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 P_0 $(P \subset D)$
- Evolution
    + Selection: Choose parent in P_i based on f(x) (fitness function)
    + Generate: Create offspring with genetic operator

+ Selection: produce new population P_{i+1}

- Iteration with N
- Finally, $argmax_{x \in D} f(x) \approx argmax_{i, x \in P_i} f(x_i)$

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

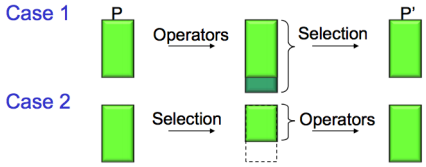
- A common way to representation for chromosomes is **fixed** length bit strings (like DNA)

0	1	0	1	0	0	0	0	1	1	0	0	1	1

- Other types: Integer (presenting for $\log_2 N$ 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

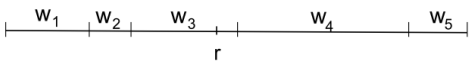
- Keep best k chromosomes based on f(x)
- Fast but no randomness

. Probabilistic

Select x with probability p(x) with the idea $p(x) > p(y) when\ f(x) > f(y)$. Then we have probability of chromosome:

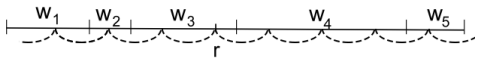
$$p(x) = \frac{f(x)}{\sum_{y \in P} f(y)}$$

. Roulette Wheel Selection



- split [0,1] into n bins with $size_{bin_i} = p(x_i)$
- random number r.
- choose bin i that r lies in

. Stochastic Universal Sampling

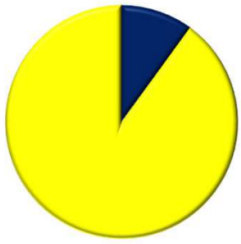


- split [0,1] into n bins with $size_{bin_i} = p(x_i)$
- random number r.
- select all bins that contain $\left(r + \frac{k}{N}\right) mod\ 1\ \ k = 0,1,2..N-1$ **(N is number of sample)**

RWS have lower variance than SUS

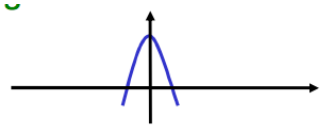
EXAMPLE: $p(x_1) = 0.9, p(x_2) = 0.1$

	RWS	SUS (in 10 samples, so 9 samples are x1 and 1 sample is x2
(x_1, x_1)	$0.9 * 0.9 = 0.81$	$\frac{C_2^9}{C_2^{10}} = 0.8$
(x_1, x_2)	$0.9 * 0.1 = 0.09$	$\frac{9 * 1}{C_2^{10}} = 0.1\ (numbers\ of\ way\ to\ choose\ first\ and\ second\ element)$
(x_2, x_1)	$0.9 * 0.1 = 0.09$	$\frac{1 * 9}{C_2^{10}} = 0.1 (numbers\ of\ way\ to\ choose\ first\ and\ second\ element)$
(x_2, x_2)	$0.1 * 0.1 = 0.01$	$\frac{C_2^1}{C_2^{10}} = 0$



**** REMIND: Probabilistic sampling problem**

- selection pressure (**measurement)**: criteria for the selection function to have **good properties to be used** in selection
  - $p_s = \frac{\hat{f}}{\bar{f}} = \frac{\max_{x \in P} f(x)}{\frac{1}{|P|}\sum_{(y \in P)} f(y)}$. with p_s = expected number of selections for best chromosomes
  - if p_s 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: $p(x) = \frac{af(x)+b}{\sum_{(y \in P)} f(y)+b}$ so $p'_s = \frac{a\hat{f}+b}{a\bar{f}+b} = \frac{\frac{\hat{f}}{\bar{f}}+\frac{b}{a\bar{f}}}{1+\frac{b}{a\bar{f}}} = \frac{p_s+b'}{1+b'}$. That means we can get p'_s by adjusting b'
  - Other adjustment: Exponential, Botzman $\exp(\frac{f}{T})$

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

- $p(x_i) = \lambda\left(1 - \frac{i}{n}\right)^k \quad k = \{1 \dots n\}$
  $\hat{p}_s = \max_i p(x_i) = p(x_0) = \lambda$
- $\bar{p} = \frac{\lambda}{n}\sum_{i=1}^{n}\left(1 - \frac{i}{n}\right)^k \approx \lambda \int_0^1 x^k dx = \frac{\lambda}{k+1} \Rightarrow p_s = k + 1$

. 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: $p = 1 - \left(\frac{n-1}{n}\right)^k \approx \frac{k}{n}$
- $p_s = \frac{\hat{f}}{\bar{f}} = \frac{\frac{k}{n}}{\frac{1}{n}} = k$, pressure increase with k

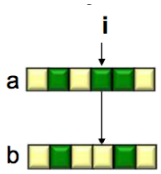
. Stochastic Tournament Selection: select 2 random chromosomes, and keep the best one with fixed probability q (0.5 < q < 1)

- probability that best chromosome is $\frac{2}{N}$ and this one win $\frac{2q}{N}$
- $p_s = \frac{\hat{f}}{\bar{f}} = 2q$

Genetic operator: generate offstrings from parents

Mutation:

- Randomly invert bits:



$$b_i = \begin{cases} a_i \; with \; 1 - p_m \\ 1 - a_i \; with \; p_m \end{cases}$$

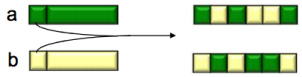
- p_m is mutation probability (0.1-0.01)
- ensure every chromosome can be transform so that all the solutions are reachable (global maximum)

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
  - Mutation is exploitative, introduce new information so it create small diversion