Real Valued Genetic Algorithm

Wakanda

Smeet Shah - 15D110003
Akshay Kumar Ahirwar - 15D170022
Arijit Pramanik - 150020094

Idea Behind Genetic Algorithms

- **Search based optimization** - does not use derivative knowledge or hessian matrix.
- Based on Darwin's theory of evolution - Natural selection and Survival of the fittest.
- Useful traits developed by parents are passed to their offsprings. It helps species to compete better in the world.
- **Evolution is an optimization process** - more fitter individuals in newer generations
- This idea is used to obtain better solutions to an optimization problem at each iteration.

Why GAs?

- GAs provide **usable near-optimal solutions** to NP-Hard problems in a short amount of time.
- GAs provide **global or near-global optimum solutions** unlike gradient-based methods which have a tendency to get stuck in local optimas.
- GAs provide 'good-enough' solutions to difficult problems **much faster**.

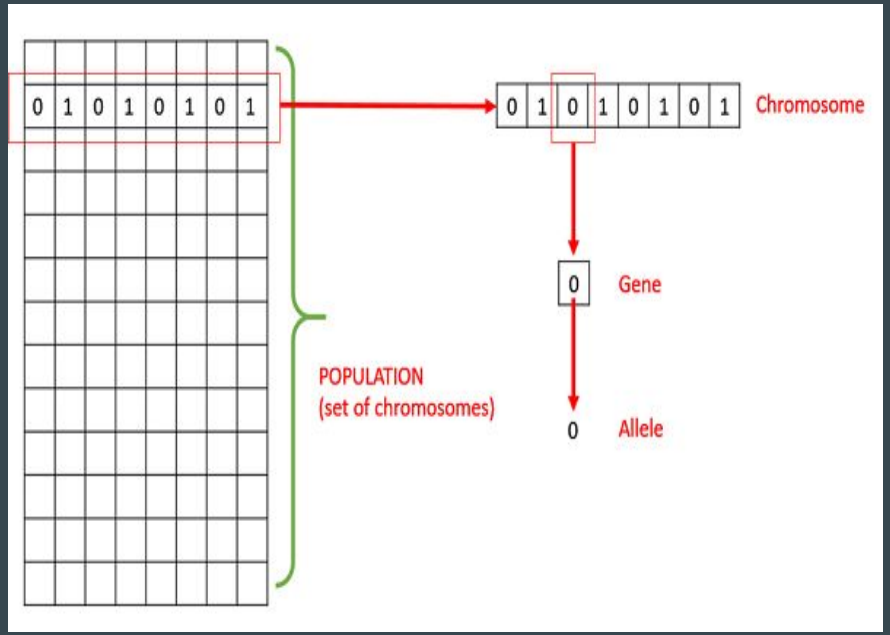
Terminology

Population: subset of solutions to the problem at a given iteration. (Parent Species)

Chromosome: one of the solutions to the problem. (One Parent)

Gene: one element position of a chromosome. (Developed function of Parent)

Allele: value of a gene



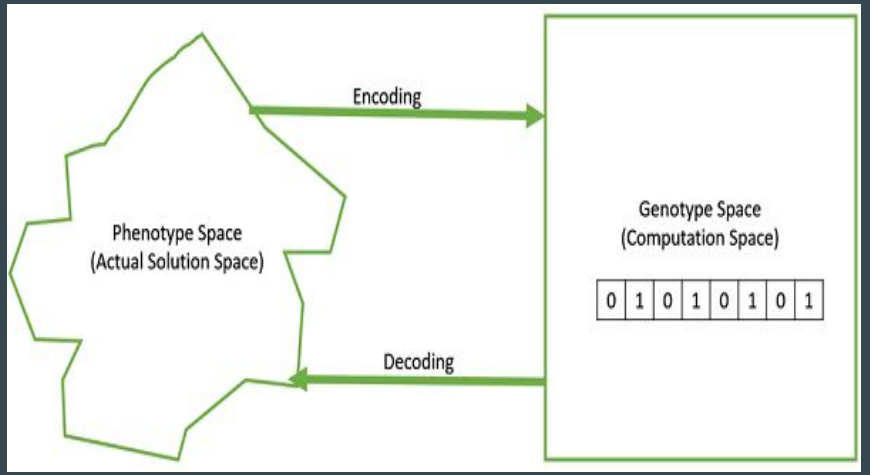
Terminology

Genotype: representation in computational space. Easy to carry out computations in this space.

Phenotype: representation in actual solution space. Real world solution.

Fitness Function: a solution as input, suitability (fitness) of solution as output.

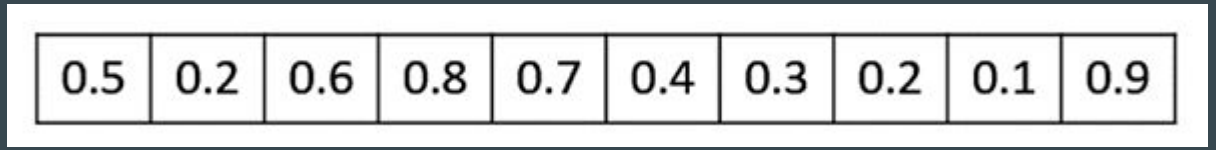
Genetic Operators: Operations for Crossover, Mutation, Selection, etc.



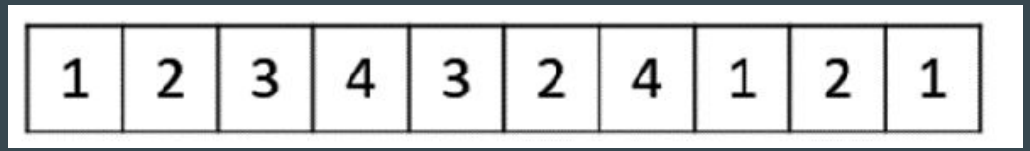
Genotype Representation

- Binary

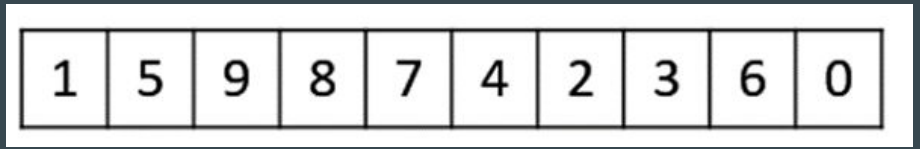
- Real Valued



- Integer

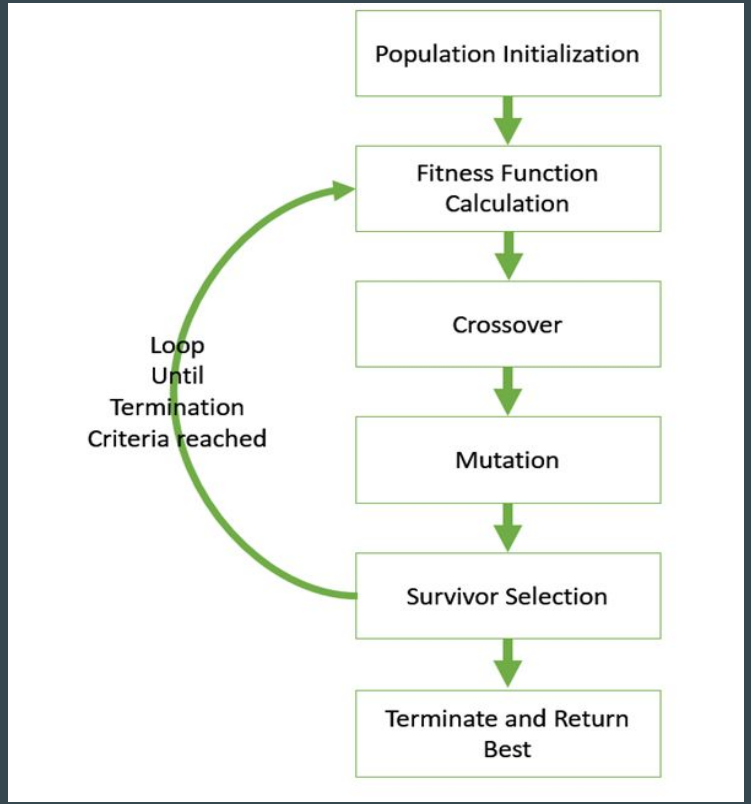


- Permutation



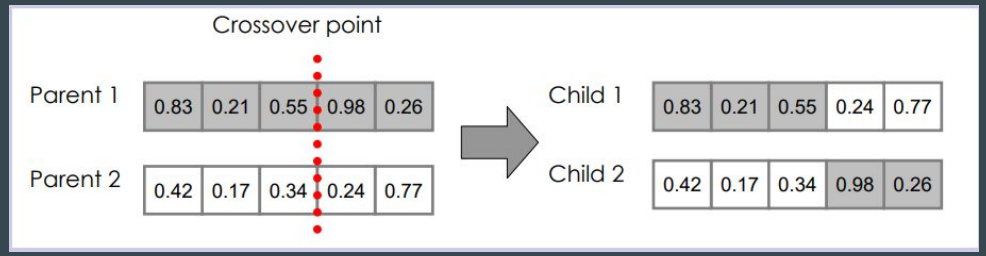
Algorithm Details

1. Give initial solutions (random or heuristic).
2. Calculate fitness and choose fitter parents for reproduction.
3. Do crossover and mutation to generate offsprings (new solutions).
4. Fitness based survivor selection between parents and offsprings.
5. Termination criteria
    a. No improvement in population
    b. Upto fixed number of iterations
    c. Upto fixed pre-defined function value

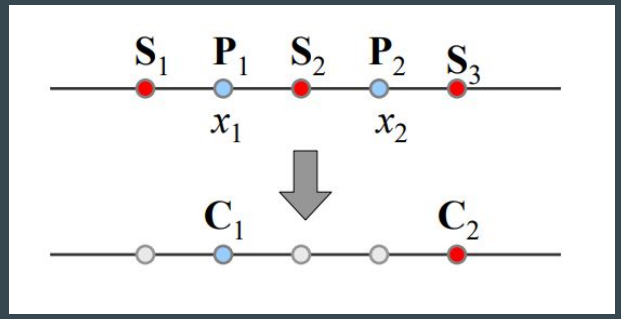


Crossover Operations

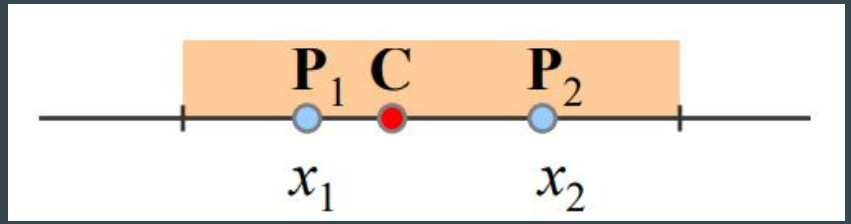
- **Single Point Crossover**



- **Linear Crossover**
  - $0.5x_1 + 0.5x_2$, $1.5x_1 - 0.5x_2$, $-0.5x_1 + 1.5x_2$
  - Best two solutions out of five

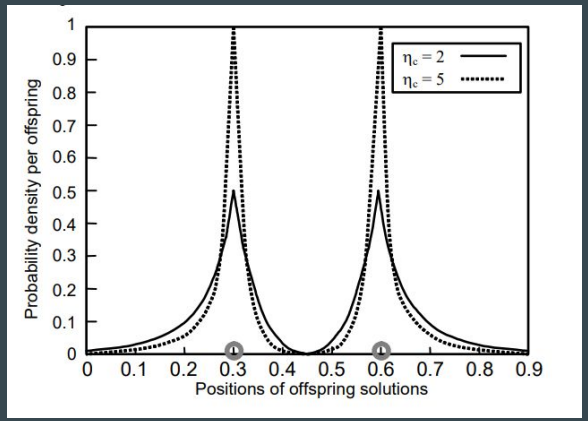


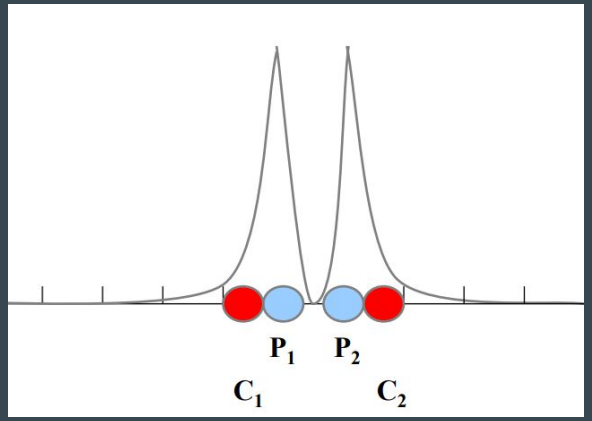
- **Blend Crossover**
  - Randomly select a solution from the range $[x_1 + \alpha(x_2-x_1), x_2 - \alpha(x_2-x_1)]$
  - Often $\alpha = 0.5$

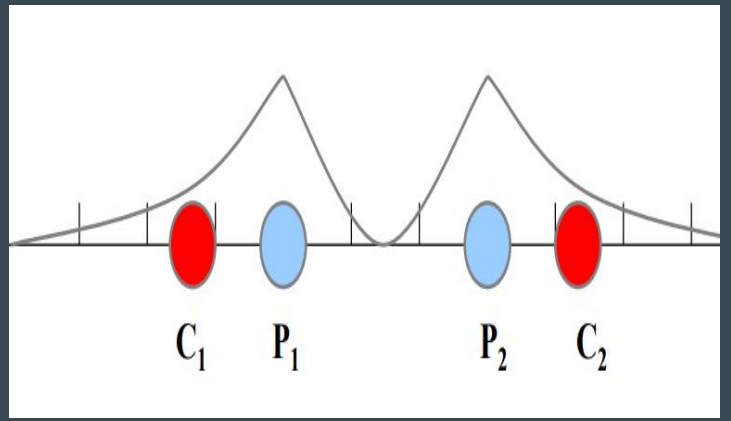


Crossover Operations

- **Simulated Binary Crossover**
  - Generate a random number u in [0,1).
  - Calculate β: $\beta = (2u)^{1/(\eta c+1)}$ if u <= 0.5, $\beta = (1/(2(1-u)))^{1/(\eta c+1)}$ otherwise
  - Generate offsprings $C_1 = 0.5[(1+\beta)x_1 + (1-\beta)x_2]$, $C_2 = 0.5[(1-\beta)x_1 + (1+\beta)x_2]$



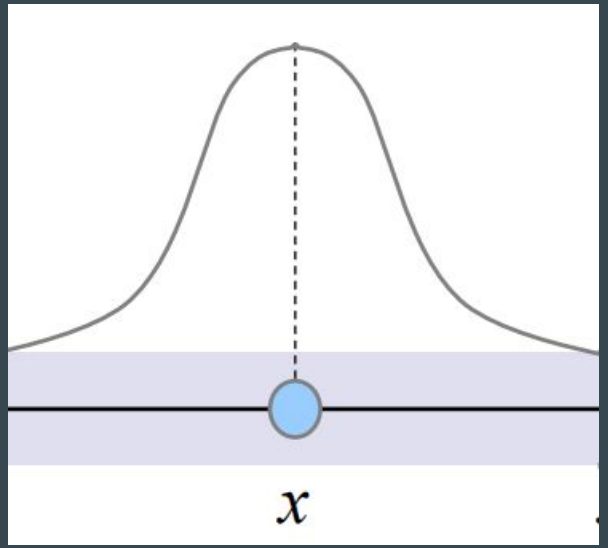




Mutation Operations

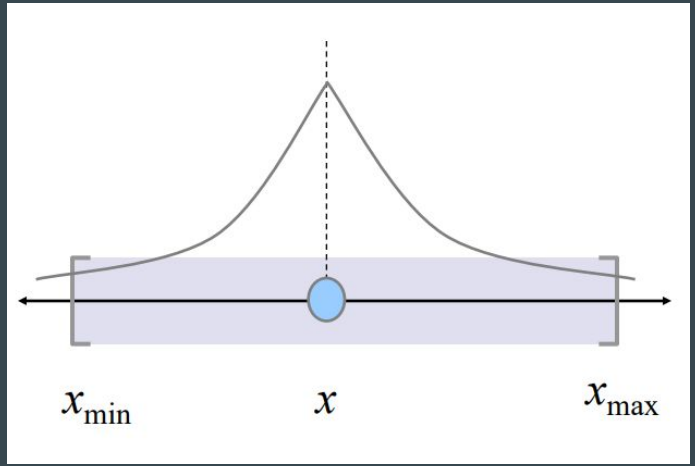
- **Random Mutation:** A new solution within entire range or in vicinity of original solution is generated.

- **Normally Distributed Mutation**
  - $x_{new} = x + N(0,\sigma^2)$



Mutation Operations

- **Non-Uniform Mutation**
  - $x_{new} = x + \tau(x_{max} - x_{min})(1 - r^{(1 - t/tmax)^b})$
    - τ takes -1 or 1, each with a probability of 0.5
    - r is a random number in [0, 1]
    - t_{max} is the maximum number of generations
    - t is the current generation number
    - b is the design parameter
  - As the generation number increases, the mutated solutions are generated closer to original solutions.



Variants of GA

GAs based on Darwinian theory gain fitness by Crossovers and Mutations only. Some GAs are hybridized with ‘local search’. We have two types of these hybrid GAs based on two models:

- **Lamarckian Model**: Traits which an individual acquires in his/her lifetime can be passed on to its offspring. If a better chromosome (new trait) is found in the local neighbourhood, it becomes the offspring.
- **Baldwinian Model**: The chromosomes encode a tendency of learning beneficial behaviors. If a better chromosome is found in the local neighbourhood, it only assigns the improved fitness to the chromosome and does not modify the chromosome itself. The change in fitness signifies the chromosomes capability to “acquire the trait”.

Some Applications

- **Trip, Traffic and Shipment Routing**: Finding shortest routes for traveling. Timing to avoid traffic tie-ups and rush hours. Most efficient use of transport for shipping.
- **Automotive Design**: Finding combinations of best materials and best engineering designs to provide faster, lighter, more fuel efficient and safer vehicles.
- **Evolvable Hardware**: Electronic circuits are created by GA computer models that use stochastic operators to evolve new configurations from old ones. New configurations give new functionalities which were not present earlier.
- **Encryption and Code Breaking**: GAs are used to encrypt data as well as to break them. GAs search large solution spaces of ciphers for one correct decryption.

Simulation Problem

Function -Rosenbrock

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2 \right]$$

Min f(x) , for n = 50,

Population - 100,

Chromosome- -6.9111 -38.8846 -39.5293 14.2131 23.9449 -11.9096 -45.5470 22.3046 -14.4033 -35.9955

44.3997 18.4499. .....upto X_{50} -50 <X < 50

Fitness function - f(x),

Parents selections- Roulette Wheel Selection

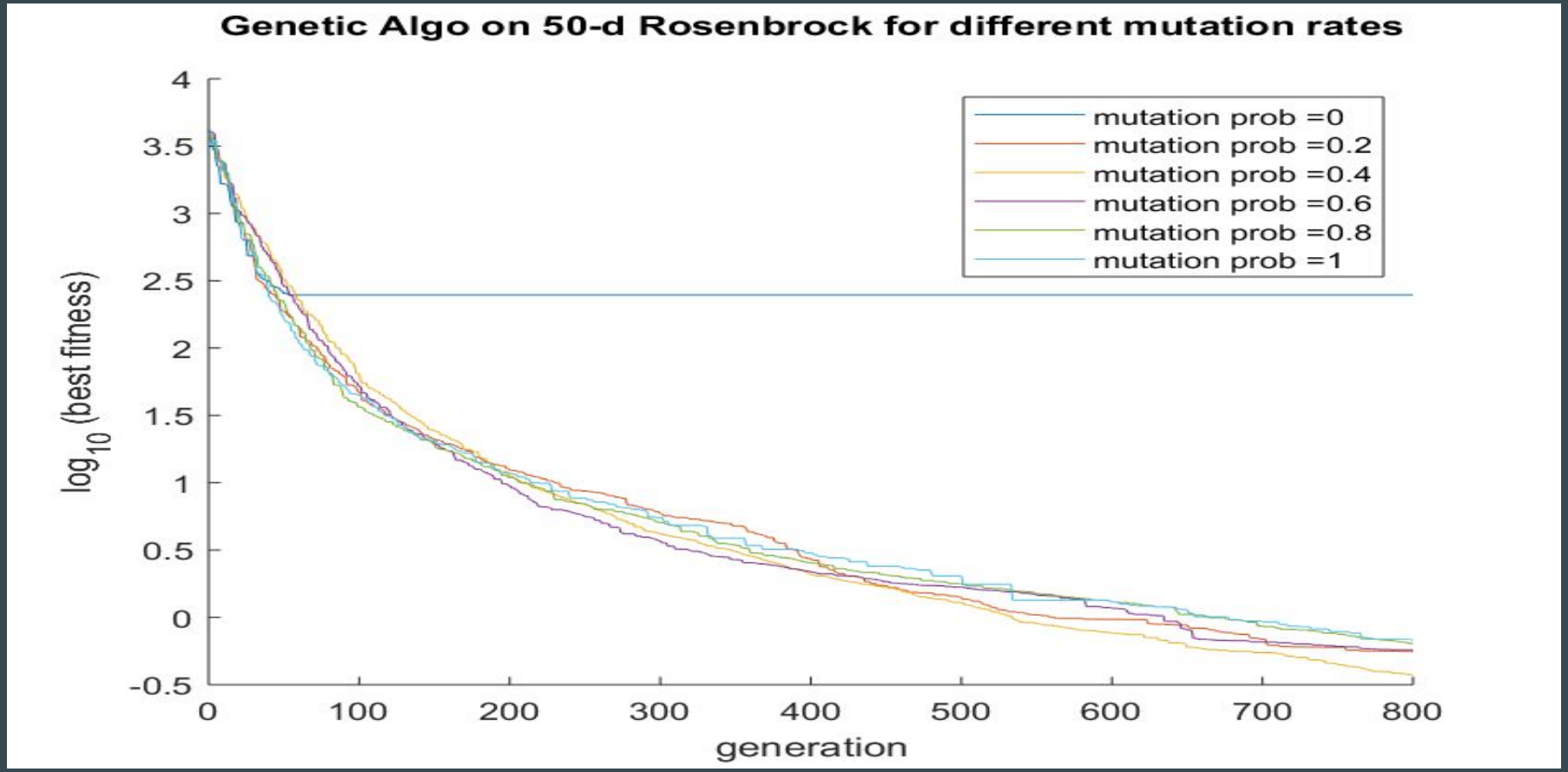
Crossover Operator- Uniform Crossover: We flip a coin for each chromosome to decide whether or not it'll be included in the offspring.

Mutation Operator- Random Resetting: In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

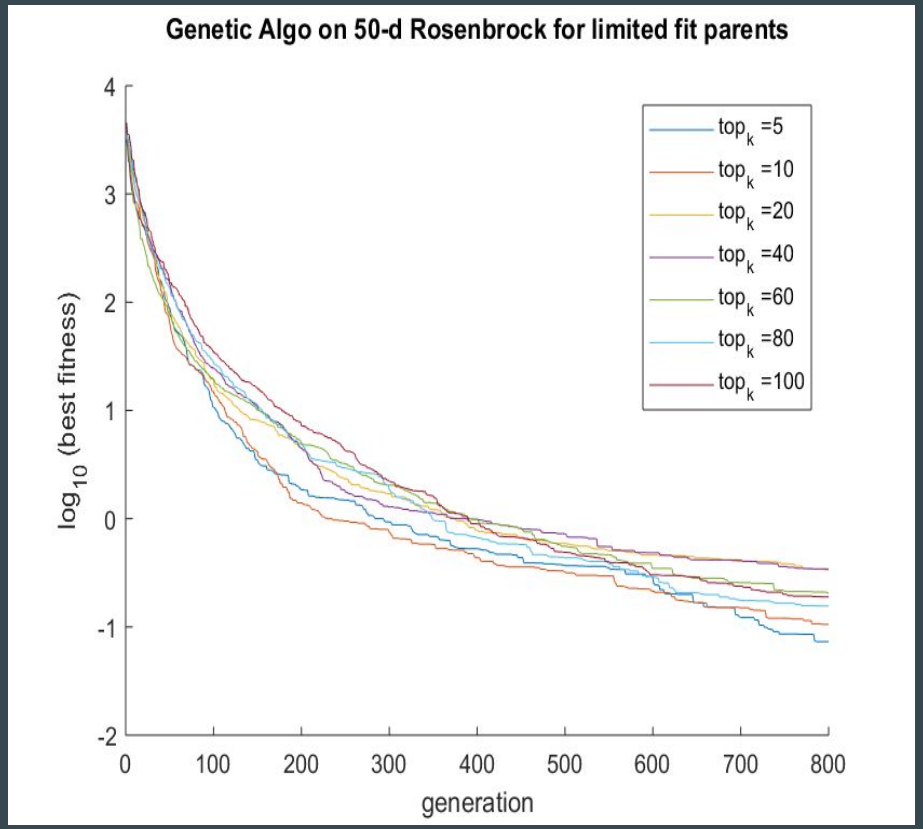
Survivor Selection- Fitness Based Selection

Termination Criteria- upto 1000 iteration.

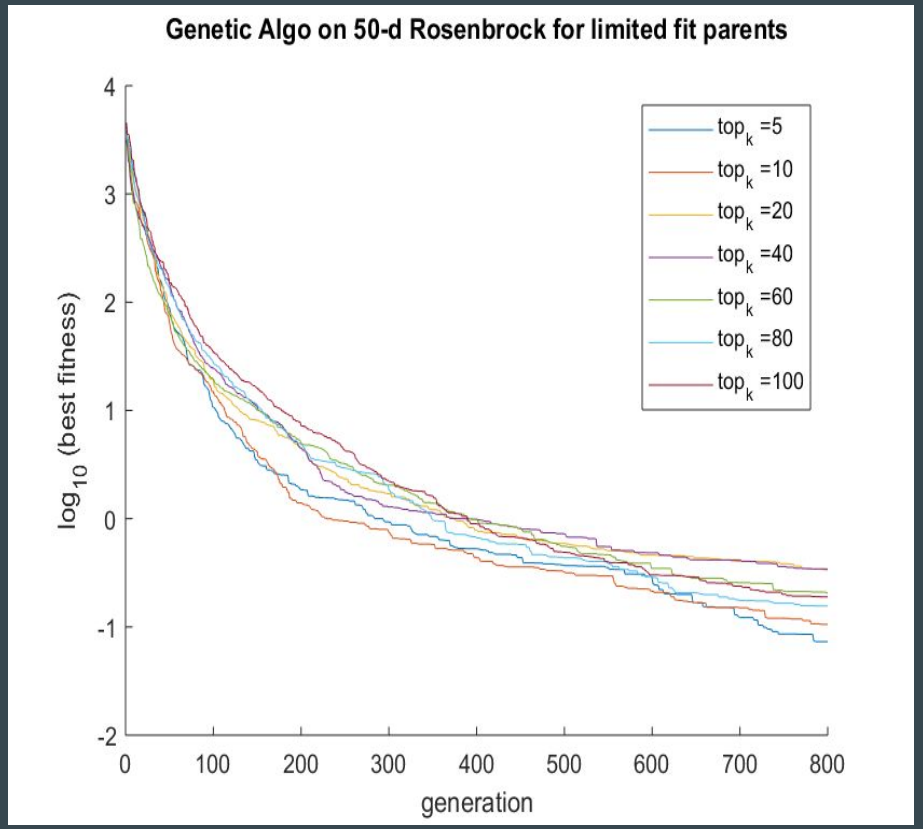
Simulation Results

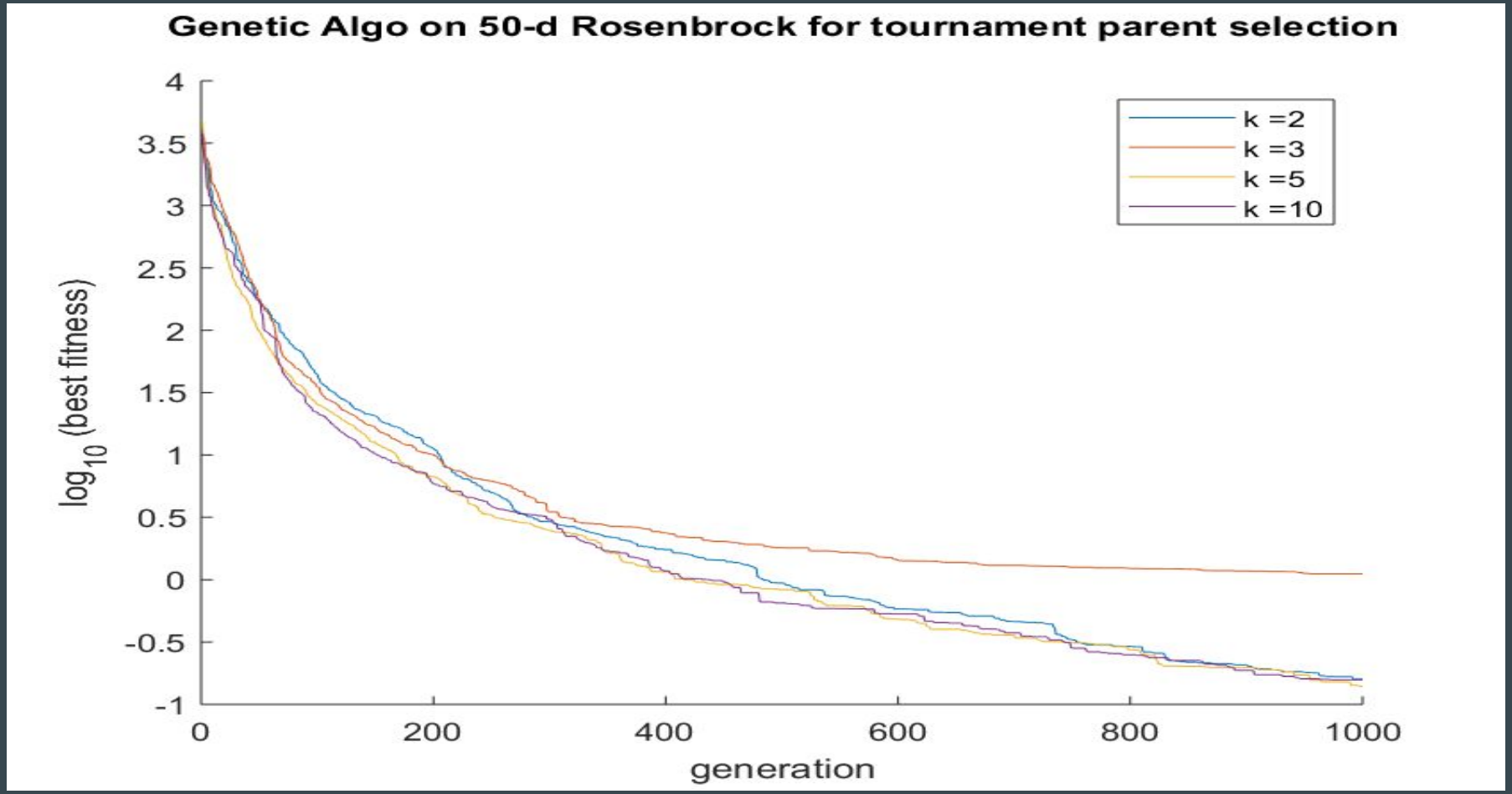


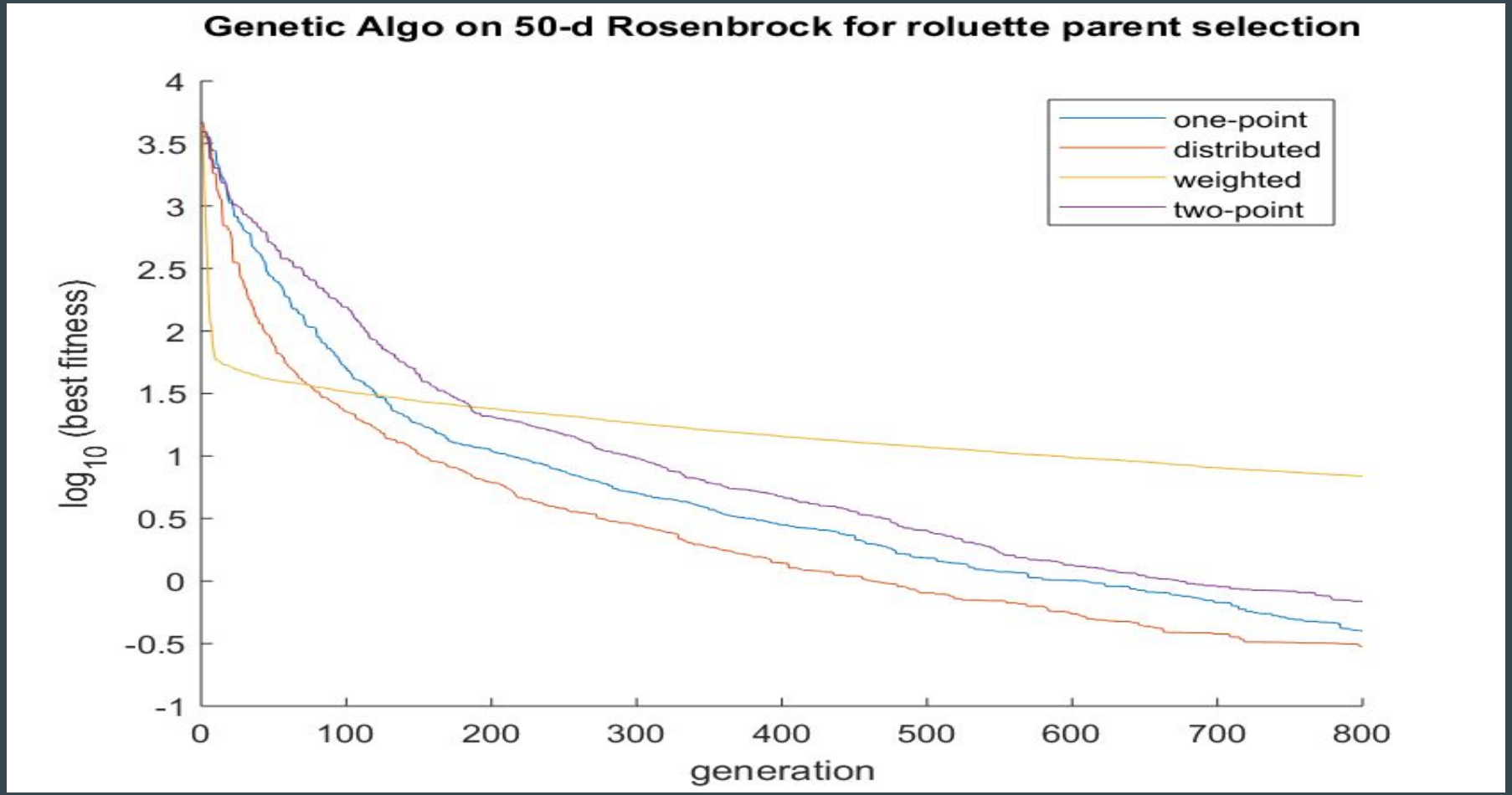
For Roulette selection of parents

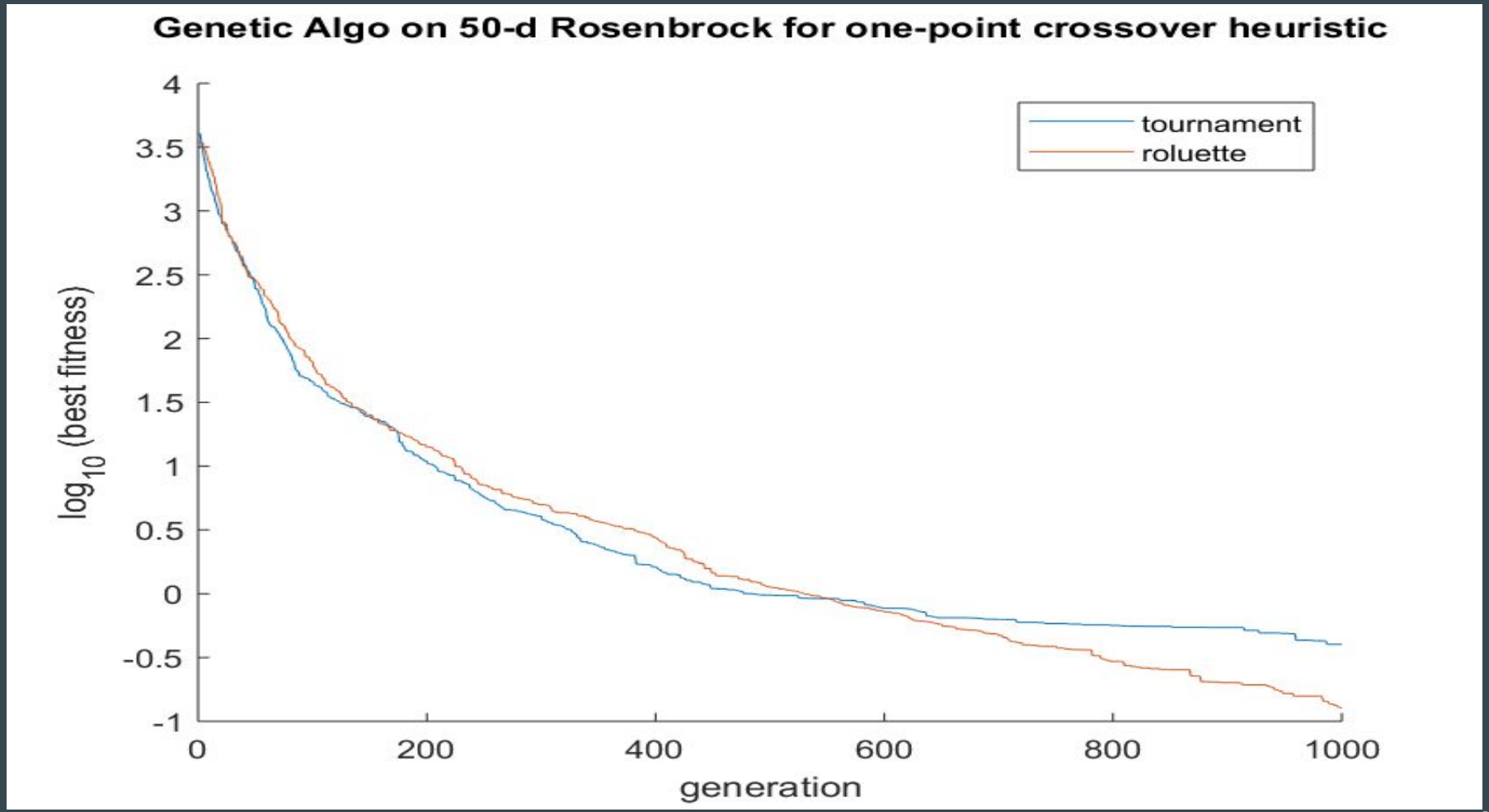


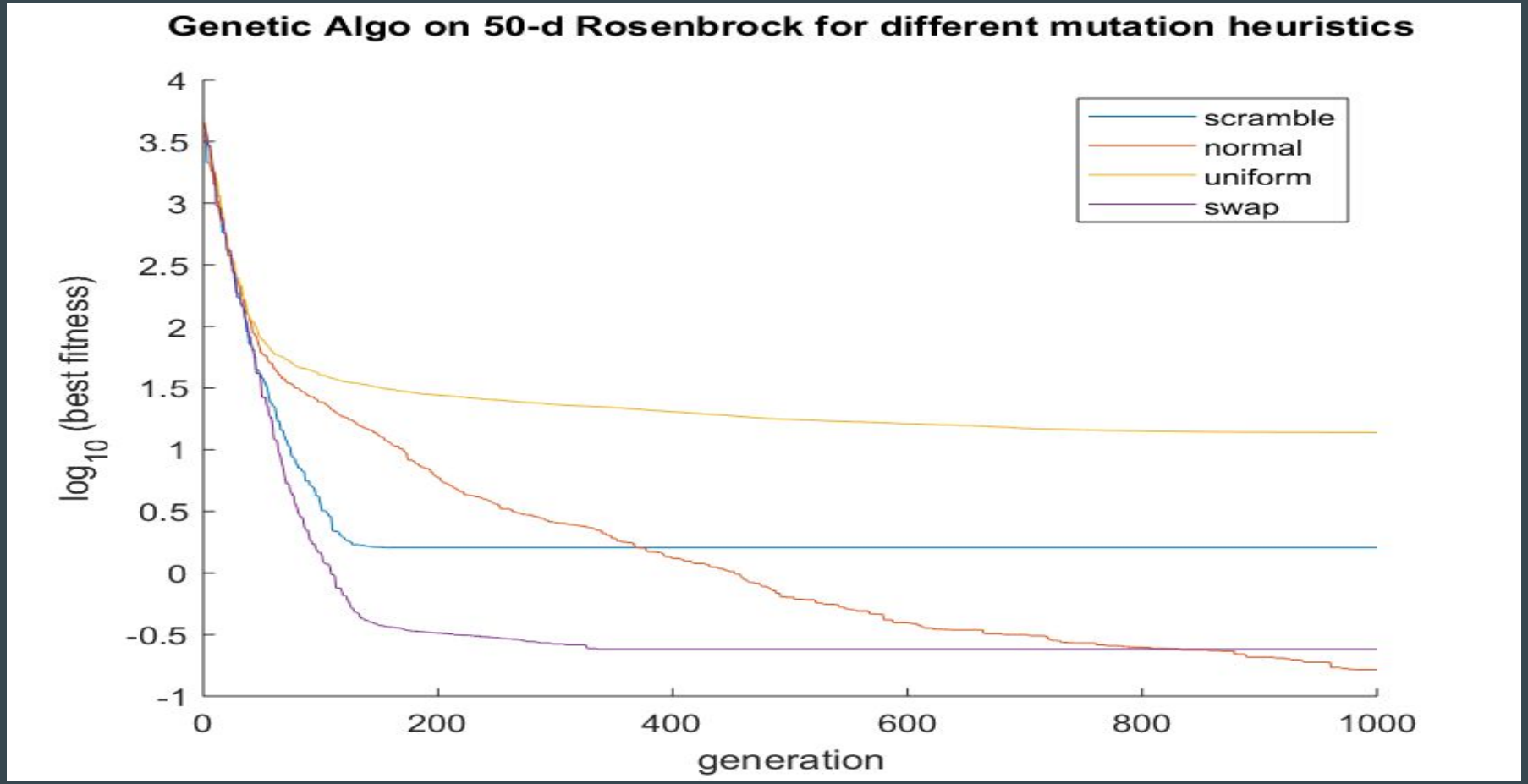
For tournament selection of parents

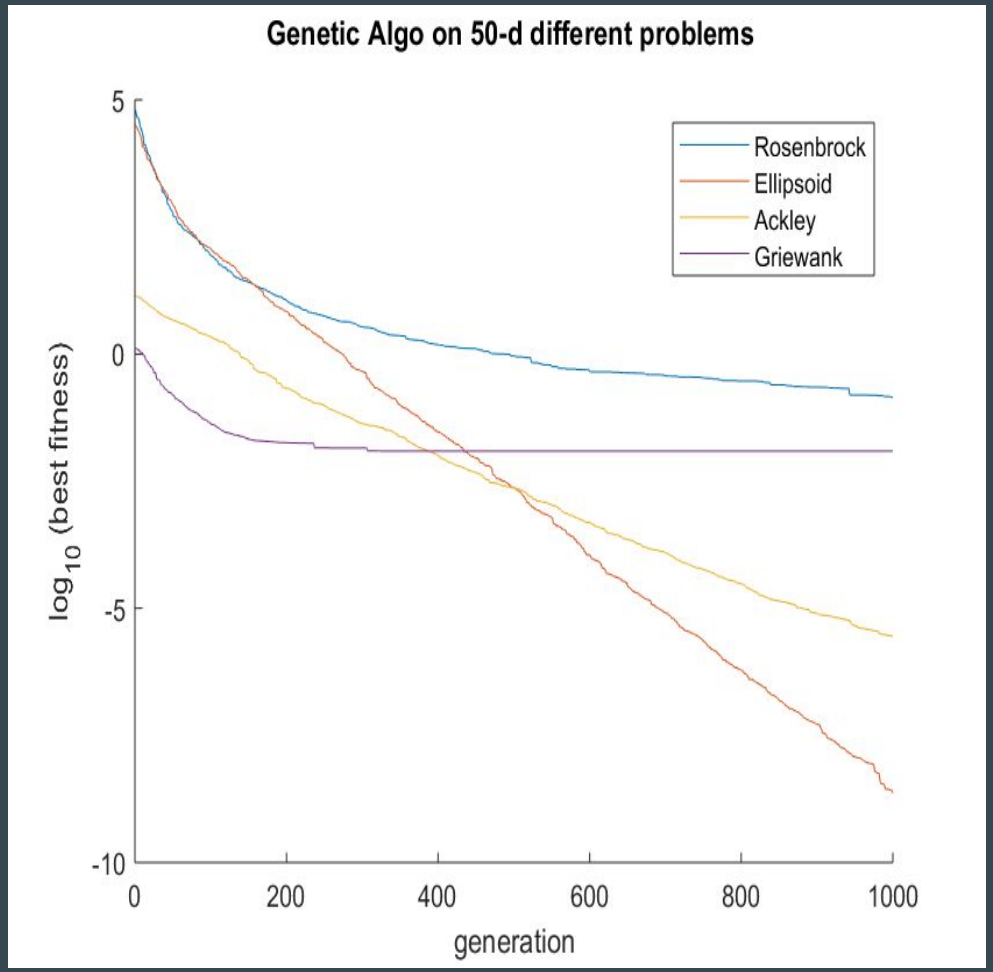



Genetic Algo on 50-d Rosenbrock for tournament parent selection
log10 (best fitness)
generation
k =2
k =3
k =5
k =10
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Genetic Algo on 50-d Rosenbrock for roluette parent selection
one-point
distributed
weighted
two-point
log10 (best fitness)
generation
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800


Genetic Algo on 50-d Rosenbrock for one-point crossover heuristic
tournament
roluette
log10 (best fitness)
generation
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Genetic Algo on 50-d Rosenbrock for different mutation heuristics
scramble
normal
uniform
swap
log10 (best fitness)
generation
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000



$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Ackley function, a=20, b=0.2, d=50
minima at origin =0

$$1 + \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Griewank function, minima at origin =0

Computation times for GA compared to quadprog/CVX :

106.05997 secs compared to 26.39482 secs

Discussions

Here, we tested our algorithm for optimal performance primarily on **Rosenbrock**. We simulated tournament and roulette heuristics for parent selection, while limiting chromosomes performing cross-over, ranked by fitness. We implemented single and double-point, uniform and blend crossovers with normally & uniformly distributed point-mutations, scramble and swap mutations.

We performed our analysis for optimal convergence for Rosenbrock, an ellipsoid, Griewank and Ackley functions (unconstrained) and for three constrained optimization problems using **merit functions**. In almost all cases, we converged to the optimal solution within a specified tolerance. For the **Griewank** function, we reach some saturation (extremely slow convergence) in some cases, as the algorithm is sensitive to initialization.

Crossover phase helps in **exploitation** of existing solutions, while mutation helps in **exploration** and helps in preventing suboptimal saturation. High rates of mutation leads to **random search**.

Advantages of GAs

- Does not require any **derivative information** (which may not be available for many real-world problems).
- Is faster and more **efficient** as compared to the traditional methods.
- Optimizes both **continuous** and **discrete** functions and also **multi-objective** problems
- Useful when the **search space is very large** and there are a large number of parameters involved.

Existing Software : **GeneHunter**

Limitations of GAs

- GAs are **not suited for all** problems, especially problems which are simple and for which derivative information is available.
- **Fitness value is calculated repeatedly** which might be computationally expensive for some problems.
- Being stochastic, there are **no guarantees on the optimality** or the quality of the solution.
- If not implemented properly, the GA **may not converge** to the optimal solution.
-

References

1.https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

2.https://engineering.purdue.edu/~sudhoff/ee630/Lecture04.pdf?fbclid=IwAR2vo_PQmhixnWIqWeSglZqlBiTJY0X63jXuNhFP-hucTu27tsJSalUXED0

3.https://en.wikipedia.org/wiki/Test_functions_for_optimization

4.https://en.wikipedia.org/wiki/Griewank_function

5.https://www.sfu.ca/~ssurjano/ackley.html

Thank You.