Genetic Algorithm - Metaheuristic approach

# Introduction

Genetic Algorithms were invented to mimic some of the processes observed in natural evolution, especially follow the principles laid down by Charles Darwin of “survival of the fittest”. The idea with GA is used to solve optimization problems.

# Intuition

- Initial

- Evolution

+ Selection: Choose parent in based on f(x) (fitness function)

+ Generate: Create offspring with genetic operator

+ Selection: produce new population

- Iteration with N

- Finally,

# Terminology

### Chromosomal Representation

Each chromosome represents a legal solution to the problem and is composed of a string of genes.

### Initial Population

The first population is usually created randomly. From empirical studies, over a wide range of function optimization problems, a population size of between 30 and 100 is usually recommended.

### Fitness Evaluation

Fitness evaluation involves defining an objective or fitness function against which each chromosome is tested under consideration. As the algorithm proceeds we would expect the individual fitness of the "best" chromosome to increase as well as the total fitness of the population as a whole.

### Selection

We need to select chromosomes from the current population for reproduction.

Genetic operator

Combine parents to create offspring by using Mutation and cross over

# Implementation Detail

The algorithm evolves the through three operators:

1. **selection** which equates to survival of the fittest;
2. **crossover** which represents mating between individuals;
3. **mutation** which introduces random modifications.

## Data representation

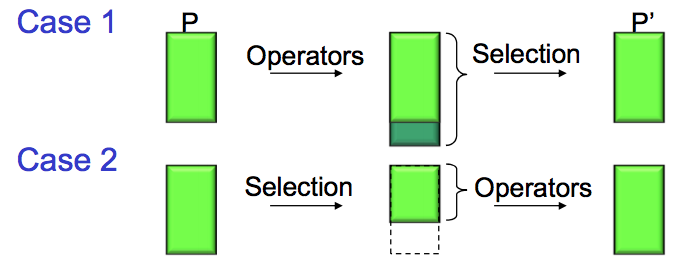
* A common way to representation for chromosomes is **fixed** length bit strings (like DNA)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

* Other types: Integer ( presenting for bits string) , reals, alphanumeric

## Genetic Evolution

* Many variances but remember the principle that keep population P of fixed size



* **Global replacement**: new population is composed of only offspring
  + Radical change but we sometimes lost good information
* **Steady state replacement**: generate small number of offsprings and replace with parent (random or worst)
  + Maybe worst offsprings will be inserted but it will be elimination in next turn.
* **Elitism**: keep k best parent and add new offsprings

## Selection Algorithm

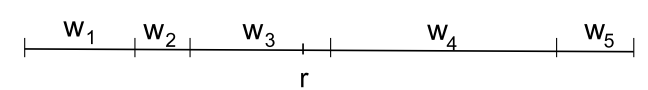
### . Deterministic Selection

* Keep best k chromosomes based on f(x)
* Fast but no randomness

### . Probabilistic

Select x with probability p(x) with the idea . Then we have probability of chromosome:

#### - Roulette Wheel Selection

- split [0,1] into n bins with

- random number r.

- choose bin i that r lies in

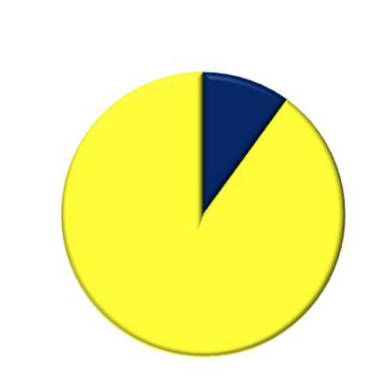
#### - Stochastic Universal Sampling

****- split [0,1] into n bins with

- random number r.

- select all bins that contain **(N is number of sample)**

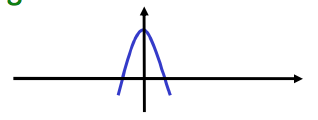
RWS have lower variance than SUS

EXAMPLE:

|  |  |  |
| --- | --- | --- |
|  | RWS | SUS ( in 10 samples, so 9 samples are x1 and 1 sample is x2 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### \*\* REMIND: Probabilistic sampling problem

* selection pressure (**measurement)**: criteria for the selection function to have **good properties to be used** in selection
  + . with = expected number of selections for best chromosomes
  + if too high, premature convergence, distribution of chromosomes is sharp,
    - too much focus and almost no search because it is trapped on a local maximum



* + - low discrimination, almost uniform sampling, selection become not really efficient.



* + Linear adjustment of f: so . That means we can get by adjusting
  + Other adjustment: Exponential, Botzman

### . Selection by Ranking: Order population decrease by fitness. Select with probability with new form of probabilities



### . Tournament Selection: choose k random chromosomes in population, keep the best one by f(x) and continue sampling until create new population

* probability that best chromosome is among k selected:
* , pressure increase with k

### . Stochastic Tournament Selection: select 2 random chromosomes, and keep the best one with fixed probability q (0.5 < q < 1)

* probability that best chromosome is and this one win

## Genetic operator: generate offstrings from parents

### Mutation:

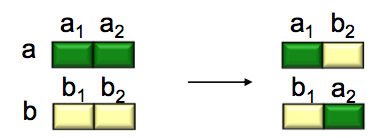
* Randomly invert bits:

|  |  |
| --- | --- |
|  |  |

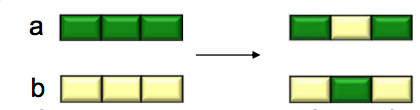
* + is mutation probability (0.1-0.01)
  + ensure every chromosome can be transform so that all the solutions are reachable (global maximum)

### Crossover:

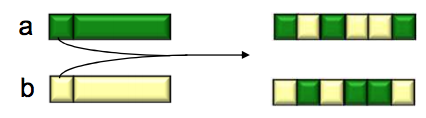
* **1 point crossove**r: choose random split



* **2 point crossover**: choose random segment



* **random crossover**: random choice for each bit (strong modification)



### Conclusion:

* Each genetic operator have their own roles
  + Cross-over is explorative, combine information from parent so it makes a big jump at fitness value.
  + Mutation is exploitative, introduce new information so it creates small diversion.

# Genetic Example:

## Problem: find over [0…31]

* Data presentation: binary code
* Fixed population size: 4
* Evolution: roulette wheel selection, 1-point crossover, bitwise mutation
* Run 1 one circle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initialization | | | Selection | | |
| Initial Population | Value x | Fitness f(x)= | Probability | Expected Count | Actual Count |
| 0 1 1 0 1 | 13 | 169 | 0.14 | 0.58 | 1 |
| 1 1 0 0 0 | 24 | 576 | 0.49 | 1.97 | 2 |
| 0 1 0 0 0 | 8 | 64 | 0.06 | 0.22 | 0 |
| 1 0 0 1 1 | 19 | 361 | 0.31 | 1.23 | 1 |
| Max | 576 | Average | 293 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Crossover | | | | |
| Population after selection | Cross-over point | Offsprings | Value x | Fitness f(x)= |
| 0 1 1 0 1 | 4 | 0 1 1 0 0  1 1 0 0 1 | 12 | 144 |
| 1 1 0 0 0 | 25 | 625 |
| 1 1 0 0 0 | 2 | 1 1 0 1 1  1 0 0 0 0 | 27 | 729 |
| 0 1 0 0 0 | 16 | 256 |
| Max | 729 | Average | 439 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Mutation with probability | | | |
| Population after crossover | Offsprings | Value x | Fitness f(x)= |
| 0 1 1 0 0 | 0 1 1 0 0 | 12 | 144 |
| 1 1 0 0 1 | 1 1 **1** 0 1 | 29 | 841 |
| 1 1 0 1 1 | **0** 1 0 1 1 | 11 | 121 |
| 1 0 0 0 0 | 1 0 0 0 0 | 16 | 256 |
| Max | 841 | Average | 340 |

Comment: Cross-over or mutation can both give better or worse offsprings but it give us a chance to find best elitism. Even in last mutation, our average population decrease and we obtain better result, it shows that loss of good population is possible and bigger population means lower chances to degenerate it.

## Hamming Cliff problem

* Problem Statemetn: for x = [-16,16]
* Data presentation:
* \*\*Optimum point , second best
  + Cross-over: cannot generate from good parent
  + Mutation: cannot flip all positions
* \*\* Neighbour:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

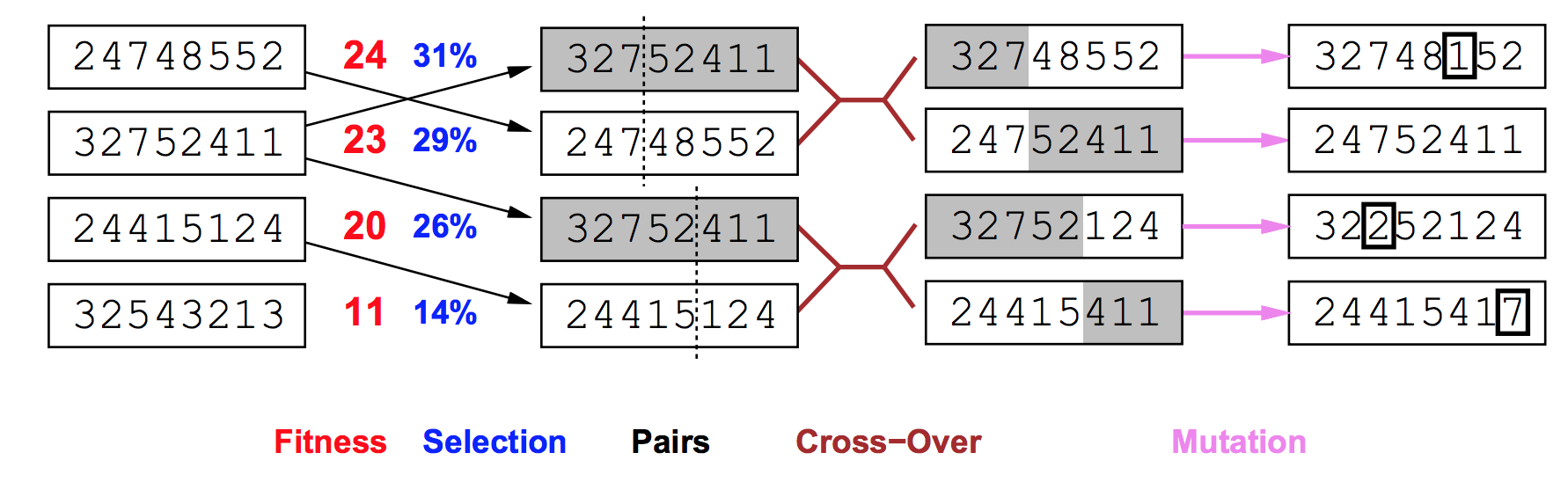
* + Neighbours have poor performance
* Problem solve by using new data presentation then g(i) and g(i+1) only differ 1 bit

## 8-Queen problem

* Problem: place 8 queens on a chessboard such that no queen attacks any others



* Data presentation: State/Position is position of each queen in each column
  + Ex: [46827135] represent a case in image above
* Fitness function: number of non-attacking queens (min=0, max=28)
  + Higher value better result



* Cross-over: swap segment
* Mutation: invert random segment (choose segment and permutation), switch 2 random positions or change it
* Reference: http://research.ijcaonline.org/volume122/number12/pxc3905005.pdf

# 2 Support theorem and some example for this algorithm

## Schema:

Definition:

* present a subset of D
* s contains chromosomes. = number of defined bit and is length of schema
* each chromosome belong to schemas

## Schema Theorem

* provide estimate of number of schemas during evolution (selection, cross-over and mutation) to analyse how chromosomes vary in schema
* Definition:
  + Population **P**, size **n**, at time **t**, schema **s**.
  + **m(s,t)** = number of chromosomes of **s** in **P** at time **t**

### Selection:

* probabilistic selection:
  + let (expect value of F over P and s)
* Assume that we choose only 1 best chromosome. Our current population is
  + Expect value of number of chromosomes after 1 selection
  + Expect value of number of chromosomes after n selections
  + Expect value of number of chromosomes of s after n selections
  + So if we have good schema, ratio and then number of good chromosomes increases

### Crossover:

* We only try applying 1 point crossover with probability that means (1- ) population (after selection step) will be in
  + **max distance** between defined binary values
  + suppose belongs to schema s:



* + - if i (a cross-over point) is outside d(s): schema preserved
    - if I is inside d(s): possible that both offspring are not in s. Schema can be destroyed.
  + We will find numbers of survive chromosomes after cross-over
    - Probability that a chromosome of s does not produce a chromosome of s is
    - E(number of chromosomes of s are destroyed)
    - Expect value of number of chromosomes of s after cross-over:

### Mutation:

* Each chromosome will be mutated with probability
  + Survival probability (probability that each bits should not be mulated)
  + Expect value of number of chromosomes of s in new population

### Conclusion:

* are less than 1, so quality of schema at beginning is quite important so that

## The Argument

* Theorem: Under reasonable assumptions, random population of size N **sample schemas** ( 100 chromo => 10^6 schemas)
* The proof: It is not very useful to understand this. You can look for it online
* Intuitive interpretation:
  + Increase a population, we also sample more schema so that we have greater chances to meet “good” schema
  + Since good schema is created during evolution, we have greater chance to get global maximum