Chapter 1
Introduction to Evolutionary Algorithms

Dae-Ki Kang

Notice and disclaimer!

- Most contents of these slides are simple and literal translation of the Korean lecture slides of the following book.
  - 쉽게 배우는 유전 알고리즘/문병로/한빛미디어
- These slide are made simply because there are no English lecture slides for the original lecture slides, i.e. purely educational purpose.
- Any form of the reuse of the materials should be done only after the permission of the original author (문병로)

Objectives of this class

- Check the history of evolutionary algorithms
- Understand the basic architecture of evolutionary algorithms and their operators
- Understand that evolutionary algorithms are not always suitable for all problems.

Evolutionary Algorithms

- Genetic Algorithm= GA
- Problem solution space search method using the evolution principle of entities in population genetics
- One field of “Evolutionary Computation”
- key features - natural selection, crossover, mutation, population, etc.

Brief of the history

- During 1950’s and 1960’s : Independent studies (without communication) by some researchers
- Genetic Algorithm(John Holland)
- Evolutionary Programming (Fogel, Owens, Walsh) – crossover was not used
- Evolution Strategy (Rechenberg) – Start with single thread. Similar to current GA
- 1975, John Holland,「Adaptation in Natural and Artificial Systems」
- 1984, due to Holland , SantaFe Institute changes research direction from Complex System to Adaptive Complex System
- 1985, 1st International Conference on Genetic Algorithms (ICGA)
- 1989, David Goldberg,「Genetic Algorithms in Search, Optimization and Machine Learning」
- 1990's Explosion of interests, growing in quantity and quality
- 1997, IEEE Transactions on Evolutionary Computation

Basic terms

- Evolutionary Computation
  - EC = GA + GP + EP + ES
    - EC : Evolutionary Computation
    - GA : Genetic Algorithm
    - GP : Genetic Programming
    - EP : Evolutionary Programming
    - ES : Evolution Strategy

- Terms
  - chromosome
  - population
  - gene
  - genotype
  - phenotype

Structure of Genetic Algorithms

[Algorithm 1-1] Structure of GA

```
Create n initial chromosomes;
repeat {
        for i ← 1 to k {
                pick two chromosomes  p1, p2;
                offspring_i ← crossover (p1, p2);
                offspring_i ← mutation (offspring_i);
        }
        replace k chromosomes in population with offspring_1 , …, offspring_k
} until (stopping condition is met);
return the best chromosomes from the population;
```

- crossover – create a new chromosome by partially combining two chromosomes
- mutation – change a very small portion of a chromosome
  - k/n : generation gap
  - k ≈ n : generational GA
  - k ≈ 1 : steady-state GA
- Stopping condition
  - e.g. – fixed number of iterations
    – probability of population convergence

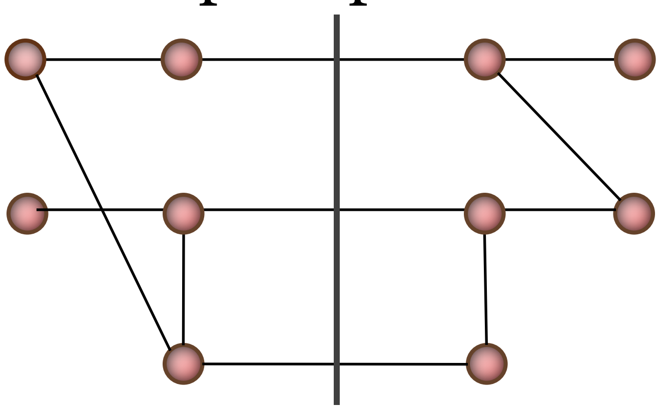
Representation

- In the past, chromosome is represented in binary
  Binary: 00110010 … 00011111 → Hexadecimal: 32…1F

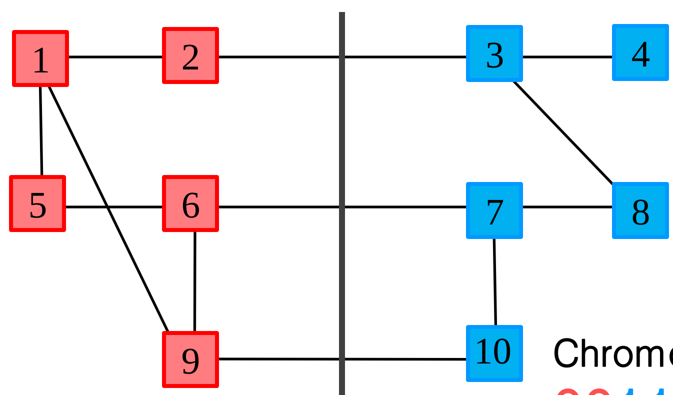
- Chromosome and solution

chromosome	solution	
1	1	Common
k	1	Rare
1	k	Very rare

- Graph Bipartition



Graph Bipartition



Chromosome
0011001101

In chromosome representation

Schema

- Patterns inside chromosomes
- Example
  - In one binary chromosome whose length is n, there are 2^n schemas
  - Pattern 11*1 is included in chromosomes 1101 and 1111
  - In chromosome 1101, there are 16 sub-patterns (1***, *1**, **0*, ***1, 11**, 1*0*, 1**1, *10*, *1*1, **01, 110*, 1*01, 11*1, *101, 1101, ****)
- Schema related terminologies
  - * means don't care
  - 0 and 1 are specific symbols.
  - Defining length is a number of symbols from the leftmost specific symbol to the rightmost specific symbol in a schema
  - order is a number of specific symbols in a schema
  - The example in the right has the order 4

1101***

order

Crossover

- An operation that combines two solutions and generate one new solution.

- Example
  - One-point

a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
a	b	c	d	e	f	q	r	s	t

  - Three-point

a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
a	b	m	n	o	p	g	h	i	t

Mutation

- An operation that introduce a new attribute that is not in the parents to a child solution.
- Example

a b c d e f g h i j

a b c d e f g h x j

  - Usually the probability is 0.015, or 0.01

- Crossover vs. Mutation
  - Crossover- more exploitation of the existing solutions
  - Mutation- more exploration of a new problem space

Replacement

- Generational GA has no difficulties in replacement
  - When all generations are replaced, there is no choice
- In steady-state GA, replacement is important for the performance
  - Replace the solution of the worst quality
  - Replace one of the parents
    - To preserve the diversity, remove the one that is most similar to the new one
    - Use Hamming distance to calculate the similarity
- Replacement should be determined with the consideration of crossover and mutation

Problems that GA is useful on

- Problems that GA can help
  - The problems that traditional derterministic methods cannot solve easily
- Problems that GA cannot help
  - The problems that traditional derterministic algorithms can solve easily
  - The nature of the problems is what traditional derterministic algorithms cannot solve easily, but the particular problem under consideration is too small (so it is easy and fast to cover all of the problem space)
    - e.g. 5-city TSP