Week 1-3

Involved Algorithms

GAs, PSO, Constraint Handling, Evolutionary Strategy, Differential Evolution, Multi-Objective Algorithms and multimodal optimization.

Principle of Evolutionary Algorithms

- Proper **encoding** is necessary.
- There is a certain index to **quantify the fitness** of every individual.
- The selection of parents and the reproduction of parents is to some extend **random**.
- The population of solutions will be better as the number of generations increases.

Introduction to Optimization

The problem can be described as:

$$Minimize\ f(x)\ over\ x \in \Omega$$

$$\Omega\ is\ the\ permissible\ region\ of\ the\ decision\ variables,\ which\ is\ defined\ by\ constraints.$$

Discrete solution can be considered as an approximate solution to the continuous problem. For example:

$$Minimize\ f(x) = x^2\ where\ -1 \leq x \leq 1$$

The value of x can be discretized for example using 32 bits plus a sign bit, so totally there are 2^{33} discrete solutions.

However, it is not efficient enough because the problems are likely to be a large numer of enumerations. Consequently, real-parameter evolutionary algorithms is much better.

Global Optimum v.s. Computational Cost

Due to the computational time, in most cases it is impossible and unnecessary to find the global solution. Algorithms such as the GA or simulated annealing can be used because these approaches make use of randomness to escape from **local optimal solutions**. However, no method can guarantee to produce the global optimal solution (especially for complex multimodal problems).

Nonlinear Programming Problems

- Can be solved by using methods of calculus, but it is **difficult** to solve these problems analytically.
- The most popular method is the **sequential quadratic programming**.
- **Iterative numerical algorithms** are ideal ways to obtain acceptable solutions.

Properties of GAs

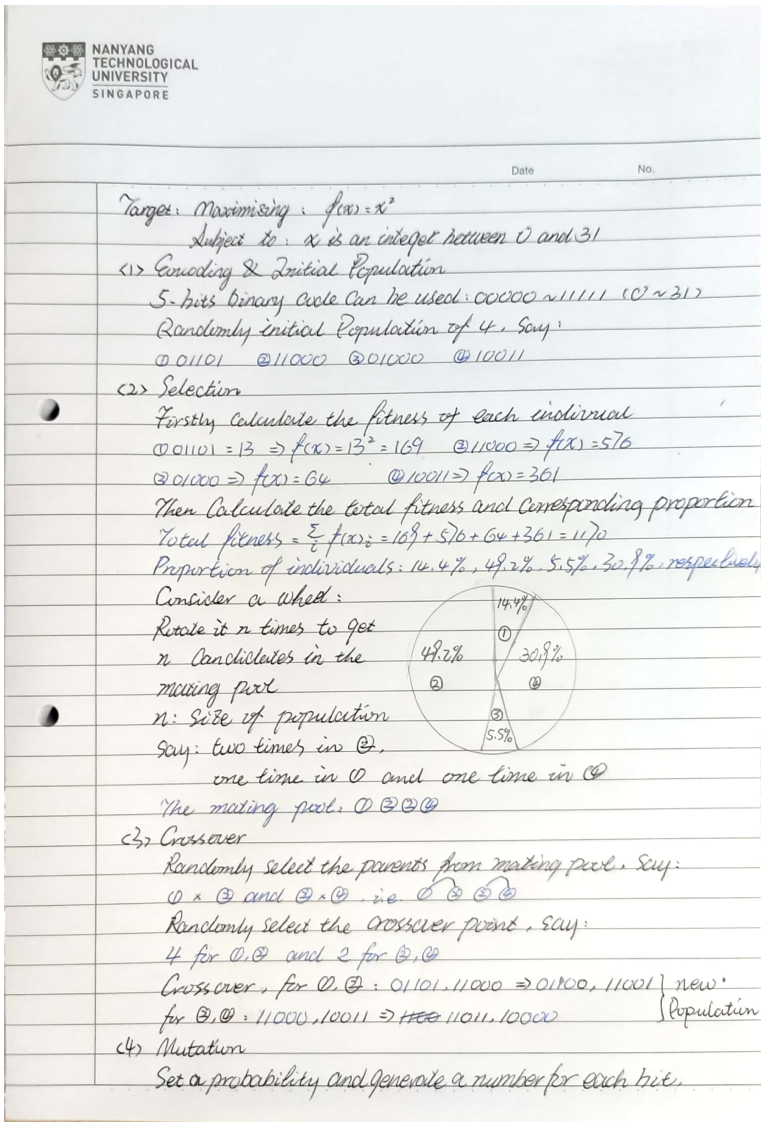
- A simple way to obtain acceptable solutions.
- Do not require the search spaceto be **continuous** or **differentiable** or **unimodal.**
- **Not deterministic rules** so that it can escape from local opimal solutions.
- Work with the objective function without gradient, nor other auxiliary information.
- GAs usually operate on the coded variables instead of directly on the decision variables themselves. However, it has better performance when the decision variables are binary.

Simple Genetic Algorithm (SGA)

▼ **Process of SGA**

1. Encoding & Initial Polulation
2. Selection (Roulette Wheel)
3. Crossover (1-point)
4. Mutation
5. Iteration

▼ **Simple Example**



if the number less than the probability, then change the value of the corresponding bit. ($0 \rightarrow 1$ or $1 \rightarrow 0$)

The above process can be represented as following:

String No.	Initial Popn.	x	$f(x)$	% of Total	No. Sel.	Mating Pool	Mate	C'over Site	New Popn.	x	$f(x)$
1	01101	13	169	14.4	1	01101	2	4	01100	12	144
2	11000	24	576	49.2	2	11000	1	4	11001	25	625
3	01000	8	64	5.5	0	11000	4	2	11011	27	729
4	10011	19	361	30.9	1	10011	3	2	10000	16	256
Sum			1170	100.0	4						1754
Average			293								439

▼ **Things should be noted**

- Due to the property of Roulette Wheel, the fitness of each individual should be positive, otherwise, linear translation should be made as a component to make sure the fitness value is positive.

$$f(x) = \begin{cases} u(x) + C_{\min} & \text{if } u(x) + C_{\min} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- The probability of mutation should be properly setted so that for every generation, there is at least mutated one mutated bit.
- The total fitness and average fitness can be utilized to quantify the performance of generation.

The computer implementation section is clear and simple in suganthan's slides, and it can be found everywhere in github. So it will not be metioned in this material.

Fitness Scaling

Motivation: In early generation, some highly fit individuals may dominate the selection and lead to a premature convergence.

▼ **Linear Fitness Scaling**

Scaled function: $g = af + b$

Where f is fitness function, a and b are parameters which can be calculated by:

Maintaining average fitness values before and after scaling: $g_{avg} = f_{avg}$

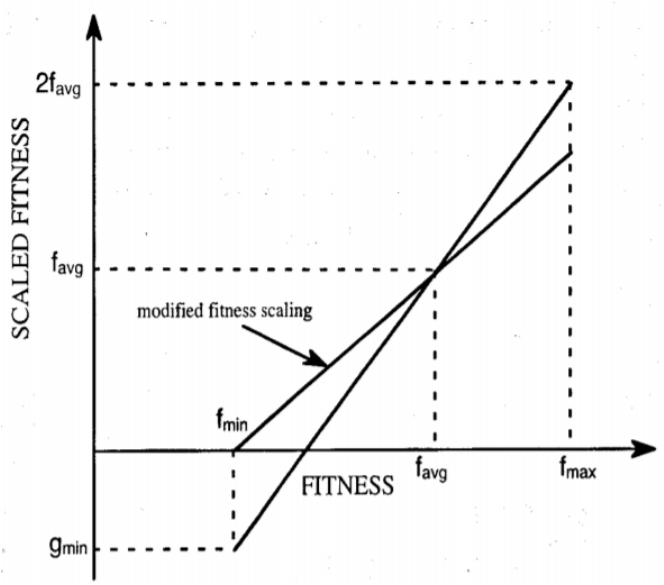
A rule of thumb: $g_{max} = Cf_{avg}$, for example, for at most 2 samples in the mating poole just set C=2

Then: $a = \frac{f_{avg}(C-1)}{f_{max}-f_{avg}}, b = f_{avg}(1-a)$

However, this scaled function may produce negative value in later generations.

One solution is modify it to: $a = \frac{f_{avg}}{f_{max}-f_{avg}}, b = f_{avg}(1-a)$

Fitness scaling is only done in the process of Selection.



Sigma Truncation

This method can be used before linear scaling

$f' = f - (f_{avg} - c\sigma)$

σ is the variance, c is a scale factor usually set around 2-3

Any negative value of f' is set to zero.

Power Law Scaling

$g(f) = f^{\alpha}$

A rule of thumb:

- If $0 \leq f \leq 1$, and if you wish to increase the separation, then $0 < \alpha < 1$.
- If $1 < f$ and you wish to increase the separation, then $1 < \alpha$

Reduce the separation in early generations and increase the separation when close to termination.

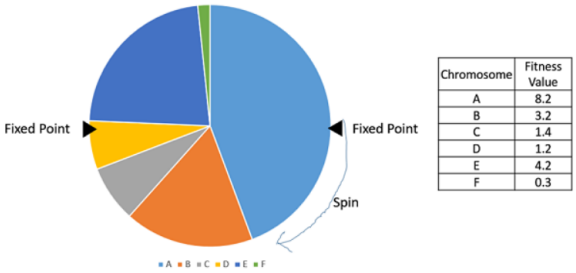
Improvement Stategies

Selection

Motivation: In early generations, individual with highly fitness may be too frequently to be selected so that it dominate the population.

Stochastic Universal Selection

Fix n point on the wheel's background and rotate the wheel once and get n individuals. n is the size of population.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Tournament Selection

Every time, 2 (or more) individuals are randomly selected and their fitness values are compared. The best on is included in the mating pool and all the individuals are returned to the population.

Repeat this process until the size of mating pool is enough.

There is some wrong of Ranking Selection in suganthan's slides, so it is not mentioned here.

Crossover

2-point Crossover

Randomly select 2 points insted of 1 point, which can combine more features of individuals.

For example: parents are 110000 and 001111, crossover points are 2 and 4.

The offsprings are 110011 and 001100.

Uniform Crossover

For each offspring, randomly select the value of its parents. For example, parents are 110000 and 001111

One of the offsprings maybe:010101

Population Replacement

Motivation: In SGA, all the generation is replaced by the offspring, however, the offspring maybe worse than their parents. This strategy is called non-overlapping.

▼ Elitism

Just copy the best or few best parents into the next generation to replace the worst children or some worst children randomly.

▼ Steady-State Reproduction

Only replace few members in every generation instead of replacing all the individuals.

▼ $\mu + \lambda$ Selection

μ offspring and λ parents are chosen as parents in the next generation.

If λ is very small, it is Elitist strategy.

Mathematical Analysis of SGA

▼ **Schemas**

Use wild card character '*' to represent a subset individuals (strings).

*101 represents the subset {1101, 0101}

If the string length l is 5, then there are 3^5 different schemas in total (for binary code).

▼ **Order of Schema**

The order of Schema H is denoted by $o(H)$, which indicates the number of fixed positions in the string.

E.g. $o(011**)$=3, $o(1****)$=1

▼ **Defining Length of Schema**

Distance between the first and last fixed positions, denoted as $\delta(H)$.

E.g. $\delta(1**1) = 4 - 1 = 3$, $\delta(011*11*) = 6 - 1 = 5$

▼ **Survival & Propagation of Schema**

$m(H, t)$ is used to represent the number of instances exist in the population at generation t.

101* is more likely to be destroyed than1*** in the evolution process.

1***1 is more likely to be destroyed than 11*** in the crossover process.

Theorem1: Short&Low order schemas with above population average fitness will get exponentially increasing numbers in the subsequent generations.

\# The author will introduce Bandit Problem later in PSO algorithm because Bandit problem seems like not important in this section.

▼ **GA and Bandit Problem**

Just as in the k-armed bandit problem, we allocate exponentially increasing numbers to the best schema.

In general, for schema with order j and length l, there are C_j^l different 2^j-armed bandit problems.

E.g. with l=7 and j=3, there are C_3^7=35 different 8-armed problems.

Building Block Hypothesis

As we known in Theorem1, the Short&Low order schemas can be called Building Block.

The number of Building Block will increase exponentially so that we can obtain Blocks with higher fitness.

Hypothesis: Building Blocks will combine into Long&High order&High fitness schema and finally get the optimal solution because of Genetic Operator.

GA-Deceptive Problems

If Short&Low order schemas do not include the global optimal solution, the algorithm will get premature convergence, which is called Deceptive Problem.

Minimal Deceptive Problem (MDP)

No order-1 problems is GA-deceptive. The smallest possible GA-deceptive problem is an order-2 problem.

Consider 4 order-2 schemas with 2 fixed positions:

[***0*****0*] f(00)

[***1*****0*] f(10)

[***0*****1*] f(01)

[***1*****1*] f(11)

We set $f(11)$ as global optimal solution

There are two situations:

Type1: $f(01) > f(00)$

Type2: $f(01) < f(00)$

For type1: GA can almost always get global optimal solution, while for type2 sometimes GA can only get the local optimal solution.

The results shown in suganthan's slides:

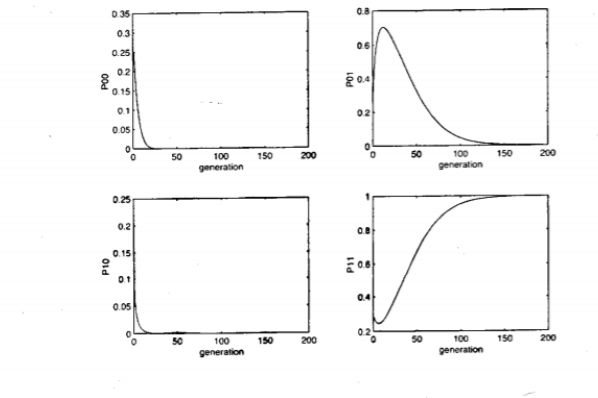


Figure 3.1 Type 1: $f_{01} = 1.05$, $f_{10} = 0$, $f_1 = 1.1$ and equal initial proportions.

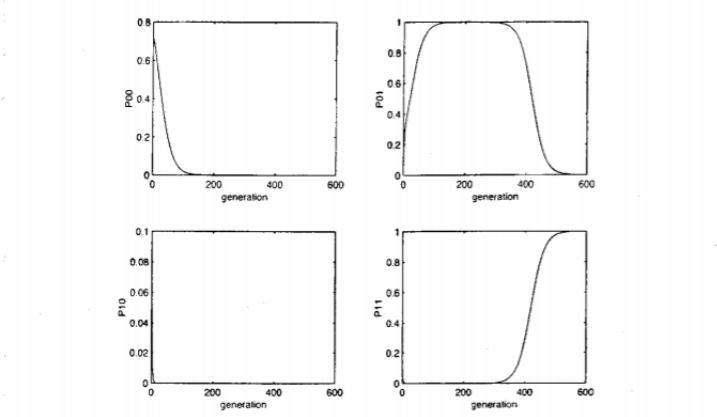


Figure 3.2 Type 1: $f_{01} = 1.05$, $f_{10} = 0$, $f_1 = 1.1$ and unequal initial proportions.

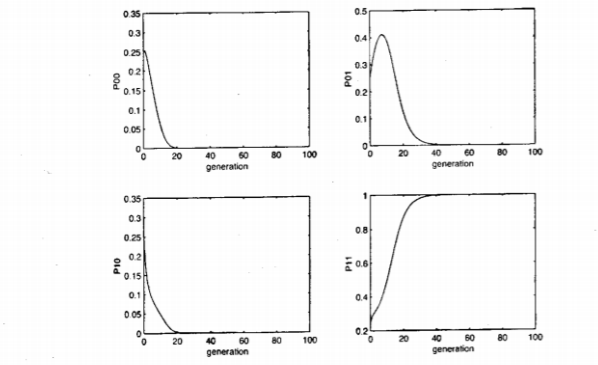


Figure 3.3 Type 2: $f_{01} = 0.9$, $f_{10} = 0.5$, $f_1 = 1.1$ and equal initial proportions.

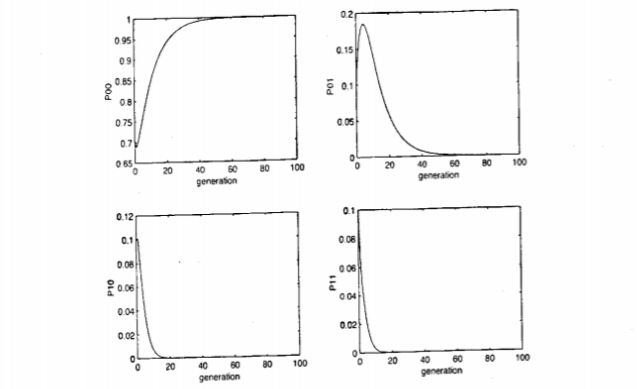


Figure 3.4 Type 2: $f_{01} = 0.9$, $f_{10} = 0.5$, $f_1 = 1.1$ and unequal initial proportions.

Solution: Adjusting the fitness function by operator like log or utilizing Gray coding. (Suganthan did no mention it.)