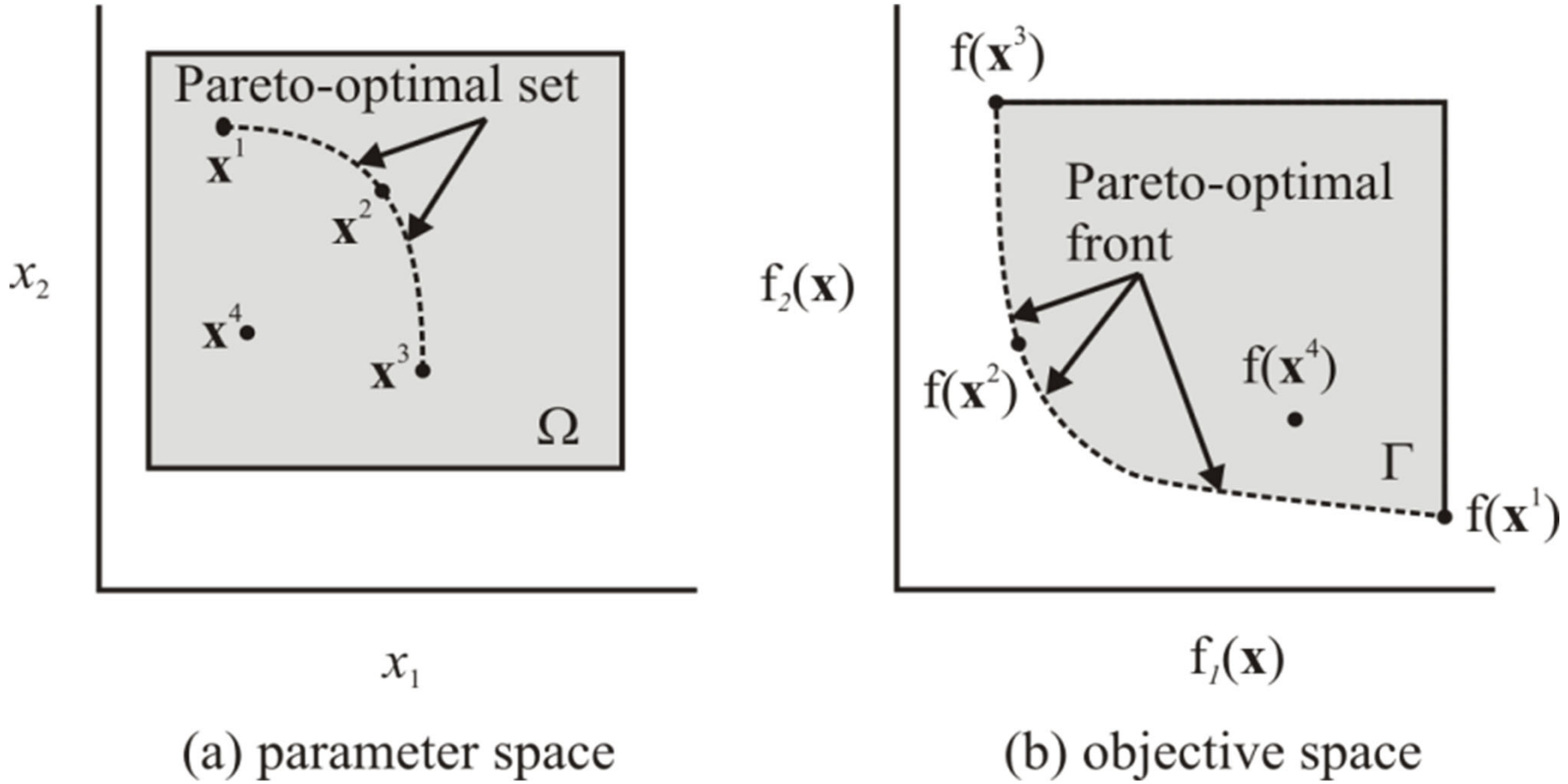
1.7 Multi-Obj. Opt. and the P.O. Front

- Definition of Dominance

- Definition of a Non-Dominated Set

1.7 Multi-Obj. Opt. and the P.O. Front

- Pareto-Optimal Set and Pareto-Optimal Front



1.7 Multi-Obj. Opt. and the P.O. Front

- Methods to Perform Multi-Objective Optimization
  - Weighted Sum Method
  - ε-Constraint Method
  - Weighted Metric Method

1.7 Multi-Obj. Opt. and the P.O. Front

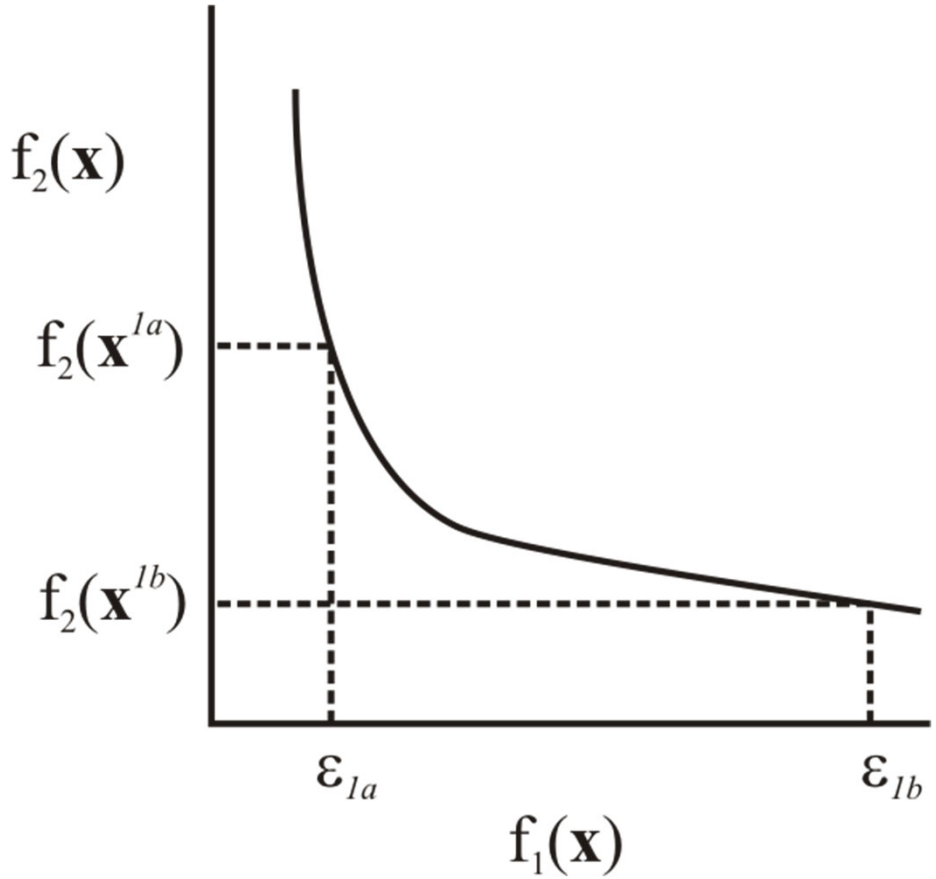
- The ε-Constraint Method
  - Approach: repeatedly solve

$$\begin{aligned} &\text{minimize} \quad f_2(\mathbf{x}) \\ &\text{subject to} \quad f_1(\mathbf{x}) \le \varepsilon \end{aligned}$$

    for different values of ε
  - This yields the Pareo-Optimal front one point (or one single-objective optimization) at a time

1.7 Multi-Obj. Opt. and the P.O. Front

- The ε-Constraint Method (continued)



1.8 Multi-Obj. Opt. Using GAs

- Non-Elitist Strategies
- Elitist Strategies
  - Elitist Non-Dominated Sorting GA
  - Distance-Based Pareto GA
  - Strength Pareto GA
  - **Elitist Non-Dominated Sorting GA (NSGA-II)**

1.8 Multi-Obj. Opt. Using GAs

- Before setting forth this NSGA-II, we need to discuss
  - Kung's method of finding a non-dominated set
  - Non-Dominated Sorting (NDS)
  - Crowding distance
  - Crowding Distance Sorting (CDS)
  - Crowding Tournament Selection (CTS)

1.8 Multi-Obj. Opt. Using GAs

- Kung's Method to find non-dominated solutions
  - Let **P** be the population
  - Let **E** be the set of non-dominated individuals
  - Algorithm
    - Step 1: Sort **P** from best to worse in terms of first objective. Call result $\mathbf{P}_s$.
    - Step 2: **E**=front($\mathbf{P}_s$)

1.8 Multi-Obj. Opt. Using GAs

- Description of front()
  - This is a recursive routine
  - Let $\mathbf{P}^{a:b}$ denote the *a*'th to *b*'th individuals within a population **P**
  - |X| is number of elements in set X

1.8 Multi-Obj. Opt. Using GAs

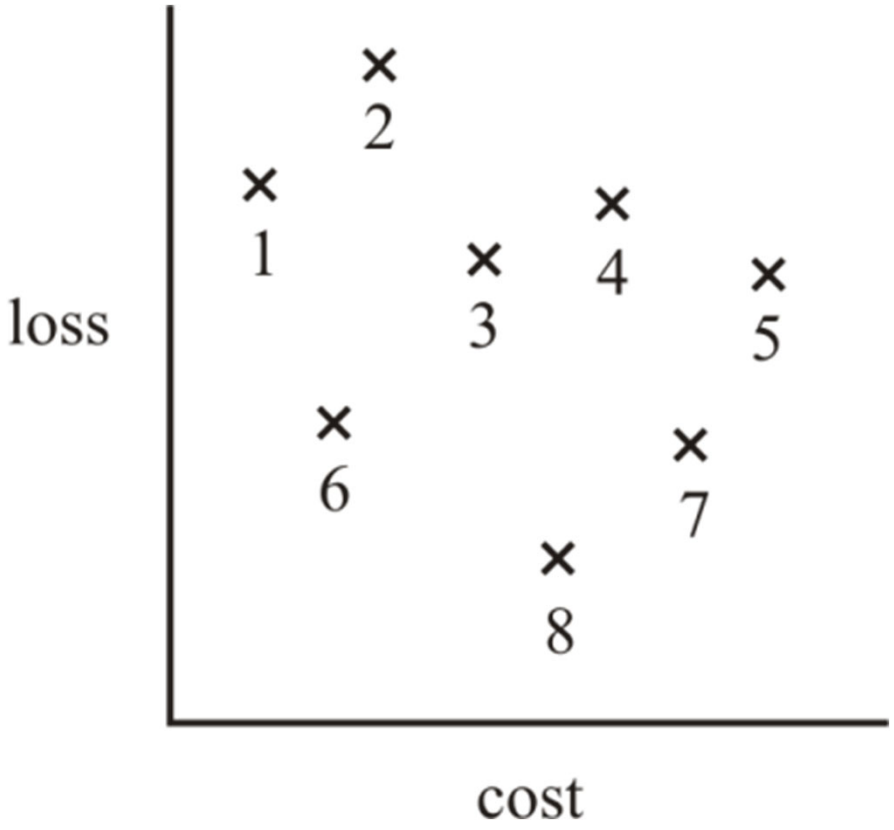
- Pseudo-code for front()

Table 1.8-1 Pseudo-code for Kung's method.

```
function R = front(S)
   if |S| = 1
      R = S
   else
      i = floor(|S|/2)
      T = front(S^(1:i))
      B = front(S^(i+1:|S|))
      N = solutions of B not dominated
          by any solution of T
      R = T ∪ N
   end
end
```

1.8 Multi-Obj. Opt. Using GAs

- Example



$$\mathbf{P} = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$\mathbf{P}_s = \{1, 6, 2, 3, 8, 4, 7, 5\}$$

1.8 Multi-Obj. Opt. Using GAs

- Example (continued)

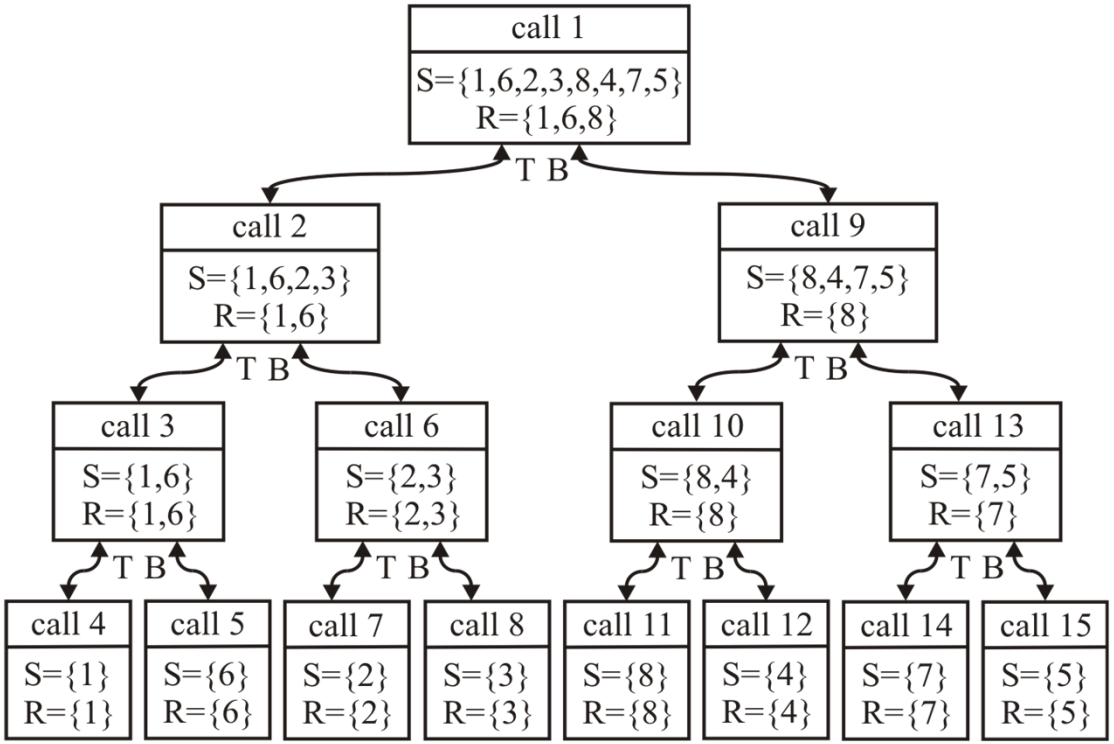
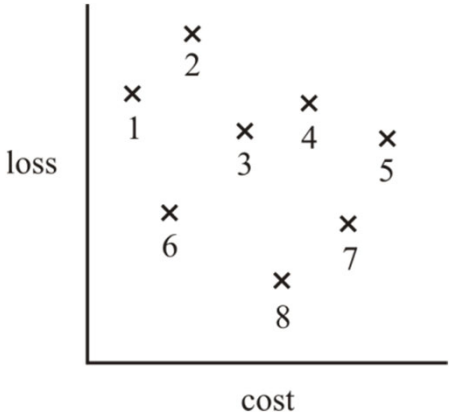


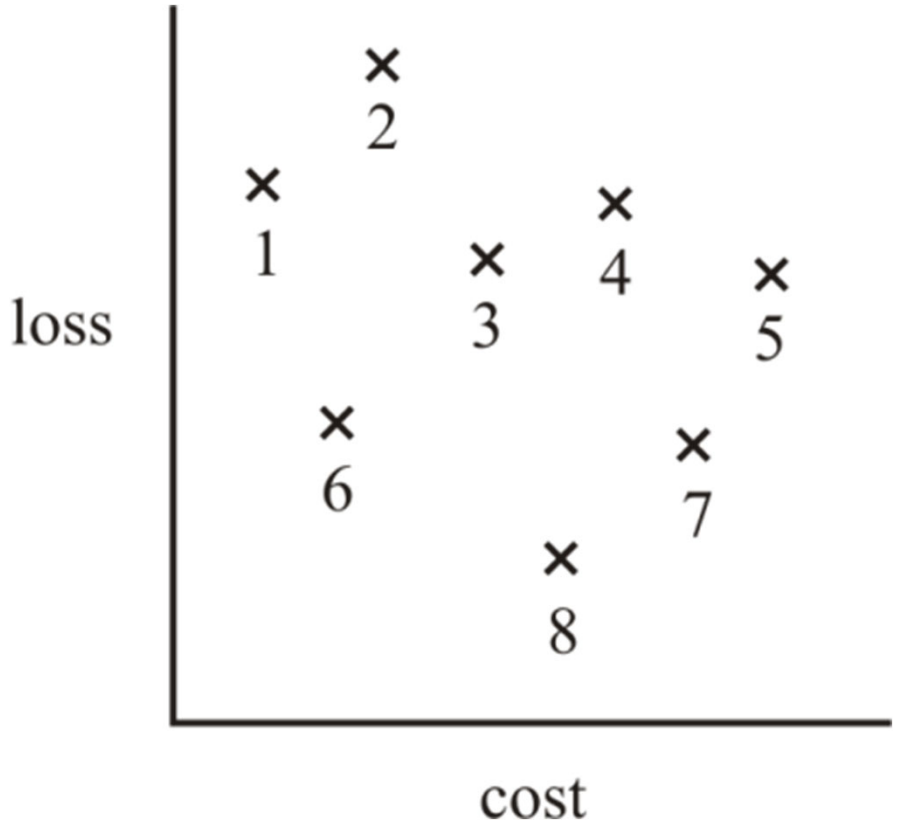
Table 1.8-1 Pseudo-code for Kung's method.

```
function R = front(S)
   if |S|=1
      R = S
   else
      i = floor(|S|/2)
      T = front(S^[1:i])
      B = front(S^[i+1:|S|])
      N = solutions of B not dominated
          by any solution of T
      R = T∪N
   end
end
```



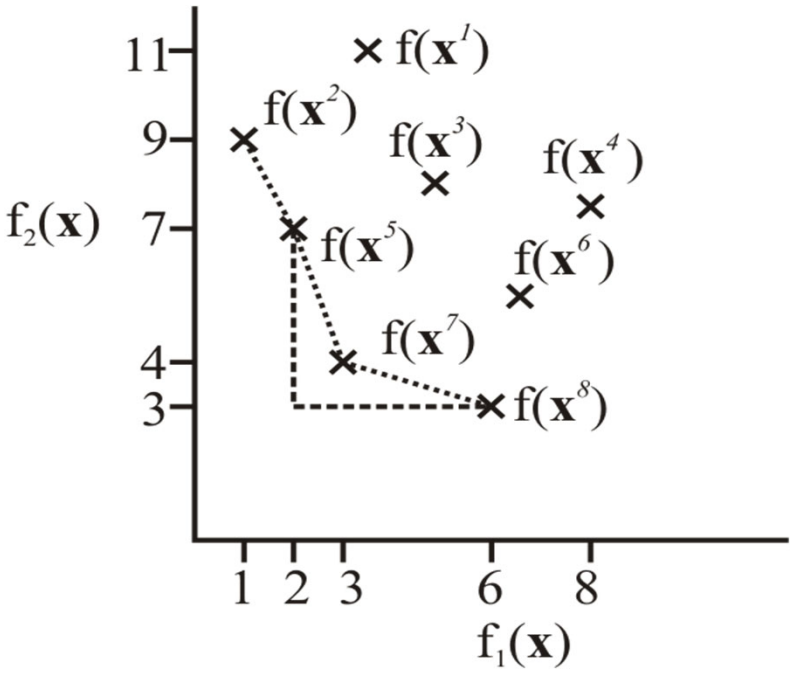
1.8 Multi-Obj. Opt. Using GAs

- Non-Dominated Sorting (NDS)



1.8 Multi-Obj. Opt. Using GAs

- Crowding distance



$$c^i = \sum_{o=1}^{O} \frac{f_o^{N_L(o,i,F_n)} - f_o^{N_S(o,i,F_n)}}{\max_{j \in P}(f_o^j) - \min_{j \in P}(f_o^j)} \qquad i \in F_n$$

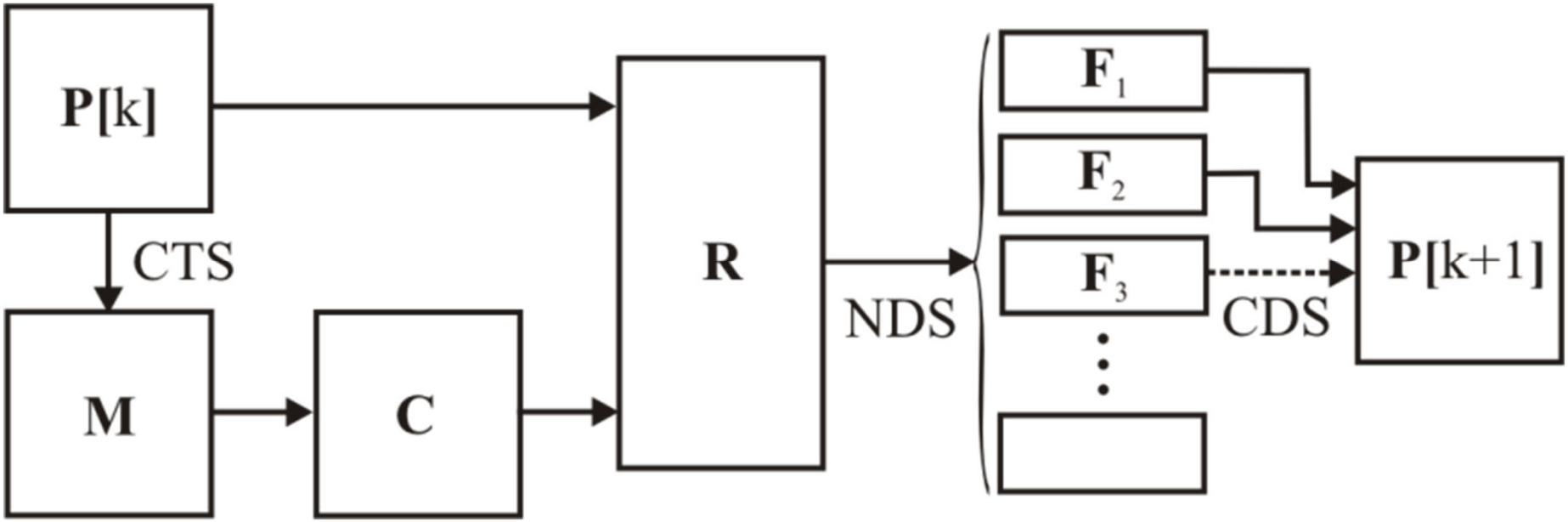
1.8 Multi-Obj. Opt. Using GAs

- Crowding Distance Sorting (CDS)
  - Sort a set of designs (normally on the same front) in terms of decreasing crowding distance

- Crowding Tournament Selection (CTS)
  - Pick two member of population
  - If on different fronts, one on best front wins
  - If on the same front, one with the highest crowding distance wins

1.8 Multi-Obj. Opt. Using GAs

- Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

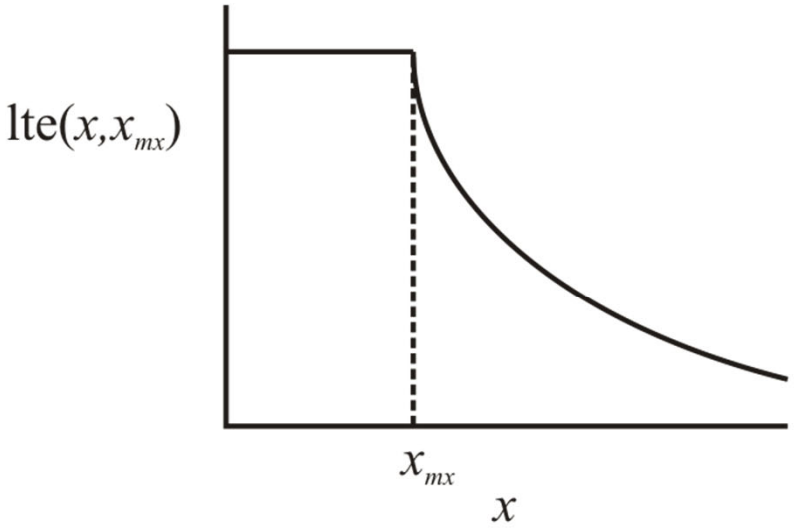


1.9 Formulation of Fitness Functions

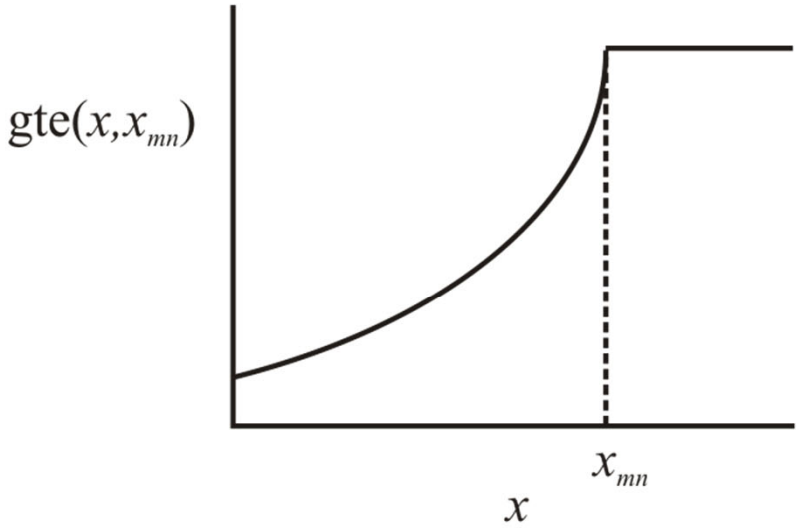
- Setting up constraints

$$\text{lte}(x, x_{mx}) = \begin{cases} 1 & x \le x_{mx} \\ \dfrac{1}{1 + x - x_{mx}} & x > x_{mx} \end{cases}$$

$$\text{gte}(x, x_{mn}) = \begin{cases} 1 & x \ge x_{mn} \\ \dfrac{1}{1 + x_{mn} - x} & x < x_{mn} \end{cases}$$



(a) less-than-or-equal-to funtion



(b) greater-than-or-equal-to function

1.9 Formulation of Fitness Functions

- Aggregation (or averaging) of constraints

$$\bar{c} = \frac{1}{C}\sum_{i=1}^{C} c_i$$

- We can then define a fitness function as

$$f_i = \begin{cases} \varepsilon(\bar{c} - 1) & \bar{c} < 1 \\ m_i & \bar{c} = 1 \end{cases} \quad \text{(maximize metric)}$$

$$f_i = \begin{cases} \varepsilon(\bar{c} - 1) & \bar{c} < 1 \\ \dfrac{1}{m_i} & \bar{c} = 1 \end{cases} \quad \text{(minimize metric)}$$

1.9 Formulation of Fitness Functions

- An approach to minimize computational effort

calculate constraint c_i

$C_I = C_I + 1$

$C_S = C_S + c_i$

if $(C_S < C_I)$

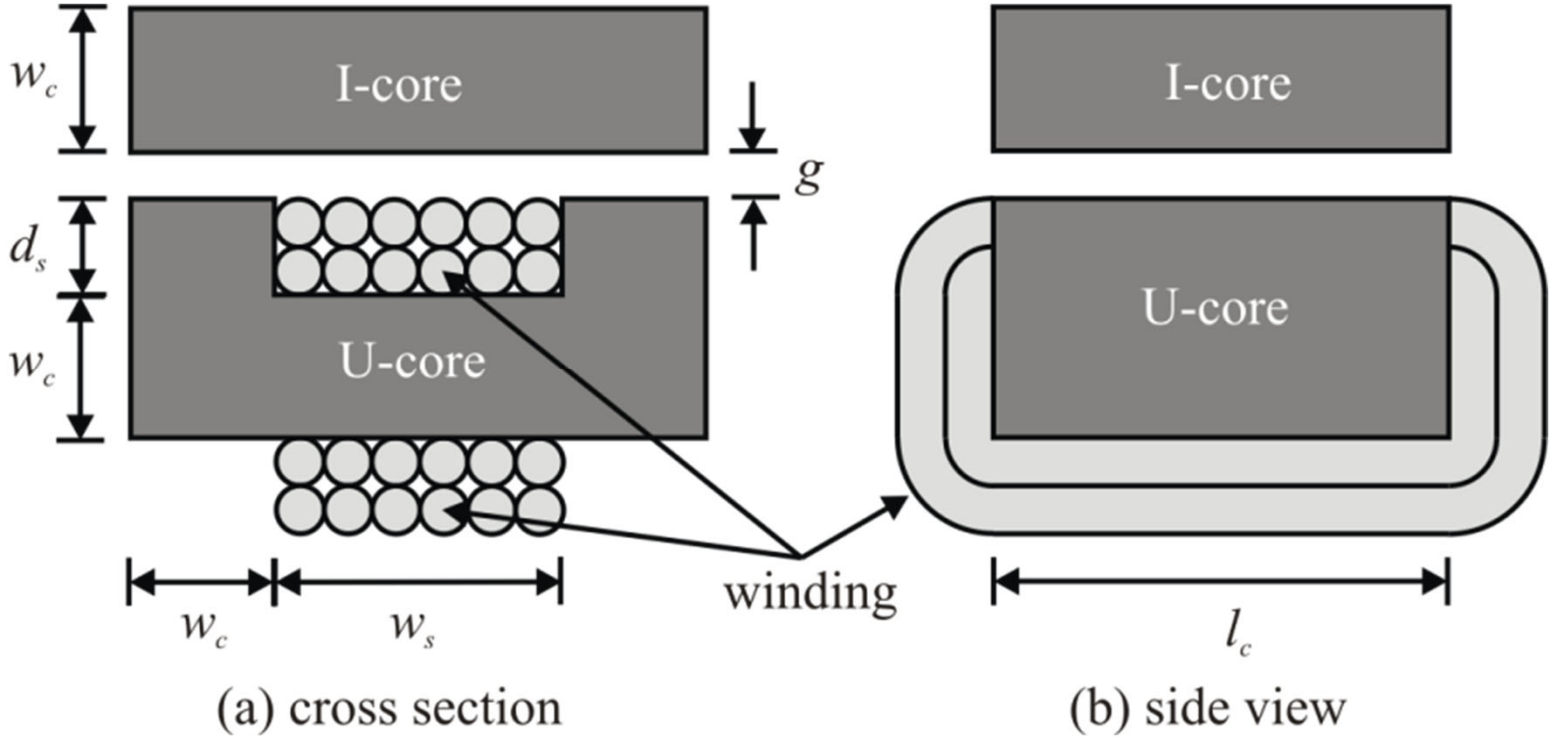
$$\mathbf{f} = \varepsilon\left(\frac{C_S - C}{C}\right)\begin{bmatrix}1 & 1 & \text{B} & 1\end{bmatrix}^T$$

return

end

1.10 A Design Example

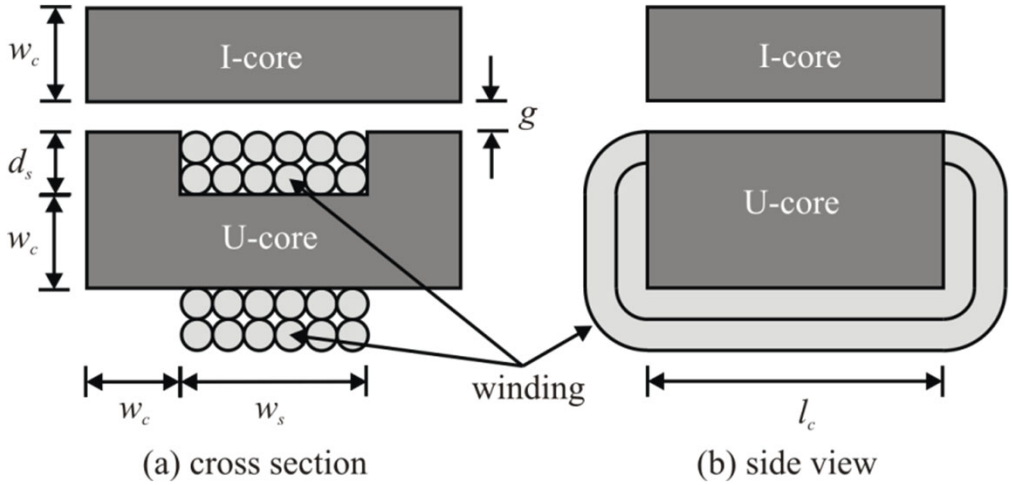
- Consider a UI core inductor



1.10 A Design Example

- Free parameters

$$\mathbf{x} = \begin{bmatrix} N^* & d_s & w_s & w_c & l_c & g \end{bmatrix}^T$$



1.10 A Design Example

- Analysis

$$N = \text{round}(N^*)$$

$$M = 2(2w_c + w_s + d_s)l_c w_c \rho_{mc} + (2l_c + 2w_c + \pi d_s)d_s w_s k_{pf} \rho_{wc}$$

$$P_{rt} = \frac{(2l_c + 2w_c + \pi d_s)N^2 i_{rt}^2}{d_s w_s k_{pf} \sigma_{wc}}$$

$$L = \frac{\mu_0 l_c w_c N^2}{2g}$$

$$B_{rt} = \frac{\mu_0 N i_{rt}}{2g}$$

$$J_{rt} = \frac{Ni}{w_s d_s k_{pf}}$$

1.10 A Design Example

- Constraints

$$c_1 = \text{gte}(L, L_{mn})$$

$$c_2 = \text{lte}(B_{rt}, B_{mx})$$

$$c_3 = \text{lte}(J_{rt}, J_{mx})$$

$$c_4 = \text{lte}(P_{rt}, P_{mx})$$

$$c_5 = \text{lte}(M, M_{mx})$$

$$\overline{c} = \frac{1}{5}\left(c_1 + c_2 + c_3 + c_4 + c_5\right)$$

1.10 A Design Example

- Fitness functions

$$f = \begin{cases} \varepsilon(\overline{c} - 1) & \overline{c} < 1 \\ \dfrac{1}{M} & \overline{c} = 1 \end{cases}$$

$$f = \begin{cases} \varepsilon(\overline{c} - 1)\begin{bmatrix} 1 & 1 \end{bmatrix}^T & \overline{c} < 1 \\ \begin{bmatrix} \dfrac{1}{M} & \dfrac{1}{P_{rt}} \end{bmatrix}^T & \overline{c} = 1 \end{cases}$$

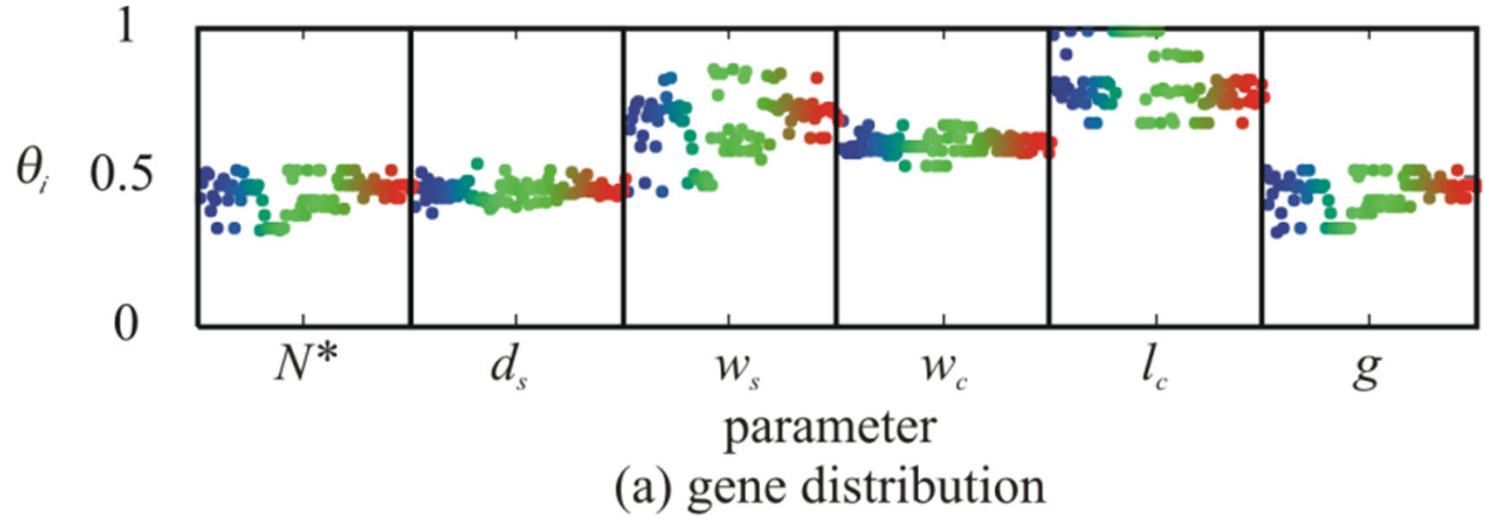
1.10 A Design Example

- Domain of Design Parameters

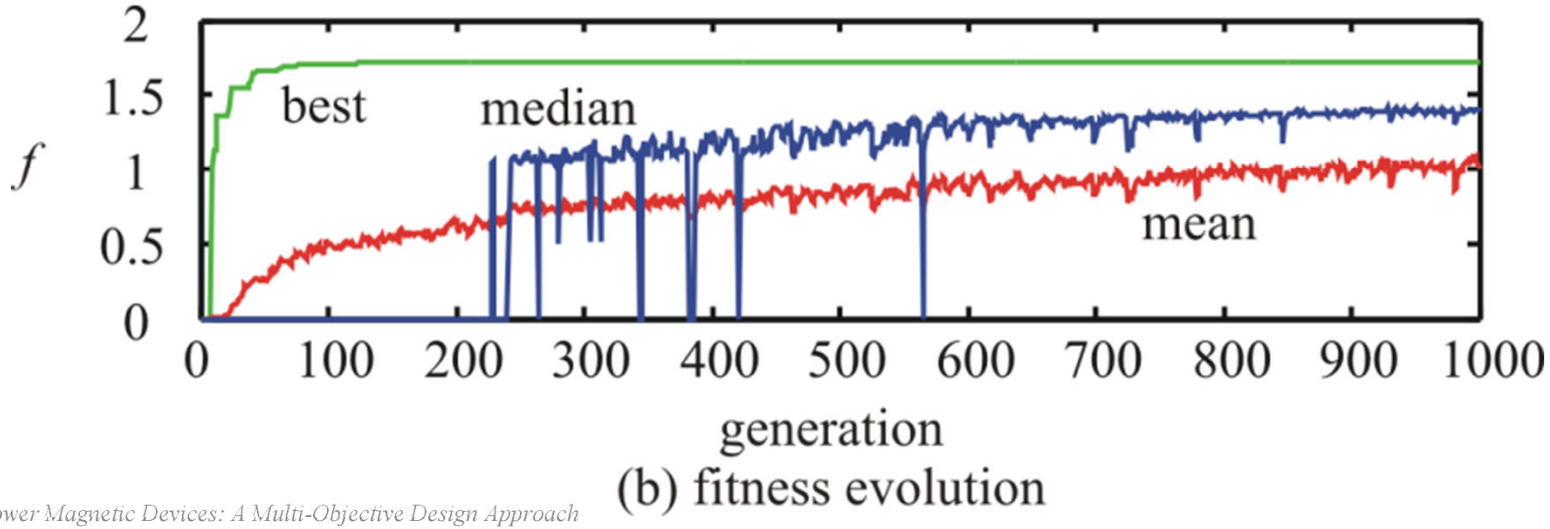
parameter	N	d_s (m)	w_s (m)	w_c (m)	l_c (m)	g (m)
min. value	1	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-5}
max. value	10^{3}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	10^{-2}
encoding	log	log	log	log	log	log
chromosome	1	1	1	1	1	1

1.10 A Design Example

- Single Objective Optimization Study



(a) gene distribution

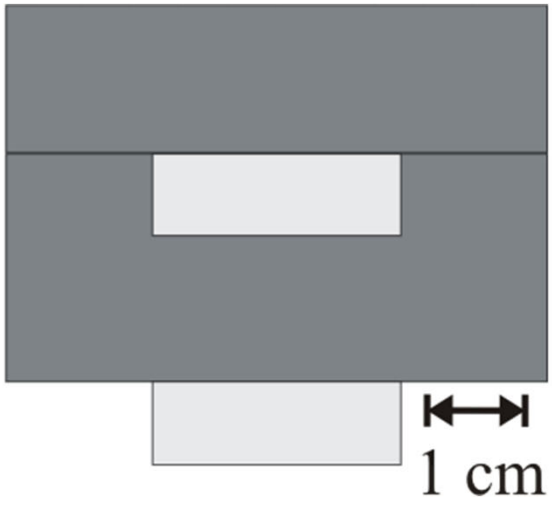


(b) fitness evolution

1.10 A Design Example

- UI-core design

Turns $N = 25$ ($N^* = 25.3$)
Slot depth d_s (mm) = 8.39
Slot width w_s (mm) = 25.5
Core width w_c (mm) = 15.0
Core length l_c (mm) = 43.3
Air gap g (mm) = 0.255
Current density (A/mm^2) = 1.67
Flux density (T) = 0.617
Inductance (mH) = 1
Mass (kg) = 0.578

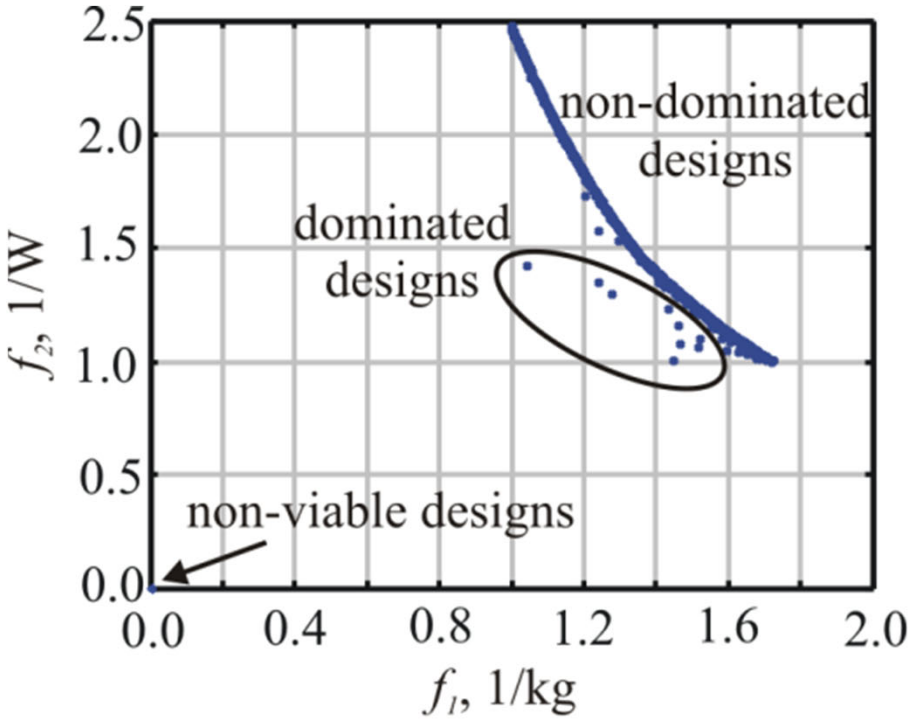


1.10 A Design Example

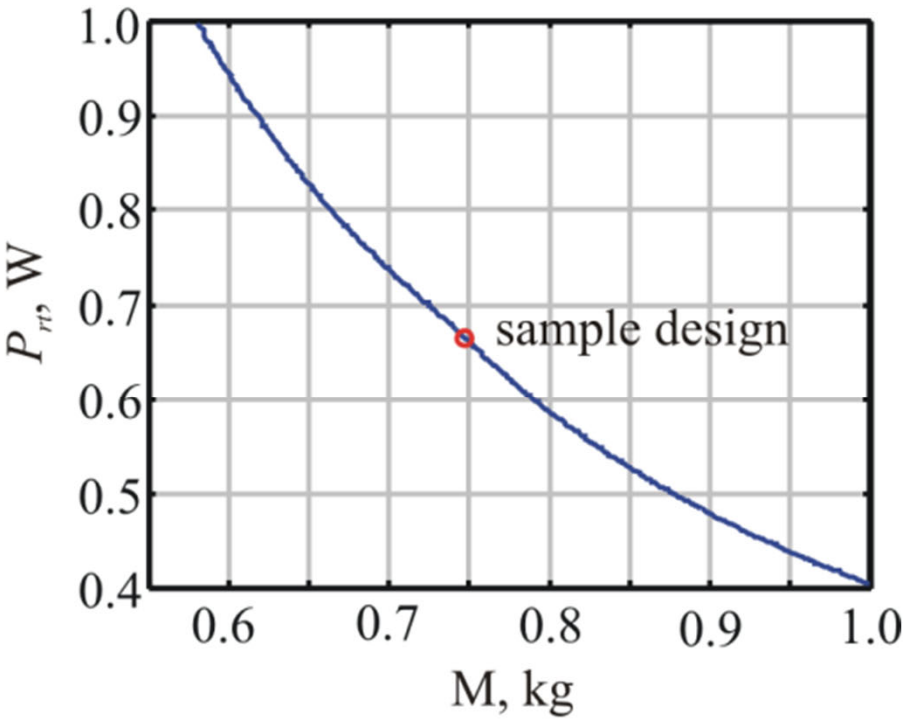
- Is the design process repeatable?
- Running 100 studies yields standard deviations (normalized to means)
  - N: 11%
  - d_s: 4.4%
  - w_s: 17%
  - w_c: 6.4%
  - l_c: 14%
  - g: 11%
  - M: 1%

1.10 A Design Example

- Multi-objective optimization



(a) objective space



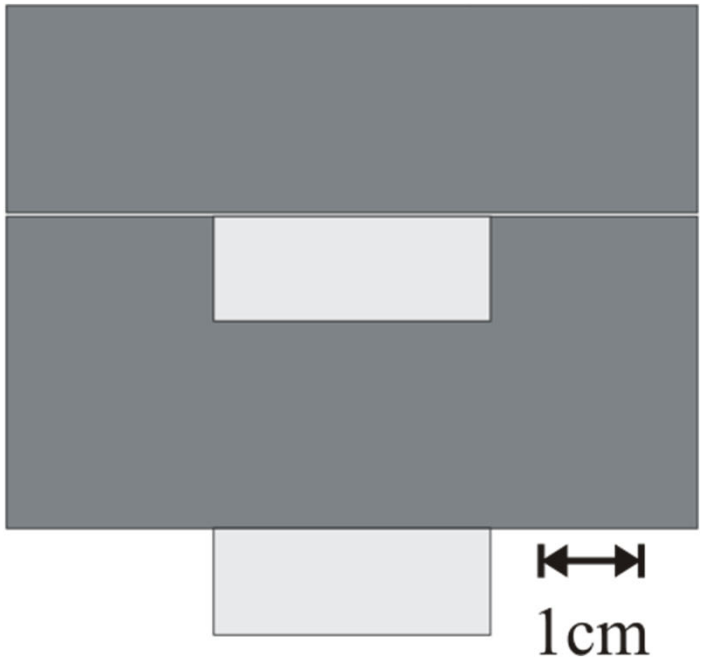
(b) Pareto-optimal front

1.10 A Design Example

- UI-core design

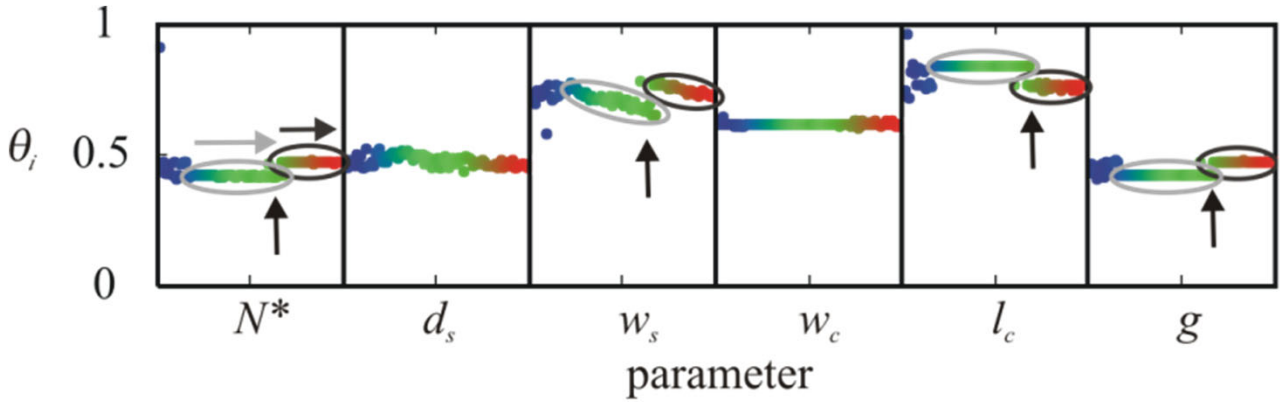
Turns $N = 19$ ($N^* = 19.5$)
Slot depth d_s (mm) = 9.87
Slot width w_s (mm) = 24.5
Core width w_c (mm) = 17.5
Core length l_c (mm) = 48.9
Air gap g (mm) = 0.194

Current density (A/mm^2) = 1.30
Flux density (T) = 0.617
Inductance (mH) = 1
Mass (kg) = 0.75

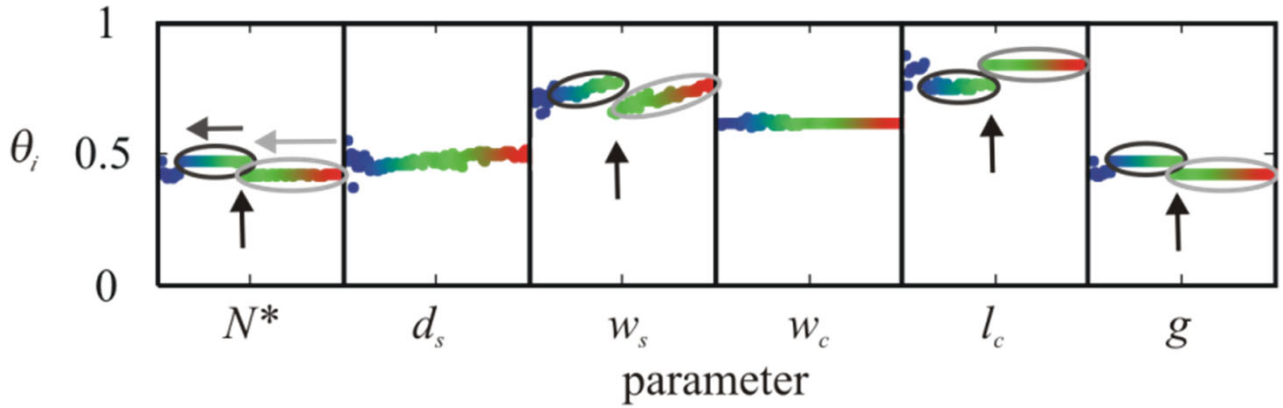


1.10 A Design Example

- Multi-objective optimization



(a) gene distribution sorted by objective 1



(b) gene distribution sorted by objective 2