Invited Talk

Network Models and Optimization:
Multiobjective GA Approach

Dept. of Industrial Eng. & Management
Yuan Ze University
March, 2009

Mitsuo Gen
Grad. School of Info., Prod. & Sys.
Waseda University
gen@waseda.jp

Table of Contents

1. Multiobjective Genetic Algorithms
2. Basic Network Models
3. Logistics Network Models
4. Communication Network Models
5. Advanced Planning and Scheduling Models
6. Project Scheduling Models
7. Assembly Line Balancing Models
8. Tasks Scheduling Models
9. Advanced Network Models

Network Models and Optimization: moGA

Series: Decision Engineering
Mitsuo Gen, Runwei Cheng, Lin Lin
2008, Approx. 710 p. 512 illus., Hardcover
ISBN: 978-1-84800-180-0

Keywords:
Genetic Algorithms, Optimization
Network Models, Logistics Models,
Scheduling Models

Chapter 1: Multiobjective Genetic Algorithms

Mitsuo Gen
Grad. School of Info., Prod. & Sys.
Waseda University
gen@waseda.jp
Chapter 1 Multiobjective Genetic Algorithms

1.1 Introduction

1.1.1 General Structure of Genetic Algorithm
1.1.2 Exploitation and Exploration
1.1.3 Population-based Search
1.1.4 Major Advantages

1.2 Implementation of Genetic Algorithms

1.2.1 GA Vocabulary
1.2.2 Encoding Issue
1.2.3 Fitness Evaluation
1.2.4 Genetic Operators
1.2.5 Handling Constraints

Chapter 1 Multiobjective Genetic Algorithms

1.3 Hybrid Genetic Algorithms

1.3.1 Genetic Local Search
1.3.2 Parameter Adaptation

1.4 Multiobjective Genetic Algorithms

1.4.1 Basic Concepts of Multiobjective Optimizations
1.4.2 Features and Implementation of Multiobjective GA
1.4.3 Fitness Assignment Mechanism
1.4.4 Performance Measures
1.4.5 Applications of Multiobjective GA

1.1 Introduction

- Evolutionary Technique is a keyword in Artificial Intelligence (AI) technology:
  - Refers to a synthesis of methodologies from Fuzzy Logic (FL), Neural Networks, Genetic Algorithm (GA) and other Evolutionary Algorithms (EA).
  - In the last decade, these methodologies have jointly provided valuable control tools for systems presenting strong combinatorial optimization problems.
  - By contrast, in most cases plants cannot be handled by traditional control strategies.

- Genetic Algorithms (GA):
  - The GAs, as powerful and broadly applicable stochastic search and optimization techniques, are perhaps the most widely known types of Evolutionary Techniques today.
1.1 Introduction

- **ERP Package System and Genetic Algorithm**
  - **SAP: APO (Advanced Planner and Optimizer) System**
    - Live Cache
    - Model Generator
    - Model Checker
    - GO!
    - Meta-Heuristics
      - Controls general strategy
      - LP solver (CPLEX)
      - Constraint-based Programming
      - Genetic Algorithms

Figure: SAP-APO optimizer architecture


- **Intelligent global SCM solutions**: in use across a variety of industries, including:
  - consumer goods
  - automotive
  - metals
  - high tech
  - electronics

In May, 1997, i2 merged with Think Systems Corporation, developers of leading demand management solutions, and Optimax Systems Corporation, developers of the OptiFlex suite of Genetic Algorithm-based scheduling and sequencing software.

1.1 Introduction

- Tecnomatix’s eM-plant and Genetic Algorithm

Evolutional Process (Multimodal Optimization Problem)

\[
\begin{align*}
\text{max } & \quad f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2) \\
\text{s. t. } & \quad -3.0 \leq x_1 \leq 12.1 \\
& \quad 4.1 \leq x_2 \leq 5.8
\end{align*}
\]

by Mathematica 4.1

- GA can design a jet engine:

- GA can design a power tower structure:

- GA can draw a criminal faces:

- GA can find an optimal multicast route:

Shinkansen Series N700’s body was shaped by Genetic Algorithm simulation
1.1 Introduction

- **Spring Design Problem**
  - **Problem boundary data:**
    - $x_1 \in \{15, 16, 17, \ldots, 35\}$
    - $x_2 \in \{100, 105, 110, \ldots, 200\}$
    - $x_3 \in \{200, 210, 220, \ldots, 450\}$
    - $x_4 \in \{3, 4, 5, \ldots, 20\}$
  - **Mathematical Model**
    - $f_1(x) = \sum_{i=1}^{4} x_i^2$
    - $f_2(x) = x_1 + x_2$
    - $f_3(x) = x_1 + x_3$
  - **Technological constraints**
    - $g_1(x) = x_1^2 - x_2^2 - 2.4 - 1.6 < 0$
    - $g_2(x) = x_3^2 - x_4 < 0$
  - **Objective:**
    - Minimize the volume of wire used to manufacturing the spring
    - Minimize the diameter of the spring
    - Minimize the length of the spring
  - **Under some technological constraints**

1.1.1 Spring Design Problem

- **Mathematical Model**
  - $f_1(x) = \frac{x_1^2}{x_2^2} + x_3^2 + x_4^2$
  - $f_2(x) = x_1 + x_2$
  - $f_3(x) = x_1 + x_3$
  - **Objective:**
    - Minimize the length of the spring
  - **Technological constraints**
    - $g_1(x) = x_1 - x_2 < 0$
    - $g_2(x) = x_3^2 - 2.4 - 1.6 < 0$
    - $g_3(x) = x_1^2 - 9.5 - 5.5 < 0$
  - **Objective:**
    - Minimize the volume of wire used to manufacturing the spring
    - Minimize the diameter of the spring
    - Minimize the length of the spring
  - **Under some technological constraints**

1.1.1 Spring Design Problem

- **GA Parameter Setting**
  - Population size $\text{popSize} = 100$
  - Crossover probability $\mu_C = 0.6$
  - Mutation probability $\mu_M = 0.5$

1.1.1 Spring Design Problem

- **Pareto Optimal Solution of Spring Design Problem**
  - **Table 8.17:** Optimal solution of Spring Design Problem
  - **Comparison between exhaustive search and GA search**
    - **Table 8.18:** Optimal solution of Spring Design Problem
    - **Enumeration**
      - Search space
        - CPU (sec.)
        - GA
        - 206,388
        - 17,400
        - 386
        - 56
1.1 Introduction

Recent Real World Projects in USA

- **Optimization of off-lease vehicle logistics**, Bank of America, $1.2 million.
- **Distribution optimization of used vehicles**, General Motors, $1.3 million.
- **Customer retention optimization**, UPBC, $1.5 million.
- **Media optimization of book advertisements**, World Changers, $2.4 million.
- **Credit card fraud detection**, Bank of America, $240,000.
- **Scheduling optimization of trains**, BNSF, $600,000.
- **Customer behavior modeling**, PKN Orlen, $300,000.
- **Investment strategy simulations**, U.S. Department of Defense, $150,000.
- **Scheduling optimization of armored cars and couriers**, Wells Fargo, $325,000.

source: Copyright: JANA Solutions Inc.

In past few years, the GA community has turned much of its attention to the optimization problems of Industrial Engineering & Operations Research:


A bibliography on genetic algorithms has been collected by Alander.


The best known algorithms in Evolutionary Technique:

- **Genetic Algorithms (GA)**, developed by Dr. Holland.

- **Evolution Strategies (ES)**, developed by Dr. Rechenberg and Dr. Schewfel.

- **Evolutionary Programming (EP)**, developed by Dr. Fogel.

- **Genetic Programming (GP)**, developed by Dr. Koza.

Invention of Genetic Algorithms

Mimicking the natural evolution process, **John Holland** invented **Genetic Algorithms** while studying the mechanics of adaptation in 1962.

John Holland
Ph.D., Professor
University of Michigan
1.1 Introduction

- **Basic GA to solving Real World Problems**

  - Holland's student, a former pipeline worker.
  - First attempt to apply GA to real world optimization problem: natural GAs pipeline operation while he did his Ph.D.

  **David E. Goldberg**
  Ph.D., Professor
  University of Illinois at Urbana-Champaign

- **Adapted GA to solving real world problems**

  - **1990s:** GA reached a stage of some maturity.

  Published in 1992, suggested an extension of GA by incorporating problem domain knowledge into classic GA

  **Zbigniew Michalewicz,**
  Ph.D., Professor,
  School of Computer Science
  University of Adelaide
  Australia

---

**"Genetic Algorithms and Engineering Design"**

by **Mitsuo Gen** and **Runwei Cheng**

- List Price: $140.00
- Our Price: $140.00
- Used Price: $124.44

Availability: Usually ships within 2 to 3 days

Hardcover - January 1, 1997: 432 pages, John Wiley & Sons, NY

**Table of Contents**

- Constrained Optimization Problems.
- Combinatorial Optimization Problems.
- Reliability Optimization Problems.
- Flow-Shop Sequencing Problems.
- Job-Shop Scheduling Problems.
- Machine Scheduling Problems.
- Transportation Problems.
- Facility Layout Design Problems.
- Selected Topics in Engineering Design.

---

**"Genetic Algorithms and Engineering Optimization"**

by **Mitsuo Gen** and **Runwei Cheng**

- List Price: $125.00
- Our Price: $125.00
- Used Price: $110.94

Availability: Usually ships within 24 hours

Hardcover - January 2000: 512 pages, John Wiley & Sons, NY

**Table of Contents**

- Combinatorial Optimization Problems.
- Multiobjective Optimization Problems.
- Fuzzy Optimization Problems.
- Reliability Design Problems.
- Scheduling Problems.
- Advanced Transportation Problems.
- Network Design and Routing.
- Manufacturing Cell Design.

---

**遺伝算法與工程設計**, 科學出版社, 1999

**Foundations of Genetic Algorithms**

by **Mitsuo Gen** and **Runwei Cheng**

WASEDA UNIVERSITY, IPS
1.1.1 General Structure of Genetic Algorithm

- In general, a GA has five basic components, as summarized by Michalewicz.


  1. A genetic representation of potential solutions to the problem.
  2. A way to create a population (an initial set of potential solutions).
  3. An evaluation function rating solutions in terms of their fitness.
  4. Genetic operators that alter the genetic composition of offspring (selection, crossover, mutation, etc.).
  5. Parameter values that genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

- The general structure of GAs
1.1.1 General Structure of Genetic Algorithm

- **Procedure of Basic GA**

  ```
  procedure: basic GA
  input: problem data, GA parameters
  output: the best solution
  begin
      t ← 0;  // t: generation number
      initialize P(t) by encoding routine;  // P(t): population of chromosomes
      evaluate P(t) by decoding routine;
      while (not terminating condition) do
          create C(t) from P(t) by crossover routine;  // C(t): offspring
          create C(t) from P(t) by mutation routine;
          evaluate C(t) by decoding routine;
          select P(t + 1) from P(t) and C(t) by selection routine;
          t ← t + 1;
      end
  output the best solution
  end
  ```

- **Darwin's Legendary Book:**
  - *On the Origin of Species by Means of Natural Selection: or, the Preservation of Favored Races in the Struggle for Life,*

- **Chromosome:**
  - Cells, whether through meiosis or mitosis, share genetic information through chromosomes.

- **Crossover:**
  - How do you get a chromosome that is totally different from the original chromosomes of both parents?
  - Crossover takes place during Meiosis.

- **Mutation**
  - Mutation takes place when an organism undergoes a spontaneous genetic change during replication.
  - During the process of replication, the nucleotides of a chromosome are altered, so rather than creating an identical copy of DNA strands.
1.1.1 General Structure of Genetic Algorithm

**Selection**
- **A. Directional selection**
- **B. Stabilizing selection**
- **C. Disruptive selection**

**Evolution**
- Evolution refers to change over time, or transformation over time.
- Evolution assumes that all natural forms arose from their ancestors and adapted over time to their environments, thus leading to variation.

1.1.1 General Structure of Genetic Algorithm

- **Genetic Representation and Initialization:**
  - The genetic algorithm maintains a population $P(t)$ of chromosomes or individuals $v_j(t), k=1, 2, \ldots, popSize$ for generation $t$.  
  - Each chromosome represents a potential solution to the problem at hand.
- **Evaluation:**
  - Each chromosome is evaluated to give some measure of its fitness $eval(v_j)$.
- **Genetic Operators:**
  - Some chromosomes undergo stochastic transformations by means of genetic operators to form new chromosomes, i.e., offspring.
  - There are two kinds of transformation:
    - **Crossover**, which creates new chromosomes by combining parts from two chromosomes.
    - **Mutation**, which creates new chromosomes by making changes in a single chromosome.
  - New chromosomes, called offspring $C(t)$, are then evaluated.
- **Selection:**
  - A new population is formed by selecting the more fit chromosomes from the parent population and the offspring population.
- **Best solution:**
  - After several generations, the algorithm converges to the best chromosome, which hopefully represents an optimal or suboptimal solution to the problem.

Example with Basic GA
- We explain in detail about how a genetic algorithm actually works with a simple examples.
- We follow the approach of implementation of genetic algorithms given by Michalewicz.

- The numerical example of unconstrained optimization problem is given as follows:

$$\max f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$$

s. t. \quad -3.0 \leq x_1 \leq 12.1 \quad 4.1 \leq x_2 \leq 5.8
1.1.1 General Structure of Genetic Algorithm

max \( f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2) \)

s.t. \(-3.0 \leq x_1 \leq 12.1\)
      \(4.1 \leq x_2 \leq 5.8\)

by Mathematica 4.1

\( f = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2) \).

- **Binary String Encoding** (Chromosome Design)
  - The **precision requirement** implies that the range of domain of each variable should be divided into at least \((b_j - a_j) \times 10^4\) size ranges.
  - The **required bits** (denoted with \(m_i\)) for a variable is calculated as follows:
    \[
    \begin{align*}
    x_1 &: (12.1 - (-3.0)) \times 10,000 = 151,000 \\
    2^{15} &< 151,000 \leq 2^{16}, \quad m_1 = 18 \text{ bits} \\
    x_2 &: (5.8 - 4.1) \times 10,000 = 17,000 \\
    2^{14} &< 17,000 \leq 2^{15}, \quad m_2 = 15 \text{ bits}
    \end{align*}
    \]
  - Precision requirement: \( m = m_1 + m_2 = 18 + 15 = 33 \text{ bits} \)

- **Binary String Decoding**
  - The **mapping** from a binary string to a real number for variable \(x_j\) is completed as follows:

<table>
<thead>
<tr>
<th>Binary Number</th>
<th>Decimal Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000010101001001</td>
<td>5417</td>
</tr>
<tr>
<td>1011110111111110</td>
<td>24318</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  x_j &= a_j + \text{decimal(substring)} \times \frac{b_j - a_j}{2^{m_j} - 1} \\
  x_1 &= -3.0 + 5417 \times \frac{12.1 - (-3.0)}{2^{16} - 1} = -2.687069 \\
  x_2 &= 4.1 + 24318 \times \frac{5.8 - 4.1}{2^{15} - 1} = 5.361653
\end{align*}
\]

- **Procedures of Binary-string Encoding and Binary-string Decoding**

**procedure:** binary-string encoding

**input:** no. of required bits \(m\)

**output:** chromosome \(v_j(\cdot)\)

**begin**

for \(i = 1\) to \(m\)

\( v_j(i) \leftarrow \text{random}[0, 1] \)

**end**

**procedure:** binary-string decoding

**input:** no. of variables \(n\),

no. of bits \(m_j\) (\(j = 1, 2, \ldots, n\))

range \([a_j, b_j]\) of variable \(x_j\) (\(j = 1, 2, \ldots, n\))

**begin**

\( s \leftarrow 0, t \leftarrow 0 \)

for \(j = 1\) to \(n\)

\( s \leftarrow s + t \cdot m_j \)

\( t \leftarrow t + 1 \)

\( x_j = a_j + \frac{b_j - a_j}{2^{m_j} - 1} \sum_{i=1}^{m_j} 2^{i-1} v_i(t) \)

**end**

**output** real number \(x_j\) (\(j = 1, 2, \ldots, n\))

**end**
1.1.1 General Structure of Genetic Algorithm

- **Initial population satisfying system constraints** (Randomly generated)
  \[ v_1 = [00001010101010101010111111111111] \Rightarrow [x_1, x_2] = [-2.687969, 5.361653] \]
  \[ v_2 = [001110110110011010000000101010010000] \Rightarrow [x_1, x_2] = [0.474101, 4.170144] \]
  \[ v_3 = [1100011101000110101010011111011110111101] \Rightarrow [x_1, x_2] = [10.419457, 4.661461] \]
  \[ v_4 = [1001101101010101100000000111111010011] \Rightarrow [x_1, x_2] = [6.159951, 4.109529] \]
  \[ v_5 = [0000101110110100111001111001111111101] \Rightarrow [x_1, x_2] = [-2.301286, 4.477282] \]
  \[ v_6 = [1111101011011010000000110101101101101111] \Rightarrow [x_1, x_2] = [11.788084, 4.174346] \]
  \[ v_7 = [1110100011111100111111101101110100101] \Rightarrow [x_1, x_2] = [9.342067, 5.121702] \]
  \[ v_8 = [001011010100011100111110111011111000100] \Rightarrow [x_1, x_2] = [-0.330256, 4.694977] \]
  \[ v_9 = [11111000110111110011110011011111111101] \Rightarrow [x_1, x_2] = [11.671267, 4.873501] \]
  \[ v_{10} = [11111010111110101000000101111010101] \Rightarrow [x_1, x_2] = [11.446273, 4.171908] \]

- **Evaluation**
  - An evaluation function plays the **role of the environment**, and it rates chromosomes in terms of their **fitness**.
  - The fitness function values of above chromosomes are as follows:
    \[ eval(v_1) = f(-2.687969, 5.361653) = 19.805119 \]
    \[ eval(v_2) = f(0.474101, 4.170144) = 17.370896 \]
    \[ eval(v_3) = f(10.419457, 4.661461) = 9.590546 \]
    \[ eval(v_4) = f(6.159951, 4.109529) = 29.406122 \]
    \[ eval(v_5) = f(-2.301286, 4.477282) = 15.686091 \]
    \[ eval(v_6) = f(11.788084, 4.174346) = 11.900541 \]
    \[ eval(v_7) = f(9.342067, 5.121702) = 17.958717 \]
    \[ eval(v_8) = f(-0.330256, 4.694977) = 19.763190 \]
    \[ eval(v_9) = f(11.671267, 4.873501) = 26.401669 \]
    \[ eval(v_{10}) = f(11.446273, 4.171908) = 10.252480 \]
  - It is clear that chromosome \( v_4 \) is the **strongest** one and that chromosome \( v_9 \) is the **weakest** one.

- **Crossover (One-cut point Crossover)**
  - Crossover used here is one-cut point method, which random selects one cut point.
  - **Exchanges** the right parts of two parents to generate offspring.
  - Consider two chromosomes as follows and the cut point is randomly selected after the 17th gene:

    - **crossing point at 17th gene**
      \[ v_1 = [10011011010101010101100000000101110011] \]
      \[ v_2 = [001110110111101101000000001101010010000] \]

    - **mutating point at 16th gene**
      \[ v_1 = [1001101101010101010110000000010111111111] \]
      \[ c_1 = [10011011010101010111000000010111101001] \]
      \[ c_2 = [00111011011011010100000001011011101000] \]

- **Mutation**
  - **Alters one or more genes** with a probability equal to the mutation rate.
  - Assume that the 16th gene of the chromosome \( v_i \) is selected for a mutation.
  - Since the gene is 1, it would be flipped into 0. So the chromosome after mutation would be:
1.1.1 General Structure of Genetic Algorithm

**Roulette Wheel Selection:**

- **input:** population \( P(t-1) \), \( C(t-1) \)
- **output:** population \( P(t) \), \( C(t) \)
- **step 1:** Calculate the total fitness \( F \) for the population.
  \[ F = \sum_{k=1}^{10} eval(v_k) = 178.135372 \]
- **step 2:** Calculate selection probability \( p_k = \frac{eval(v_k)}{F} \) for each \( v_k \).
  \[ p_1 = 0.111180, \ p_2 = 0.097515, \ p_3 = 0.053839, \ p_4 = 0.165077, \]
  \[ p_5 = 0.088057, \ p_6 = 0.066806, \ p_7 = 0.100815, \ p_8 = 0.110945, \]
  \[ p_9 = 0.148211, \ p_{10} = 0.057554 \]
- **step 3:** Calculate cumulative probability \( q_k = \sum_{j=1}^{k} p_j = q_{k-1} + p_k \) for each \( v_k \).
  \[ q_1 = 0.111180, \ q_2 = 0.208695, \ q_3 = 0.262534, \ q_4 = 0.427611, \]
  \[ q_5 = 0.515668, \ q_6 = 0.582475, \ q_7 = 0.683290, \ q_8 = 0.704234, \]
  \[ q_9 = 0.942446, \ q_{10} = 1.000000 \]
- **step 4:** Generate a random number \( r \) from the range \([0, 1]\).
  
  0.301431, 0.322062, 0.766503, 0.881893, 0.350871, 0.583392, 0.177618, 0.343242, 0.032685, 0.197577

Next Generation

- \( v_1' = [0011101101001110100000010111001] \), \( f(6.159951, 4.109598) = 29.406122 \)
- \( v_2' = [0011101101001110100000010111001] \), \( f(6.159951, 4.109598) = 29.406122 \)
- \( v_3' = [0011101101001110100000010111001] \), \( f(-0.330256, 4.694977) = 19.763190 \)
- \( v_4' = [1111100101111110110100010111101110] \), \( f(11.907206, 4.873501) = 5.702781 \)
- \( v_5' = [0011101101001110100000010111001] \), \( f(8.024130, 4.170248) = 19.91025 \)
- \( v_6' = [11010001011111101100010111101111110] \), \( f(9.34067, 5.121702) = 17.958717 \)
- \( v_7' = [0011101101001110100000010111001] \), \( f(6.159951, 4.109598) = 29.406122 \)
- \( v_8' = [0011101101001110100000010111001] \), \( f(6.159951, 4.109598) = 29.406122 \)
- \( v_9' = [00000101010011011111111111111111110] \), \( f(-2.687969, 5.361653) = 19.805199 \)
- \( v_{10}' = [0011101111101110110000010110110000] \), \( f(0.474101, 4.170248) = 17.370896 \)

It is clear that chromosomes \( v_1, v_2, v_7, v_8 \) are strongest chromosomes.

1.1.1 General Structure of Genetic Algorithm

**Procedure of GA for Unconstrained Optimization**

- **input:** problem data, GA parameters
- **output:** the best solution

1. **begin**
   \( t \leftarrow 0 \);\ 
   initialize \( P(t) \) by binary string encoding;\ 
   evaluate \( eval(P) \) by binary string decoding;\ 
   while (not terminating condition) do\ 
     create \( C(t) \) from \( P(t) \) by one-cut point crossover;\ 
     create \( C(t) \) from \( P(t) \) by replacement mutation;\ 
     evaluate \( eval(C) \) by binary string decoding;\ 
     select \( P(t+1) \) from \( P(t) \) and \( C(t) \) by roulette wheel selection;\ 
     \( t \leftarrow t+1 \);\ 
   end\ 
   output the best solution;\ 
**end**
The test run is terminated after 1000 generations.

We obtained the best chromosome in the 331th generation as follows:

\[
\begin{align*}
\text{max } f(x_1, x_2) &= 21.5 + x_1 \sin(4 \pi x_1) + x_2 \sin(20 \pi x_2) \\
\text{s.t. } -3.0 &\leq x_1 \leq 12.1 \\
&\quad 4.1 \leq x_2 \leq 5.8 \\
\text{eval} (\mathbf{v}^*) &= f(11.631407, 5.724824) = 38.818208 \\
x_1^* &= 11.631407 \\
x_2^* &= 5.724824 \\
f(x_1^*, x_2^*) &= 38.818208
\end{align*}
\]

Search can be performed with either blind strategies or heuristic strategies.

- **Blind search** strategies do not use information about the problem domain.
- **Heuristic search** strategies use additional information to guide search move along with the best search directions.

There are two important issues in search strategies:

- **Exploiting the best solution**
  - Exploits the best solution for possible improvement ignoring the exploration of the search space.
- **Exploring the search space**
  - Explores the search space, ignoring the exploitation of the promising regions of the search space.
1.1.2 Exploitation and Exploration

- GA is a class of general purpose search methods combining elements of directed and stochastic search.

![Diagram showing exploitation and exploration in a fitness landscape](image)

1.1.3 Population-based Search

### Genetic Algorithm (population-to-population approach)

- Genetic algorithms perform a multiple directional search by maintaining a population of potential solutions.
- The population-to-population approach is hoped to make the search escape from local optima.
- Population undergoes a simulated evolution: at each generation the relatively good solutions are reproduced, while the relatively bad solutions die.
- Genetic algorithms use probabilistic transition rules to select someone to be reproduced and someone to die so as to guide their search toward regions of the search space with likely improvement.

#### Non-evolution Method (point-to-point approach)

- Generally, algorithm for solving optimization problems is a sequence of computational steps which asymptotically converge to optimal solution.
- Most of classical optimization methods generate a deterministic sequence of computation based on the gradient or higher order derivatives of objective function.
- The methods are applied to a single point in the search space.
- The point is then improved along the deepest descending direction gradually through iterations.
- This point-to-point approach takes the danger of falling in local optima.

![Diagram showing the non-evolution method](image)
1.1.3 Population-based Search

- Example of Genetic Algorithm for Unconstrained Numerical Optimization (Michalewicz, 1996)

\[
\begin{align*}
\text{max } f(x) &= x \sin(\pi x) + 1 \\
-1.0 &\leq x \leq 2.0
\end{align*}
\]

![Graph showing the function \( f(x) = x \sin(\pi x) + 1 \) over the interval \(-1.0 \leq x \leq 2.0\).]

1.1.4 Major Advantages

- There are **three major advantages** when applying GA to optimization problems:
  - **Adaptability:**
    - GA does not have much mathematical requirements about the optimization problems.
  - **Robustness:**
    - The use of evolution operators makes GA very effective in performing global search (in probability).
    - Most of conventional heuristics usually perform local search.
  - **Flexibility:**
    - GA provides us a great flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem.

Chapter 1 Multiobjective Genetic Algorithms

1.1 Introduction

1.2 Implementation of Genetic Algorithms

- 1.2.1 GA Vocabulary
- 1.2.2 Encoding Issue
- 1.2.3 Fitness Evaluation
- 1.2.4 Genetic Operators
- 1.2.5 Handling Constraints

1.3 Hybrid Genetic Algorithms

1.4 Multiobjective Genetic Algorithms

1.2.1 GA Vocabulary

- GA is rooted in both **natural genetics** and **computer science**.
  - The terminologies used in GA literatures are mixture of the **natural** and the **artificial**.

<table>
<thead>
<tr>
<th>Table 1.1 Explanation of GA terms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Genetic Algorithms</strong></td>
</tr>
<tr>
<td>chromosome (string, individual)</td>
</tr>
<tr>
<td>genes (bits)</td>
</tr>
<tr>
<td>locus</td>
</tr>
<tr>
<td>alleles</td>
</tr>
<tr>
<td>phenotype</td>
</tr>
<tr>
<td>genotype</td>
</tr>
</tbody>
</table>

![Diagram showing the relationship between ID, chromosome, and genes.]
1.2.2 Encoding Issue

- **How to encode a solution** of the problem into a chromosome is a key issue for genetic algorithms.
  - In Holland’s work, encoding is carried out using binary strings.
  - For many GA applications, especially for the problems from industrial engineering world, the simple GA was difficult to apply directly as the binary string is not a natural coding.
- During last ten years, various non-string encoding techniques have been created for particular problems. For example:
  - The real number coding for constrained optimization problems
  - The integer coding for combinatorial optimization problems.
- Choosing an appropriate representation of candidate solutions to the problem at hand is the foundation for applying genetic algorithms to solve real world problems, which conditions all the subsequent steps of genetic algorithms.
- For any application case, it is necessary to analyse carefully to result in an appropriate representation of solutions together with meaningful and problem-specific genetic operators.

---

1.2.2 Encoding Issue

- **Basic features of genetic algorithms** is that they work on coding space and solution space alternatively:
  - Genetic operations work on coding space (chromosomes).
  - While evaluation and selection work on solution space.
  - Natural selection is the link between chromosomes and the performance of their decoded solutions.

![Diagram of Solution space (phenotype space) vs Coding space (genotype space)](image)

---

1.2.2 Encoding Issue

- According to **what kind of symbol is used**:
  - Binary encoding
  - Real number encoding
  - Integer/literal permutation encoding
  - A general data structure encoding

- According to **the structure of encodings**:
  - One-dimensional encoding
  - Multi-dimensional encoding

- According to **the length of chromosome**:
  - Fixed-length encoding
  - Variable length encoding

- According to **what kind of contents is encoded**:
  - Solution only
  - Solution + parameters

---

1.2.2 Encoding Issue

- **For nonstring coding approach**, there are **three critical issues** emerged concerning with the encoding and decoding between chromosomes and solutions (or the mapping between phenotype and genotype):
  - The **feasibility** of a chromosome
    - The feasibility refers to the phenomenon that whether or not a solution decoded from a chromosome lies in the feasible region of a given problem.
  - The **legality** of a chromosome
    - The legality refers to the phenomenon that whether or not a chromosome represents a solution to a given problem.
  - The **uniqueness** of mapping

---

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  65

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  66

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  67

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  68
1.2.2 Encoding Issue

- Feasibility and Legality is shown as following:

Solution space

Encoding space

- The **infeasibility of chromosomes** originates from the nature of the constrained optimization problem.
  - Whatever methods, conventional ones or genetic algorithms, must handle the constraints.
  - For many optimization problems, the **feasible region** can be represented as a system of equalities or inequalities (linear or nonlinear).
  - For such cases, **many efficient penalty methods** have been proposed to handle infeasible chromosomes.
  - In constrained optimization problems, the optimum typically occurs at the boundary between feasible and infeasible areas.
  - The **penalty approach** will force genetic search to approach to optimum from both side of feasible and infeasible regions.

- The **illegality of chromosomes** originates from the nature of encoding techniques.
  - For many combinatorial optimization problems, **problem-specific encodings** are used and such encodings usually *yield to illegal offspring by a simple one-cut point crossover operation*.
  - Because an illegal chromosome can not be decoded to a solution, it means that such chromosome can not be evaluated, *repairing techniques* are usually adopted to convert an illegal chromosome to a legal one.
  - For example, the well-known **partial-mapped crossover (PMX) operator** is essentially a kind of two-cut point crossover for permutation representation together with a repairing procedure to resolve the illegitimacy caused by the simple two-cut point crossover.
  - Orvosh and Davis have shown many combinatorial optimization problems using GA.
  - It is relatively easy to repair an infeasible or illegal chromosome and the repair strategy did indeed surpass other strategies such as rejecting strategy or penalizing strategy.
1.2.2 Encoding Issue

- The mapping from chromosomes to solutions (decoding) may belong to one of the following three cases:
  - 1-to-1 mapping
  - n-to-1 mapping
  - 1-to-n mapping

![Diagram showing solution space and encoding space with mappings]

1.2.2 Encoding Issue

- Properties of Encodings (cont.)
  - Property 6 (*Uniqueness*):
    - The mapping from chromosomes to solutions (decoding) may belong to one of the following three cases (Fig. 1.4): 1-to-1 mapping, n-to-1 mapping and 1-to-n mapping. The 1-to-1 mapping is the best one among three cases and 1-to-n mapping is the most undesired one.
  - Property 7 (*Heritability*):
    - Offspring of simple crossover (i.e., one-cut point crossover) should correspond to solutions which combine the basic feature of their parents.
  - Property 8 (*Locality*):
    - A small change in chromosome should imply a small change in its corresponding solution.

1.2.2 Encoding Issue

- Properties of Encodings
  - Property 1 (*Space*):
    - Chromosomes should not require extravagant amounts of memory.
  - Property 2 (*Time*):
    - The time complexity of executing evaluation, recombination and mutation on chromosomes should not be in a higher order.
  - Property 3 (*Feasibility*):
    - A chromosome corresponds to a feasible solution.
  - Property 4 (*Legality*):
    - Any permutation of a chromosome corresponds to a solution.
  - Property 5 (*Completeness*):
    - Any solution has a corresponding chromosome.

1.2.3 Fitness Evaluation

- Fitness Evaluation
  - Fitness evaluation is to check the solution value of the objective function.
  - Subjected to the problem constraints.
  - In general, the objective function provides the mechanism evaluating each individual.

- When evaluating the fitness function of some chromosome:
  - We have to design a *decoding procedure* depending on the chromosome.
1.2.4 Genetic Operators

- Genetic operators are used to alter the genetic composition of chromosomes during representation.
- There are an evolutionary operator:
  - Selection
    - Directing a GA search toward promising region in the search space.
- There are several common genetic operators:
  - Crossover
    - Operating on two chromosomes at a time and generating offspring by combining both chromosomes’ features.
  - Mutation
    - Producing spontaneous random changes in various chromosomes.
  - Immigration
    - Replace the worst members of the population with the random generated members.

a. Conventional Operators

- One-cut Point Crossover:
  
  ```
  parents  \( x = [x_1, x_2, ..., x_i, ..., x_{i-1}, x_{i+1}, ..., x_n] \)  
  \( y = [y_1, y_2, ..., y_i, y_{i+1}, y_{i+2}, ..., y_n] \)  
  
  offspring  \( x' = [x_1, x_2, ..., x_i, ..., x_{i-1}, y_{i+1}, y_{i+2}, ..., y_n] \)  
  \( y' = [y_1, y_2, ..., y_i, x_{i+1}, x_{i+2}, ..., x_n] \)  
  ```

- Random Mutation (Boundary Mutation):
  
  ```
  parent  \( x = [x_1, x_2, ..., x_i, ..., x_{i-1}, x_{i+1}, ..., x_n] \)  
  
  offspring  \( x' = [x_1, x_2, ..., x_i, ..., x_{i-1}, x_{i+1}, ..., x_n] \)  
  ```

b. Arithmetical Operators

- Crossover
  - Suppose that these are two parents \( x_1 \) and \( x_2 \), the offspring can be obtained by \( \lambda_1 x_1 + \lambda_2 x_2 \) with different multipliers \( \lambda_1 \) and \( \lambda_2 \).
  
  \[
  x_1' = \lambda_1 x_1 + \lambda_2 x_2 \\
  x_2' = \lambda_1 x_2 + \lambda_2 x_1 
  \]

  - Convex Crossover
    - If \( \lambda_1 + \lambda_2 = 1 \), \( \lambda_1 > 0, \lambda_2 > 0 \)
  - Affine Crossover
    - If \( \lambda_1 + \lambda_2 = 1 \)
  - Linear Crossover
    - If \( \lambda_1 + \lambda_2 = 2 \), \( \lambda_1 > 0, \lambda_2 > 0 \)
b. Arithmetical Operators

- Nonuniform Mutation (Dynamic Mutation)
  - For a given parent \( x \), if the element \( x_k \) of it is selected for mutation, the resulting offspring is \( x' = [x_1, \ldots, x'_k, \ldots, x_n] \), where \( x'_k \) is randomly selected from two possible choice:
    \[
    x'_k = x_k + \Delta(t, x_k - x') \quad \text{or} \quad x'_k = x_k - \Delta(t, x_k - x')
    \]
  - where \( x_k' \) and \( x_k'' \) are the upper and lower bounds for \( x_k \).
  - The function \( \Delta(t, y) \) returns a value in the range \([0, y]\) such that the value of \( \Delta(t, y) \) approaches to \( 0 \) as \( t \) increases (\( t \) is the generation number):
    \[
    \Delta(t, y) = y \cdot r \cdot \left( 1 - \frac{t}{T} \right)^b
    \]
  - where \( r \) is a random number from \([0, 1]\), \( T \) is the maximal generation number, and \( b \) is a parameter determining the degree of nonuniformity.

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  81

c. Direction-based Operators

- This operation use the values of objective function in determining the direction of genetic search:
- Direction-based crossover
  - Generate a single offspring \( x' \) from two parents \( x_1 \) and \( x_2 \) according to the following rules:
    \[
    x' = r \cdot (x_2 - x_1) + x_2
    \]
    where \( 0 < r \leq 1 \)
- Directional mutation
  - The offspring after mutation would be:
    \[
    x' = x + r \cdot d
    \]
    where \( d = \frac{f(x_1, \ldots, x_n + \Delta x_1, \ldots, x_n) - f(x_1, \ldots, x_n)}{\Delta x_i} \)
    \( r \) is a random nonnegative real number

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  82

d. Stochastic Operators

- Unimodal Normal Distribution Crossover (UNDX)
  - The UNDX generates two children from a region of normal distribution defined by three parents.
  - In one dimension defined by two parents \( x_1 \) and \( x_2 \), the standard deviation of the normal distribution is proportional to the distance between parents \( x_1 \) and \( x_2 \).
  - In the other dimension orthogonal to the first one, the standard deviation of the normal distribution is proportional to the distance of the third parent \( x_3 \) from the line.
  - The distance is also divided by \( \sqrt{n} \) in order to reduce the influence of the third parent.

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  83

- Unimodal Normal Distribution Crossover (UNDX)
  - Assume
    \[
    x_1, x_2: \text{the parents vectors} \quad C_1, C_2: \text{the child vectors} \quad n: \text{the number of variables} \quad d_1: \text{the distance between parents } x_1 \text{ and } x_2 \quad d_2: \text{the distance of parents } x_3 \text{ from the axis connecting parents } x_1 \text{ and } x_2 \quad z_1: \text{a random number with normal distribution } N(0, \sigma_1^2) \quad z_k: \text{a random number with normal distribution } N(0, \sigma_k^2) \quad k = 2, 3, \ldots, n \quad \sigma_1 = \alpha d_1, \quad \sigma_k = \beta d_2 / \sqrt{n} \quad e_i = (x_2 - x_1)/x_2 - x_1 \quad e_i \perp e_j, i, j = 1, 2, \ldots, n, i \neq j
    \]
  - The children are generated as follows:
    \[
    C_1 = m + z_1 e_1 + \sum_{k=2}^{n} z_k e_k \quad C_2 = m - z_1 e_1 - \sum_{k=2}^{n} z_k e_k
    \]
    \( m = (x_1 + x_2)/2 \)
    \( z_1 \sim N(0, \sigma_1^2), \quad z_k \sim N(0, \sigma_k^2) \quad k = 2, 3, \ldots, n \)
    \( \sigma_1 = \alpha d_1, \quad \sigma_k = \beta d_2 / \sqrt{n} \)

Soft Computing Lab.  WASEDA UNIVERSITY, IPS  84
d. Stochastic Operators

Box 1

- **Gaussian Mutation**
  
  An chromosome in evolution strategies consists of two components \((x, \sigma)\), where the first vector \(x\) represents a point in the search space, the second vector \(\sigma\) represents standard deviation.
  
  An offspring \((x', \sigma')\) is generated as follows:
  
  \[
  \sigma' = \sigma \cdot e^{N(0, \Delta \sigma)}
  \]
  
  \[
  x' = x + N(0, \Delta \sigma')
  \]
  
  where \(N(0, \Delta \sigma)\) is a vector of independent random Gaussian numbers with a mean of zero and standard deviation \(\sigma\).

1.2.4 Genetic Operators - Selection

- **The principle** behind genetic algorithms is essentially Darwinian natural selection.
- **Selection** provides the driving force in a genetic algorithm and the selection pressure is a critical in it.
  - Too much, the search will terminate prematurely.
  - Too little, progress will be slower than necessary.
  - **Low selection pressure** is indicated at the start to the GA search in favor of a wide exploration of the search space.
  - **High selection pressure** is recommended at the end in order to exploit the most promising regions of the search space.
- The selection directs GA search towards promising regions in the search space.
- During last few years, many selection methods have been proposed, examined, and compared.

Box 2

1.2.4 Genetic Operators - Selection

- **Sampling Space**
  - In Holland’s original GA, parents are replaced by their offspring soon after they give birth.
  - This is called as generational replacement.
  - Because genetic operations are blind in nature, offspring may be worse than their parents.
  - To overcome this problem, several replacement strategies have been examined.
  - Holland suggested that each offspring replaces a randomly chosen chromosome of the current population as it was born.
  - De Jong proposed a crowding strategy.
  - In the crowding model, when an offspring was born, one parent was selected to die. The dying parent was chosen as that parent was most closely resembled the new offspring using a simple bit-by-bit similarity count to measure resemblance.

- **Note that in Holland's works, selection refers to choosing parents for recombination and new population was formed by replacing parents with their offspring. They called it as reproductive plan.**
- Since Grefenstette and Baker's work, selection is used to form next generation usually with a probabilistic mechanism.
  - Michalewicz gave a detail description on simple genetic algorithms where offspring replaced their parents soon after they were born at each generation and next generation was formed by roulette wheel selection (Michalewicz, 1994).
1.2.4 Genetic Operators - Selection

- **Stochastic Sampling**
  - The **selection phase** determines the actual number of copies that each chromosome will receive based on its survival probability.
  - The **selection phase** is consist of two parts:
    - Determine the chromosome’s expected value.
    - Convert the expected values to the number of offspring.
  - A **chromosome’s expected value** is a real number indicating the average number of offspring that a chromosome should receive. The sampling procedure is used to convert the real expected value to the number of offspring.
    - Roulette wheel selection
    - Stochastic universal sampling

- **Deterministic Sampling**
  - Deterministic procedures which select the best chromosomes from parents and offspring.
    - $(\mu+\lambda)$-selection
    - $(\mu, \lambda)$-selection
    - Truncation selection
    - Block selection
    - Elitist selection
    - The generational replacement
    - Steady-state reproduction

1.2.4 Genetic Operators - Selection

- **Mixed Sampling**
  - Contains both random and deterministic features simultaneously.
    - Tournament selection
    - Binary tournament selection
    - Stochastic tournament selection
    - Remainder stochastic sampling

- **Regular Sampling Space**
  - Containing all offspring but just part of parents

![Selection based on regular sampling space](image)

- **Population**
  - $P_1$
  - $P_2$
  - $P_3$
  - $P_4$

- **Crossover**
  - $P_1'$
  - $P_2'$

- **Mutation**
  - $P_3'$

- **Replacement**
  - $P_4'$

- **New population**
  - $P_1''$
  - $P_2''$
  - $P_3''$
  - $P_4''$
### 1.2.4 Genetic Operators - Selection

#### Enlarged sampling space
- containing all parents and offspring

#### Selection based on enlarged sampling space

#### 1.2.4 Genetic Operators - Selection

**Scaling Mechanisms**

- Linear scaling
  \[ f_k' = a \times f_k + b \]

- Power low scaling
  \[ f_k' = f_k^\alpha \]

- Normalizing scaling
  \[ f_k' = \frac{f_k - f_{\min} + \gamma}{f_{\max} - f_{\min} + \gamma}, \quad 0 < \gamma < 1 \text{ (for maximization problem)} \]

- Boltzmann scaling
  \[ f_k' = e^{f_k/T} \]

**Selection Probability**

- **Fitness scaling** has a twofold intention:
  - To maintain a reasonable differential between relative fitness ratings of chromosomes.
  - To prevent a too-rapid takeover by some super chromosomes in order to meet the requirement to limit competition early on, but to stimulate it later.

- Suppose that the raw fitness \( f_k \) (e.g. objective function value) for the \( k \)-th chromosomes, the scaled fitness \( f_k' \) is:
  \[ f_k' = g(f_k) \]

- Function \( g(\cdot) \) may take different form to yield different scaling methods.

### 1.2.5 Handling Constraints

- Several techniques have been proposed to handle constraints with Genetic Algorithms (GAs):

  - **Rejecting Strategy**
    - Rejecting strategy discards all infeasible chromosomes created throughout the evolutionary process.

  - **Repairing Strategy**
    - Repairing a chromosome involves taking an infeasible chromosome and generating a feasible one through some repairing procedure.
    - Repairing strategy depends on the existence of a deterministic repair procedure to convert an infeasible offspring into a feasible one.

  - **Modifying Genetic Operator Strategy**
    - One reasonable approach for dealing with the issue of feasibility is to invent problem-specific representation and specialized genetic operators to maintain the feasibility of chromosomes.

  - **Penalty Strategy**
    - These strategies above have the advantage that they never generate infeasible solutions but have the disadvantage that they consider no points outside the feasible regions.
1.2.5 Handling Constraints

**Penalty Methods**


  - Penalty techniques transform a constrained problem into an unconstrained problem by penalizing infeasible solutions, in which a penalty term is added to the objective function for any violation of the constraints.
  - The basic idea of the penalty technique is borrowed from conventional optimization.

<table>
<thead>
<tr>
<th>Conventional Optimization</th>
<th>Genetic Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ow to choose a proper value of the penalty term so as to get a fast convergence and avoid premature termination.</td>
<td>ow to determine the penalty term so as to strike a proper balance between the information preservation and the selection pressure for the infeasible solutions and avoid both under-penalty and over-penalty.</td>
</tr>
<tr>
<td>ow to method to generate a sequence of feasible points whose limit is an optimal solution to the original problem.</td>
<td>eep some infeasible solution in population so as to force genetic search toward optimal solution from both side of feasible and infeasible region.</td>
</tr>
</tbody>
</table>

**Evaluation Function with Penalty Term**

- Two possible methods to construct the evaluation function with penalty term.
- One method is to take the addition form expressed as follows:

\[
\text{eval} \cdot f(x) \cdot p(x)
\]

  - where \( x \) represents a chromosome, \( f(x) \) the objective function of problem and \( p(x) \) the penalty term. For maximization problems we usually require that

\[
\begin{align*}
    p(x) & \quad \text{if } x \text{ is feasible} \\
    p(x) & \quad \text{otherwise}
\end{align*}
\]

  - Let \( p(x)_{\text{max}} \) and \( f(x)_{\text{min}} \) be the maximum of \( p(x) \) and minimum of \( f(x) \) among infeasible solutions in current population respectively. We also require that

\[
\begin{align*}
    p(x)_{\text{max}} & \leq f(x)_{\text{min}}
\end{align*}
\]

  - to avoid negative fitness value.

**The second method** is to take the multiplication form expressed as follows:

\[
\text{eval} \cdot f(x) \cdot p(x)
\]

- In this case for maximization problems we require that

\[
\begin{align*}
    p(x) & \quad \text{if } x \text{ is feasible} \\
    p(x) & \quad \text{otherwise}
\end{align*}
\]

and for minimization problems we require that

\[
\begin{align*}
    p(x) & \quad \text{if } x \text{ is feasible} \\
    1 & \quad \text{otherwise}
\end{align*}
\]

- Note that for the minimization problems the fitter chromosome has the lower value of \( \text{eval} \cdot f(x) \).

**Several techniques for handling infeasibility** have been proposed in the area of genetic algorithms. In general we can classify them into two classes:

- constant penalty approach is known to be less effective for complex problems and most recent research works put the attention on the variable penalty.

- variable penalty approach contains two components:
  - variable penalty ratio: it can be adjusted according to the degree of violation of constraints and the iteration number of genetic algorithms.
  - penalty amount for the violation of constraints.

- Essentially the penalty is a function of the distance from the feasible area. This can be given in the following three possible ways:

  - absolute distance of a single infeasible solution.
  - relative distance of all infeasible solution in the current population.
  - the adaptive penalty term.

- The penalty approaches can be further distinguished by the dependence of the problem or the existence of parameter.
1.3 Hybrid Genetic Algorithms

- There are two kinds of hybrid GAs.
  - Adaptation to Problems
    - Advocates modifying some components of genetic algorithms such as representation, crossover, mutation, and selection to choose an appropriate form of the algorithm to meet the nature of a given problem.
  - Adaptation to Evolutionary processes
    - Suggests a way to tune the parameters of the changing configurations of genetic algorithms while solving the problem.
    - Divided into five classes: hybrid GA: hGA:
      - Adaptive parameter settings
      - Adaptive genetic operators
      - Adaptive selection
      - Adaptive representation
      - Adaptive fitness function

1.3.1 Genetic Local Search

- Applying a local search technique to GA loop
  - hill climbing method

- Weakness of conventional GA approach to the problem of combinatorial nature of design variables
- hybrid Genetic Algorithm hGA
  - Conventional GAs have not any scheme for locating local search area resulting from GA loop.
  - Improving
  - Applying a local search technique to GA loop.
  - The identification of the correct settings of genetic parameters such as population size probabilities of crossover and mutation operators is not an easy task.
  - Improving
  - Parameter control approach of GA

Fig. Hill climbing method
1.3.1 Genetic Local Search

- Applying a local search technique to GA loop
  - Iterative hill climbing method

- The general structure of hybrid GA by Local Search

![Fig. Iterative hill climbing method](image)

1.3.2 Parameter Adaptation

- The general procedure of hybrid GA by Local Search method

```
procedure: hybrid GA
input: problem data, GA parameters
output: the best solution
begin
  t ←
  initial P  by encoding routine
  evaluate P  by decoding routine
  while (not terminating condition) do
    create C  from P  by crossover routine
    create C  from P  by mutation routine
    climb C  by local search routine
    evaluate C  by decoding routine
    select P  from P + C  by selection routine
    t ← t+1
  end
  output the best solution
end
```

1.3.2 Parameter Adaptation

- Parameter Control Approach of GA
  - The behaviors of GA are characterized by the balance between exploitation and exploration in the search space.

![GA diagram](image)
1.3.2 Parameter Adaptation

- **Parameter Control Approach of GA**
  - The **balance** between **exploitation** and **exploration** in the search **space** is strongly affected by the **effectiveness of GA**.
  - Usually, fixed parameters are used in most applications of GA and are determined with a set-and-test approach.
  - Since GA is an intrinsically dynamic and adaptive process, the use of constant parameters is thus in contrast to the general evolutionary spirit.
  - Therefore, it is a natural idea to try to modify the values of strategy parameters during the run of the genetic algorithm by using the following three ways.
  - **Deterministic**: using some deterministic rule.
  - **Adaptive**: taking feedback information from the current state of search.
  - **Self-adaptive**: employing some self-adaptive mechanism.

---

**Srinivas and Patnaik’s Approach**

- **Heuristic Updating Strategy**
  - This scheme is to control $P_c$ and $P_m$ with respect to the fitness of offspring at each generation.
  - **procedure**: regulation of $P_c$ and $P_m$ using the fitness of offspring
  - **input**: GA parameters $P_c$, $P_m$, $t-1$
  - **output**: $P_c$, $P_m$, $t$

```plaintext
begin
  if $f_{off} > 0.1$ then
    $p_c$ = $p_c$ $(t-1)$; $p_m$ = $p_m$ $(t-1)$;
  else
    if $f_{off} < 0.1$ then
      $p_c$ = $p_c$ $(t-1)$; $p_m$ = $p_m$ $(t-1)$;
    else
      $p_c$ = $p_c$ $(t-1)$; $p_m$ = $p_m$ $(t-1)$;

output $P_c$, $P_m$, $t$
end
```

---

**IEEE-SMC 1 4**

- Heuristic Updating Strategy
  - This scheme is to control $P_c$ and $P_m$ using various fitness at each generation.
  - $$p_c = \begin{cases} k_1 (f_m - f_c), & f_m > f_c \vspace{0.5em} \\ f_m - f_c, & f_m < f_c \end{cases}$$
  - $$p_m = \begin{cases} k_1 (f_m - f_c), & f_m > f_c \vspace{0.5em} \\ f_m - f_c, & f_m < f_c \end{cases}$$

where
- $f_m$: maximum fitness value at each generation.
- $f_c$: average fitness value at each generation.
- $f_m$: the larger of the fitness values of the chromosomes to be crossed.
- $f_c$: the fitness value of the $i$th chromosome to which the mutation with a rate $P_m$ is applied.

---

**Coordination Strategy between the FLC and GA**

- **Parameter Control Approach using Fuzzy Logic Controller (FLC)**
  - **Heuristic Updating Strategy**
  - This scheme is to control $P_c$ and $P_m$ using various fitness at each generation.
  - **procedure**: regulation of $P_c$ and $P_m$ using the fitness of offspring
  - **input**: GA parameters $P_c$, $P_m$, $t-1$
  - **output**: $P_c$, $P_m$, $t$

---

**Figure: Coordinated strategy between the FLC and GA**

- Crossover LC
- Mutation LC
- GA

---

**Soft Computing Lab. WASEDA NIVE SIT IPS**

111
1.3.2 Parameter Adaptation

- Basic Concept
  
  A heuristic updating strategy for the crossover and mutation rates is to consider
  changes of average fitness in the GA population of two continuous generations.
  For example in minimization problem we can set the change of the average
  fitness at generation \( t \) to be:

  \[
  \Delta f_{\text{avg}}(t) = (\frac{1}{\text{parSize}} \sum_{i=1}^{\text{parSize}} f(t) - \frac{1}{\text{offSize}} \sum_{i=1}^{\text{offSize}} f_i(t))
  \]

  where
  - parSize: population size satisfying the constraints
  - offSize: offspring size satisfying the constraints

- Defuzzification table for control actions
  
  For simplicity the defuzzification table for determining the action of
  crossover FLC was set up. It is formulated as follows: Song et al. 1 7.

  Table: Defuzzification table for control action of crossover

<table>
<thead>
<tr>
<th>( y )</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

1.3.2 Parameter Adaptation

- Procedure: regulation of \( p_C \) and \( p_M \) using the average fitness
  
  Input: GA parameters \( p_C, p_M, t-1, \Delta f_{\text{avg}}(t-1), \Delta f_{\text{avg}}(t) \).
  
  Output: \( p_C, p_M, t \)

  Begin
  
  if \( \epsilon \leq \Delta f_{\text{avg}}(t-1) \leq \gamma \) and \( \epsilon \leq \Delta f_{\text{avg}}(t) \leq \gamma \)
  then increase \( p_C \) and \( p_M \) for next generation

  if \( -\gamma \leq \Delta f_{\text{avg}}(t-1) \leq -\epsilon \) and \( -\gamma \leq \Delta f_{\text{avg}}(t) \leq -\epsilon \)
  then decrease \( p_C \) and \( p_M \) for next generation

  if \( -\epsilon \leq \Delta f_{\text{avg}}(t-1) \leq \epsilon \) and \( -\epsilon \leq \Delta f_{\text{avg}}(t) \leq \epsilon \)
  then rapidly increase \( p_C \) and \( p_M \) for next generation

  Output \( p_C, p_M, t \)

  End

- Where
  - \( \epsilon \): a given real number in the proximity of zero (in our case, \( \epsilon = 0.1 \).
  - \( \gamma \): a given maximum values of a fuzzy membership function (in our case, \( \gamma = 1.0 \).
  - \( -\gamma \): a given minimum values of a fuzzy membership function (in our case, \( -\gamma = -1.0 \).

1.3.2 Parameter Adaptation

- Oumousis and Atsaras 2 proposed Saw-tooth GA stGA
  
  Utilises a variable population size following the periodic scheme
  presented in Fig. in the form of a saw-tooth function.

- Oumousis V. C. P. Atsaras A Saw-tooth Genetic Algorithm
  
  Combining the Effects of Variable Population Size and
  Initialisation to Enhance Performance IEEE Trans. on
  Evolutionary Computation vol.1 no.1 pp.1-2 2.

- Diagram: Saw-tooth Genetic Algorithm
1.3.2 Parameter Adaptation

  - There are three genetic operators are comprised:
    - Crossover (same with a simple GA)
    - Immigration (a kind of random search)
    - Heuristic mutation (a kind of heuristic search)
  - Auto-tuning strategy turns the probabilities of above three genetic operators.
    - $p_C$: crossover probability
    - $p_i$: immigration probability
    - $p_M$: heuristic mutation probability
    - $p_C + p_i + p_M = 1$

- Auto-tuning Strategy by FLC
  - auto-tuning for exploration and exploitation $T[p_M \land (p_C \lor p_i)]$
  - auto-tuning for genetic exploration and random exploration $T[p_C \land p_i]$

- Immigration
  - Moed et. al. (1990) proposed an immigration operator
  - For certain types of functions, allows increased exploration while maintaining nearly the same level of exploitation for the given population size.
  - Immigration operator
    - step 1: The algorithm is modified to include immigration, with each generation generated.
    - step 2: Evaluate $popSize \cdot p_i$ random members ($p_i$, called the immigration rate).
    - step 3: Replace the $popSize \cdot p_i$ worst members of the population with the $popSize \cdot p_i$ random members.

\[ \Delta f_{avg}(t) = \frac{(f_{parSize}(t) - f_{offSize}(t))}{f_{parSize}(t)} \]
\[ = \frac{\sum_{j=1}^{parSize} f_j(t) - \sum_{j=1}^{offSize} f_j(t)}{parSize} \frac{\sum_{j=1}^{offSize} f_j(t)}{parSize} \]

where $parSize$ and $offSize$ are the parent and offspring population sizes satisfying constraints, respectively.

- We define $p_C$ and $p_M$ by using correlation coefficient of the individuals in the current generation.
  - $p_M = \text{regulation 1}(\Delta f_{avg}(t), i = 1, 2, \ldots, u)$
  - $p_C = \text{regulation 2}(\Delta f_{avg}(t), i = 1, 2, \ldots, u)$
  - $p_i = 1 - (p_M + p_C)$
1.3.2 Parameter Adaptation

**Pseudo code of Regulation of \( p_M \)**

```plaintext
procedure : Regulation 1 for \( p_M \)
begin
  if \( \Delta f \leq \alpha \) then
    \( p_M = 0.2; \)
  if \( \alpha \leq \Delta f \leq 2^\alpha - \alpha \) then
    \( p_M = 0.2 + \mu_M \cdot (\Delta f - \alpha); \)
  if \( \Delta f \geq 2^\alpha - \alpha \) then
    \( p_M = 0.8; \)
end
```

Fig.: Membership function of \( p_M \)

\[
\Delta f = \sum_{i=1}^{n} 2^{i-1} \cdot \lambda(\Delta f_{avg}(t-i)), \ t \geq u \\
\mu_M = (0.8 - 0.2)/(2^\alpha - 2\alpha) \\
\lambda(x) = \begin{cases} 
  1, & \text{if } x \geq 0 \\
  0, & \text{otherwise} 
\end{cases} \\
0 \leq \alpha \leq 2^\alpha/4
\]

121

**Pseudo code of Regulation of \( p_C \)**

```plaintext
procedure : Regulation 2 for \( p_C \)
begin
  if \( \Delta f \leq \alpha \) then
    \( p_C = 0.8 \cdot (1 - p_M); \)
  if \( \alpha \leq \Delta f \leq 2^\alpha - \alpha \) then
    \( p_C = 0.8 \cdot (1 - p_M) - 0.1 \cdot \mu_C \cdot (\Delta f - \alpha); \)
  if \( \Delta f \geq 2^\alpha - \alpha \) then
    \( p_C = 0.1; \)
end
```

Fig.: Membership function of \( p_C \)

\[
\Delta f = \sum_{i=1}^{n} 2^{i-1} \cdot \lambda(\Delta f_{avg}(t-i)), \ t \geq u \\
\mu_C = (0.8 \cdot (1 - p_M) - 0.1)/(2^\alpha - 2\alpha) \\
\lambda(x) = \begin{cases} 
  1, & \text{if } x \geq 0 \\
  0, & \text{otherwise} 
\end{cases} \\
0 \leq \alpha \leq 2^\alpha/4
\]

122

1.3.2 Parameter Adaptation

**Parameter tuning process of genetic operators by auto-tuning strategy :**

![Fig. 4: Self control of genetic operators by proposed approach](image)

123

**Fig.: Typical Convergence Plot of atGA, fc-aGA and simple GA**

Chapter 1 Multiobjective Genetic Algorithms

1.1 Introduction
1.2 Implementation of Genetic Algorithms
1.3 Hybrid Genetic Algorithms

1.4 Multiobjective Genetic Algorithms
1.4.1 Basic Concepts of Multiobjective Optimizations
1.4.2 Features and Implementation of Multiobjective GA
1.4.3 Fitness Assignment Mechanism
1.4.4 Performance Measures

1.4.1 Basic Concepts of Multiobjective Opti.

- Optimization deals with the problem of seeking solutions over a set of possible choices to optimize certain criteria.
- Multiobjective Optimization Problems (MOP) arise in the design, modeling, and planning of many complex real systems.
- Almost every important real-world decision making problem involves multiple and conflicting objectives.
  - Need to be tackled while respecting various constraints
  - Leading to overwhelming problem complexity.
- Genetic Algorithms have received considerable attention as a novel approach to multiobjective optimization problem.

- Multiobjective optimization problem with $q$ objective functions and $m$ nonlinear constraints can be represented:
  \[
  \begin{align*}
  \text{max} & \quad \{ z_i = f_i(x), \ z_2 = f_2(x), \ldots, \ z_q = f_q(x) \} \\
  \text{s.t.} & \quad g_i(x) \leq 0, \quad i = 1, 2, \ldots, m \\
  & \quad x \geq 0
  \end{align*}
  \]

- The feasible region in the decision space denoted by the set $S$, is as follows:
  \[ S = \{ x \in \mathbb{R}^n \mid g_i(x) \leq 0, \ i = 1, 2, \ldots, m, \ x \geq 0 \} \]

- The feasible region in the criterion space denoted by the set $Z$, is as follows:
  \[ Z = \{ z \in \mathbb{R}^q \mid z_1 = f_1(x), \ z_2 = f_2(x), \ldots, z_q = f_q(x), \ x \in S \} \]
1.4.1 Basic Concepts of Multiobjective Opti.

**Definition 1**: For a given point \( z^0 \in Z \), it is **nondominated** if and only if there does not exist another point \( z \in Z \) such that, for the maximization case,
- \( z_k \geq z^0_k \), for all \( k \)
- \( z_k > z^0_k \), for at least one \( k \)
where, \( z^0 \) is a **dominated point** in the criterion space \( Z \).

**Definition 2**: For a given point \( x^0 \in S \), it is **efficient** if and only if there does not exist another point \( x \in S \) such that, for the maximization case,
- \( f_i(x) \geq f_i(x^0) \), for all \( k \)
- \( f_i(x) > f_i(x^0) \), for at least one \( k \)
where, \( x^0 \) is **inefficient**.

---

**Example 1**: Two-objective (bicriteria) linear programming

\[
\begin{align*}
\text{max} & \quad z_1 = f_1(x) = -x_1 + 3x_2 \\
\text{max} & \quad z_2 = f_2(x) = 3x_1 + x_2 \\
\text{s. t.} & \quad g_1(x) = x_1 + 2x_2 - 2 \leq 0 \\
& \quad g_2(x) = 2x_1 + x_2 - 2 \leq 0 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

We can observe that both regions are convex and the extreme points of \( Z \) are the images of extreme points of \( S \).

---

**The concept of Pareto optimal solutions**

(maximization case)

- Pareto optimal solution
- dominated solution

---

**The extreme points in the feasible region \( S \) of the decision space are shown in Fig. 1.13:**

\[
\begin{align*}
\text{max} & \quad z_1 = f_1(x) = -x_1 + 3x_2 \\
\text{max} & \quad z_2 = f_2(x) = 3x_1 + x_2 \\
\text{s. t.} & \quad g_1(x) = x_1 + 2x_2 - 2 \leq 0 \\
& \quad g_2(x) = 2x_1 + x_2 - 2 \leq 0 \\
& \quad x_1, x_2 \geq 0 \\
\end{align*}
\]

\( S = \{ x \mid g_i(x) \leq 0, \forall i, x \geq 0 \} \)

Fig. 1.13 Feasible region and efficient solutions in decision space
1.4.1 Basic Concepts of Multiobjective Opti.

The corresponding extreme points in the feasible region Z of the criterion space are shown in Fig. 1.14:

\[ z^1 = [f_1(x^1), f_2(x^1)] = [0, 0] \]
\[ z^2 = [f_1(x^2), f_2(x^2)] = [-1, 3] \]
\[ z^3 = [f_1(x^3), f_2(x^3)] = [4/3, 8/3] \]
\[ z^4 = [f_1(x^4), f_2(x^4)] = [3, 1] \]

Pareto optimal solutions

<table>
<thead>
<tr>
<th>x = [x_1, x_2]</th>
<th>f(x) = [f_1(x), f_2(x)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 0]</td>
<td>[-1, 3]</td>
</tr>
<tr>
<td>[2/3, 2/3]</td>
<td>[4/3, 8/3]</td>
</tr>
<tr>
<td>[0, 1]</td>
<td>[3, 1]</td>
</tr>
</tbody>
</table>

Fig. 1.14 Feasible region and non-dominated solutions in criterion space

In real decision-making cases, we are usually asked to select one of the non-dominated solutions as a final solution, i.e., a best compromised solution to a given problem.

Efficient solutions:
- In a basic feature of solutions to multiobjective optimization problems there is a set of efficient solutions that cannot simply be compared with each other.

Best compromised solution:
- In real decision-making cases, we are usually asked to select one of the non-dominated solutions as a final solution, i.e., a best compromised solution to a given problem.

How to make a final choice from those alternative solutions, i.e., non-dominated solutions or Pareto optimal solutions essentially depends on one’s subjective preference.

A preference is usually represented by a binary relation.

For a given pair, say \( u \) and \( v \), one and only one of the following relation can occur:
- \( u \) is better than or preference to \( v \), denoted by \( u \succ v \).
- \( u \) is worse than or less preferred to \( v \), denote by \( u \prec v \).
- \( u \) is equivalent to or equally preferred to \( v \), denoted by \( u \sim v \).
- The preference relation between \( u \) and \( v \) is indefinite, denote by \( u ? v \).
Chapter 1 Multiobjective Genetic Algorithms

1.1 Introduction
1.2 Implementation of Genetic Algorithms
1.3 Hybrid Genetic Algorithms

1.4 Multiobjective Genetic Algorithms
  1.4.1 Basic Concepts of Multiobjective Optimizations
  1.4.2 Features and Implementation of Multiobjective GA
  1.4.3 Fitness Assignment Mechanism
  1.4.4 Performance Measures
  1.4.5 Applications of Multiobjective GA

1.4.2 Features and Implementation of moGA

- Procedure of Multiobjective Genetic Algorithm (moGA)

```
procedure: Multiobjective Genetic Algorithm
input: problem data, GA parameters
output: Pareto optimal solutions E
begin
  t ← 0;
  initialize P(t) by encoding routine;
  calculate objectives f_i(P), i=1, 2, ..., q;
  create Pareto E(P) by nondominated routine;
  evaluate eval(P) by fitness assignment routine;
  while (not terminating condition) do
    create C(t) from P(t) by crossover routine;
    create C(t) from P(t) by mutation routine;
    calculate objectives f_i(C), i=1, 2, ..., q;
    update Pareto E(P, C) by nondominated routine;
    evaluate eval(P, C) by fitness assignment routine;
    select P(i+1) from P(t) and C(t) by selection routine;
    t ← t + 1;
  end
output Pareto optimal solutions E(PC);
end
```
1.4.3 Fitness Assignment Mechanism

**Fitness Assignment Mechanism**

- One special issue arising in solving multiobjective optimization problems is how to determine the fitness values of chromosomes according to multiple objectives.
- Since the 1980s, several fitness assignment mechanisms have been proposed and applied in multiobjective optimization problems.
- Most fitness assignment mechanisms are just different approach and suitable to different cases of multiobjective optimization problems.

### Generation 1: Vector evaluation approach

**Vector Evaluated Genetic Algorithm (veGA: Schaffer, 1985)**

- **veGA** is the first notable work to solve MOP problems in which it uses a vector fitness measure to create the next generation.
- The selection step in each generation becomes a loop. Each time through the loop the appropriate fraction of the next generation, or subpopulation, is selected on the basis of each objective.
- The entire population is shuffled thoroughly to apply crossover and mutation operators. This is performed to achieve the mating of individuals of different subpopulations.

### Generation 2: Pareto ranking + Diversity

- multiobjective GA (moGA), Fonseca and Fleming 1993
- non-dominated sorting GA (nsGA), Deb 1995

### Generation 3: Weighted Sum + Elitist Preserve

- random weight GA (rwGA), Ishibuchi and Murata 1998
- adaptive weight GA (awGA), Gen and Cheng 2000
- strength Pareto Evolutionary Algorithm II (spEAII), Zitzler et al. 2002
- non-dominated sorting GA II (nsGAII), Deb 2002
- interactive adaptive-weight GA (i-awGA), Lin and Gen 2008

**Schematic of veGA Selection**

<table>
<thead>
<tr>
<th>chromosome</th>
<th>performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>popSize</td>
<td>q</td>
</tr>
</tbody>
</table>

- Select $n$ subgroups using each objective in turn
- Shuffle
- Apply genetic operators
- generation $t+1$: $\vdots$
Generation 1: Vector evaluation approach

- A simple two-objective problem with one variable is used by Schaffer to test the properties of veGA:
  \[
  \begin{align*}
  \min f_1(x) &= x^2 \\
  \min f_2(x) &= (x-2)^2 \\
  \text{s. t.} & \quad x \in \mathbb{R}
  \end{align*}
  \]

- Population in different generations obtained using veGA

Generation 2: Pareto ranking + Diversity

- Pareto Ranking Approach
  - Pareto ranking approach consists of assigning rank 1 to nondominated chromosomes and removing them from contention, then finding the nondominated chromosomes among the remaining ones, rank 2; and so on.
  - This approach assigns all nondominated solutions in identical fitness value in order to give them an equal probability of reproduction.
  - A Pareto ranking-based fitness assignment method was first suggested by Goldberg.


Generation 2: Pareto ranking + Diversity

- Goldberg’s method (1989)
  - This method is one of means of achieving equal reproductive potential for all Pareto individuals.

  - procedure: Goldberg’s method
    - input: chromosomes \( v_k, k=1, 2, \ldots, \text{popSize} \)
    - output: fitness \( \text{eval}(v_k) \)
    - step 1: rank 1 is given to the nondominated chromosomes
    - step 2: removing them from contention.
    - step 3: finding the next nondominated chromosomes, removing them from contention, rank 2 is assigned to them.
    - step 4: process continues until the entire population is ranked, and output fitness \( \text{eval}(v_k) \).
**Generation 2: Pareto ranking + Diversity**

- **Multiobjective Genetic Algorithm (moGA: Fonseca and Fleming, 1993)**
  - The method proceeds by sorting the population according to the ranks, and ties may be broken by random choice.

**procedure:** moGA method
**input:** chromosomes $v_k$, $k=1, 2, ..., popSize
**output:** fitness $eval(v_k)$

- step 1: rank 1 is given to the nondominated chromosomes
- step 2: removing them from contention.
- step 3: finding the next nondominated chromosomes, removing them from contention, rank equally to the number of its dominating individuals plus one.
- step 4: process continues until the entire population is ranked, and output fitness $eval(v_k)$.

![Fig. 1.18 Simple case with two objectives to be minimized using moGA](image)

**Generation 3: Weighted Sum + Elitist Preserve**

- **Non-dominated Sorting Genetic Algorithm (nsGA: Deb, 1995)**
  - The population is ranked based on Goldberg’s Pareto ranking method.

**procedure:** Non-dominated Sorting GA
**input:** chromosomes $v_k$, $k=1, 2, ..., popSize
**output:** fitness $eval(v_k)$

- step 1: The nondominated individuals are first identified and assigned a large dummy fitness value.
- step 2: To maintain diversity in the population, these individuals are shared with their dummy fitness values.
- step 3: After sharing, these nondominated individuals are ignored temporarily.
- step 4: The second nondominated front in the rest of the population is identified and assigned a dummy fitness value that is kept smaller than the minimum shared dummy fitness of the previous front.
- step 5: This process is continued until the entire population is classified into several fronts, and output fitness $eval(v_k)$.

- A stochastic remainder proportionate selection is used to reproduce a new generation.

**Random-weighted Genetic Algorithm (rwGA: Murata et al., 1998)**

- Murata, Ishibuchi & Tanaka (1998) proposed a Random-weight Genetic Algorithm (rwGA) to obtaining a variable search direction toward the Pareto frontier.


- Random-weight Genetic Algorithm gives the GAs a tendency to sample the area toward a fixed point in the criterion space.

- Random-weight Genetic Algorithm gives the GAs a tendency to demonstrate a variable search direction, therefore, the ability to sample the area uniformly over the entire frontier.
Generation 3: Weighted Sum + Elitist Preserve

- **Random-weighted Genetic Algorithm**

  - **Procedure of random-weighted Genetic Algorithm**

    ```plaintext
    procedure: rwGA
    input: the objective $f_i(v_i)$ of each chromosome $v_i$, $k=1,2,...,q$, $\forall i \in \text{popSize}$
    output: fitness value $\text{eval}(v_i)$, $\forall i \in \text{popSize}$
    
    begin
      $r_j \leftarrow \text{random}[0,1]$, $j=1,2,...,q$; // non-negative random number
      $w_i \leftarrow r_j / \sum_{j=1}^{q} r_j$, $k=1,2,...,q$;
      $\text{eval}(v_i) \leftarrow \sum_{i=1}^{q} w_i \left(f_i(v_i) - z_i^{\min}\right)$, $\forall i$;
    
    output $\text{eval}(v_i)$, $\forall i$;
    
    end
    ```

  - For a problem to maximize $q$ objective functions, weighted-sum objective is given as follows:
    \[ z = \sum_{i=1}^{q} w_i f_i(x) \]
    
  - Random-weight $w_i$ is calculated by the following equation:
    \[ w_i = \frac{r_i}{\sum_{j=1}^{q} r_j}, \quad k = 1, 2, ..., q \]
    where $r_j$ are non-negative random number between $[0, 1]$.

  - Before selecting a pair of parents for crossover operation, a new set of random weights is specified. The selection probability $p_i$ for individual $i$ is then defined by the following linear scaling function:
    \[ p_i = \frac{z_i - z_i^{\min}}{\sum_{j=1}^{q} \text{popSize} \left(z_j - z_j^{\min}\right)} \]
    where $z_i^{\min}$ is the worst fitness value in the current population.


  - Gen and Cheng (2000) proposed an **Adaptive Weight Genetic Algorithm (awGA)** which utilizes some useful information from the current population to readjust weights to obtain a search pressure toward a positive ideal point.


  - For the examined solutions at each generation, they define two extreme points (maximum: $z^*$, minimum: $z^\dagger$):
    \[ z^* = [z_1^{\max}, z_2^{\max}, ..., z_q^{\max}] \]
    \[ z^\dagger = [z_1^{\min}, z_2^{\min}, ..., z_q^{\min}] \]
    where $z_1^{\max}$ and $z_2^{\min}$ are the maximal and minimal values for the $k$th objective as defined by the following equations:
    \[ z_1^{\max} = \max \{ f_i(x) \mid x \in P \}, \quad k = 1, 2, ..., q \]
    \[ z_1^{\min} = \min \{ f_i(x) \mid x \in P \}, \quad k = 1, 2, ..., q \]
    $P$: set of solution candidates.
Generation 3: Weighted Sum + Elitist Preserve

The weighted-sum objective function for a given chromosome \( x \) is given by the following equation:

\[
z(x) = \sum_{k=1}^{q} f_k(x) - z_k^{min}
\]

where \( w_k \) is adaptive weight for objective \( k \):

\[
w_k = \frac{1}{z_k^{max} - z_k^{min}}, \quad k = 1, 2, \ldots, q
\]

The equation driven above is a hyperplane defined by the following extreme points in current solutions:

\[
\begin{bmatrix}
z_1^{max} & z_1^{min} & \cdots & z_k^{min} & \cdots & z_q^{min} \\
z_1^{min} & z_1^{max} & \cdots & z_k^{min} & \cdots & z_q^{min} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
z_1^{min} & z_1^{min} & \cdots & z_k^{max} & \cdots & z_q^{min} \\
z_1^{min} & z_1^{min} & \cdots & z_k^{min} & \cdots & z_q^{max}
\end{bmatrix}
\]

Example of a bi-criteria maximization problem:

\[
\begin{align*}
\text{max} \; \{ z_i = f_i(x), \; z_j = f_j(x) \} \\
\text{s.t.} \; g_i(x) \leq 0, \; i = 1, 2, \ldots, m
\end{align*}
\]

two extreme points are identified as:

\[
\begin{align*}
z_1^{max} &= \max \{ f_i(x_j), \; j = 1, 2, \ldots, \text{popSize} \} \\
z_2^{max} &= \max \{ f_j(x_j), \; j = 1, 2, \ldots, \text{popSize} \} \\
z_1^{min} &= \min \{ f_i(x_j), \; j = 1, 2, \ldots, \text{popSize} \} \\
z_2^{min} &= \min \{ f_j(x_j), \; j = 1, 2, \ldots, \text{popSize} \}
\end{align*}
\]

the adaptive weights are calculated as

\[
w_k = \frac{1}{z_k^{max} - z_k^{min}}
\]

Procedure of adaptive weighted Genetic Algorithm

```
procedure: awGA
input: the objective \( f_i \) of each chromosome \( v_i \), \( i = 1, 2, \ldots, q \), \( \forall i \in \text{popSize} \)
output: fitness value \( \text{eval}(v_i) \), \( \forall i \in \text{popSize} \)

begin

\[
\begin{align*}
\{ z_k^{max} \} &= \max f_i(v_i), \; k = 1, 2, \ldots, q; \quad \text{maximum extreme point } z^* \\
\{ z_k^{min} \} &= \min f_i(v_i), \; k = 1, 2, \ldots, q; \quad \text{minimum extreme point } z^*
\end{align*}
\]

\[
w_k = \frac{1}{z_k^{max} - z_k^{min}}, \; k = 1, 2, \ldots, q
\]

\[
\text{eval}(v_i) = \sum_{k=1}^{q} w_k \left( f_k(v_i) - z_k^{min} \right), \; \forall i
\]

output \( \text{eval}(v_i) \), \( \forall i \);

end
```
**Generation 3: Weighted Sum + Elitist Preserve**

- **Strength Pareto Evolutionary Algorithm** *(spEA: Zitzler et al., 1999)*
  - Zitzler, E. and L. Thiele (1999) proposed a new evolutionary approach to multicriteria optimization, the Strength Pareto Evolutionary Algorithm (SPEA), that combines several features of previous multiobjective EA’s in a unique manner.
  - It is characterized by:
    - Storing non-dominated solutions externally in a second, continuously updated population.
    - Evaluating an individual’s fitness dependent on the number of external non-dominated points that dominate it.
    - Preserving population diversity using the Pareto dominance relationship.
    - Incorporating a clustering procedure in order to reduce the non-dominated set without destroying its characteristics.

**spEA** *(Zitzler et al., 1999)*
- The fitness assignment procedure is a two-stage process.
  - First, the individuals in the external non-dominated set \( P \) are ranked.
    \[
    s_i = \frac{n}{N+1}
    \]
    where \( s_i \) is proportional to the number of population members \( j \neq i \) for which \( i \succ j \). \( n \) is the number of individuals in \( P \) that are covered by \( i \) and \( N \) is the size of \( P \).

**Procedure of strength Pareto Evolutionary Algorithm**

```plaintext
procedure: spEA
input: the objective \( f_k \) \((v_k)\) of each chromosome \( v_k \), \( k = 1,2,\ldots,q \), \( \forall i \in \text{popSize} \)
output: fitness value \( \text{eval}(v_i) \), \( \forall i \in \text{popSize} \)
begin
  non-dominated set \( P' \leftarrow \emptyset \);
  dominated set \( P \leftarrow \emptyset \);
  for \( i = 1 \) to \( \text{popSize} \) do
    if \( v_i \) is non-dominated solution then
      \( P' \leftarrow P' \cup \{ v_i \} \);
    else
      \( P \leftarrow P \cup \{ v_i \} \);
      \( N \leftarrow |P| \);
      \( n_i \leftarrow |\{ j \mid i \not\succ j, j \in P \}| \), \( \forall i \in P' \);
      \( s_i \leftarrow n_i / N + 1 \), \( \forall i \in P' \);
      \( \text{eval}(v_i) \leftarrow s_i \), \( \forall i \in P' \);
      \( \text{eval}(v_i) \leftarrow 1 + \sum_{j \not\succ i} s_j \), \( \forall j \in P \);
    end
  output \( \text{eval}(v_i) \), \( \forall i \);
end
```
**Fast nondominated sort**
- They first sorting a population into different nondomination levels.
- Nondomination rank (R)

**Crowding distance assignment**
- The crowding-distance computation requires sorting the population according to each objective function value in ascending order of magnitude.

**Crowding distance (D)**
- For each objective function value, the distance to the nearest individual is calculated.
- The distance is used to determine the relative spacing of solutions.

**Crowding distance (D) for nsGA II.**
Generation 3: Weighted Sum + Elitist Preserve

- The new population $P_{t+1}$ is now used for selection, crossover, and mutation to create a new population $Q_{t+1}$.
- It is important to note that they used a binary tournament selection operator, but the selection criterion is based on the crowded-comparison operator $<_n$.
- Crowded-comparison operator is defined as follows:

$$\text{if } |R_i < R_j| \text{ or } |R_i = R_j \text{ and } D_i > D_j|$$

$$\text{then } i \sim_n j$$

where, $R_i$ is non-domination rank and $D_i$ is crowding distance.

![Fig. Schematic of nsGA II selection](image)

Generation 3: Weighted Sum + Elitist Preserve

- Interactive Adaptive-weight Genetic Algorithm (i-awGA; Lin and Gen, 2005)
  - Generally, to combine Pareto ranking-based approach, there is a clear classification between non-dominated solution and dominated solution.
  - However, it is difficult to keep the two solutions distinct in non-dominated set (or dominated set).
  - Different points from the Pareto ranking-based fitness assignment with the weighted-sum based fitness assignment:
    - Assign weights to each objective function and combines the weighted objectives into a single objective function.
    - However, sometimes there are assigned fitness values of dominated solutions is better than several non-dominated solutions by adaptive-weight based fitness assignment approach.
  - Lin and Gen improved adaptive-weight fitness assignment approaches considering disadvantages of weighted-sum approach and Pareto ranking-based fitness assignment approach called as Interactive Adaptive-weight GA (i-awGA).

Generation 3: Weighted Sum + Elitist Preserve

- We first define two extreme points:
  - The maximum extreme point $z^* = \{z_{1}^{\text{max}}, z_{2}^{\text{max}}\}$ and the minimum extreme point $z^t = \{z_{1}^{\text{min}}, z_{2}^{\text{min}}\}$ in criteria space.
    - where $z_{1}^{\text{max}}, z_{2}^{\text{max}}, z_{1}^{\text{min}}, z_{2}^{\text{min}}$ are the maximum value and minimum value for objective 1 and objective 2 in the current population.
  - Calculate the adaptive weight $w_{i} = 1/(z_{1}^{\text{max}} - z_{1}^{\text{min}})$ for objective 1 and the adaptive weight $w_{2} = 1/(z_{2}^{\text{max}} - z_{2}^{\text{min}})$ for objective 2.
  - Afterwards, calculate the penalty term $p(x^k)$:
    - If $x^k$ is non-dominated solution in the non-dominated set $P$, $p(x^k) = 0$.
    - Otherwise $p(x^k) = 1$ for dominated solution $x^k$.
  - Last, calculate the fitness value of each chromosome:
    - By combining the method which calculates average weight of individuals in the chromosome as follows:
      $$\text{eval}(v_k) = w_1(f_1^k - z_1^{\text{min}}) + w_2(f_2^k - 1/2z_2^{\text{min}}) + p(v_k), \quad \forall k \in \text{popSize}$$
Chapter 1 Multiobjective Genetic Algorithms

1.1 Introduction
1.2 Implementation of Genetic Algorithms
1.3 Hybrid Genetic Algorithms
1.4 Multiobjective Genetic Algorithms
  1.4.1 Basic Concepts of Multiobjective Optimizations
  1.4.2 Features and Implementation of Multiobjective GA
  1.4.3 Fitness Assignment Mechanism
  1.4.4 Performance Measures
  1.4.5 Applications of Multiobjective GA

Reference Solution Set $S^*$

- Reference solution set $S^*$ of each test problem was found using all algorithms which be used in computational experiments.
  - Each algorithm was applied to each test problem with much longer computation time and larger memory storage than the other computational experiments.
  - Generally, we used the very large parameter specifications in all algorithms for finding the reference solution set of each test problem.

1.4.4 Performance Measures

a. The number of obtained solutions $|S_j|$


- Let $S$ be the union of the $J$ solution sets.
- Let $S_j$ be a solution set ($j=1, 2, 3, \ldots, J$). For comparing $J$ solution sets ($S_1, S_2, \ldots, S_J$), we use the number of obtained solution $|S_j|$

\[ R_{NDS}(S_j) = \left\{ \frac{|S_j|}{|S|} \right\} \]

where $S$ is a reference solution set for evaluation the solution set $S_j$. $d_{nr}$ is the distance between a current solution $x$ and a reference solution $r$.

\[ d_{nr} = \sqrt{(f_1(r) - f_1(x))^2 + (f_2(r) - f_2(x))^2} \]

- The $D_{1r}$ measure is not the average distance from each solution in $S_j$ to its nearest reference solution in $S^*$.
- It is referred to as the *generation distance*.
- While the generation distance can only evaluate the proximity of the solution set $S_j$ to $S^*$, $D_{1r}(S_j)$ can evaluate the distribution of $S_j$ as well as the proximity of $S_j$ to $S^*$.

b. The Ratio of nondominated solutions $R_{NDS}(S_j)$

- A straightforward performance measure of the solution set $S_j$ with respect to the $J$ solution sets is the ratio of solutions in $S_j$ that are not dominated by any other solutions in $S$.
- The $R_{NDS}(S_j)$ measure can be written as follows:

\[ R_{NDS}(S_j) = \left\{ \frac{|S_j - \{x \in S_j \mid \exists r \in S^*: r < x\}|}{|S_j|} \right\} \]
1.4.5 Applications of Multiobjective GA

- **Bicriteria Linear Transportation Problem**

- **Bicriteria Minimum Spanning Tree Problem**

- **Bicriteria Nonlinear Programming Problem**

- **Bicriteria Network Design Problem**

---

### a. Bicriteria Linear Transportation Problem

The allocation matrix was used as the chromosome representation.

The special crossover operation and mutation operation proposed by Vignaux and Michalewicz were adopted.

The fitness value for each individual was determined by the adaptive hyperplane method.

The Pareto solutions found by the proposed method are depicted in Figure.

---

### b. Bicriteria Minimum Spanning Tree Problem


Consider the minimum spanning tree problem, where each edge has two associated positive real numbers. Then it can be formulated as the following bicriteria optimization problem:

\[
\min \begin{cases} 
  z_1(x) = \sum_{i=1}^{n} w_1 x_i, \\ 
  z_2(x) = \sum_{i=1}^{n} w_2 x_i 
\end{cases} 
\]

s. t. \( x \in T \)

where \( x \) is a binary decision variable defined as:

\[
x_i = \begin{cases} 
  1, & \text{if edge } e_i \text{ is selected} \\
  0, & \text{otherwise} 
\end{cases}
\]

\( T \) denotes the set of all spanning trees corresponding to a given problem.
b. Bicriteria Minimum Spanning Tree Problem

The Prüfer number was adopted as the tree encoding. It is capable for representing all possible spanning trees.

Uniform crossover and perturbation mutation were used as genetic operations, and the adaptive hyperplane method was used to determine fitness values for each tree.

The Pareto solutions found by the proposed method are depicted in the right figure.

c. Bicriteria Nonlinear Programming Problem

Osyczka and Kundu used the following bicriteria nonlinear programming problem to test the distance method.


\[
\begin{align*}
\min f_1(x) &= -25(x_1-2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2 \\
\min f_2(x) &= (x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \\
s. t. \\
g_1(x) &= x_1 + x_2 - 2 \geq 0 \\
g_2(x) &= 6 - x_1 - x_2 \geq 0 \\
g_3(x) &= 2 + x_1 - x_2 \geq 0 \\
g_4(x) &= 2 - x_1 + 3x_2 \geq 0 \\
g_5(x) &= 4 - (x_1 - 3)^2 - x_2 \geq 0 \\
g_6(x) &= (x_1 - 3)^2 + x_4 - 4 \geq 0 \\
0 \leq x_i \leq 10, \quad i = 1, 2, 3, 4, 5, 6
\end{align*}
\]

d. Bicriteria Network Design Problem


In a network with flow capacities and costs on the arcs, Bicriteria Network Design Problem is to determine both the maximum possible flow \(z_1\) and minimum cost \(z_2\) in the same time, from a source to a sink.

\[
\begin{align*}
\max \quad z_1 &= f \\
\min \quad z_2 &= \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij} \\
s. t. \quad \sum_{j=1}^{m} x_{ij} - \sum_{i=1}^{m} x_{ji} &= \begin{cases} f & (i = 1) \\ 0 & (i = 2, 3, \cdots, m - 1) \\ -f & (i = m) \end{cases} \\
0 \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A \\
f & > 0
\end{align*}
\]
d. Bicriteria Network Design Problem

- The numerical example, presented by Munakata and Hashier, was adopted. The problem comprises 25 nodes and 49 arcs. It is given as follows:

![Bicriteria Network Design Problem Diagram](image)

**Summary**

- The Genetic Algorithms (GA), as powerful and broadly applicable stochastic search and optimization techniques, are perhaps the most widely known types of Evolutionary Computation methods or Evolutionary Optimization today.

- In this chapter, we have introduced the following subjects:
  - Introduction
  - Implementation of Genetic Algorithms
    - GA Vocabulary
    - Encoding Issue
    - Fitness Evaluation
    - Genetic Operators
    - Handling Constraints
  - Hybrid Genetic Algorithms
    - Genetic Local Search
    - Parameter Adaptation

**Multiobjective Optimization Problems (MOP)** arise in the design, modeling, and planning of many complex real systems. Almost every important real-world decision making problem involves multiple and conflicting objectives.

Genetic Algorithms have received considerable attention as a novel approach to multiobjective optimization problem.

We explained how to treat the following multiobjective optimization problems by genetic algorithms:

1.4.1 Basic Concepts of Multiobjective Optimizations
1.4.2 Features and Implementation of Multiobjective GA
1.4.3 Fitness Assignment Mechanism
1.4.4 Performance Measures
1.4.5 Applications of Multiobjective GA