Computational Intelligence
(Part I, Fall 2021)

Instructor: **Dr. Yanan Sun**

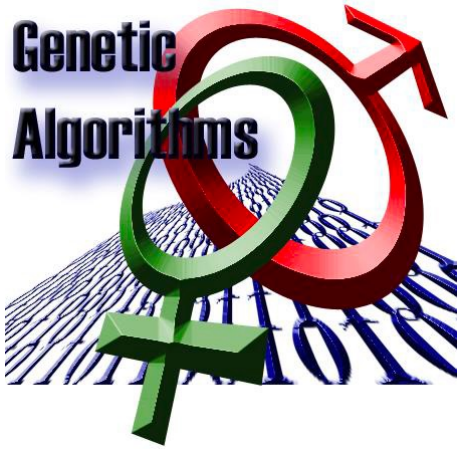
College of Computer Science

ysun@scu.edu.cn

https://yn-sun.github.io

GENETIC ALGORITHM

. . . . . .

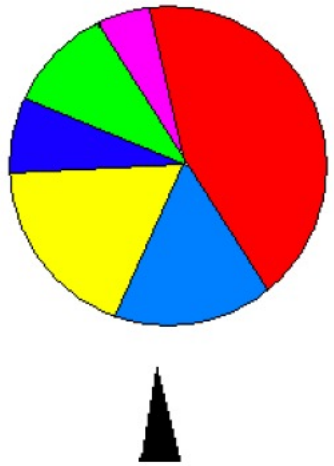


Selection Strategy

- We want to have some way to ensure that better fitted individuals have a better chance of being chosen for reproduction than poorly fitted ones.
- This will give us selection pressure which will drive the population forward.
- On the other hand, we have to be careful to give less good individuals at least some chance of being parents -they may include some useful genetic material.
- Risk of loss diversity

Roulette Wheel Selection

- Parents are selected according to their fitness. The better the chromosomes are, the more chances they are to be selected.
- The size of the section in the roulette wheel is proportional to the value of the fitness function of every chromosome -the bigger the value is, the larger the section is.
- Better (fitter) individuals have:
  - more space
  - more chances to be selected



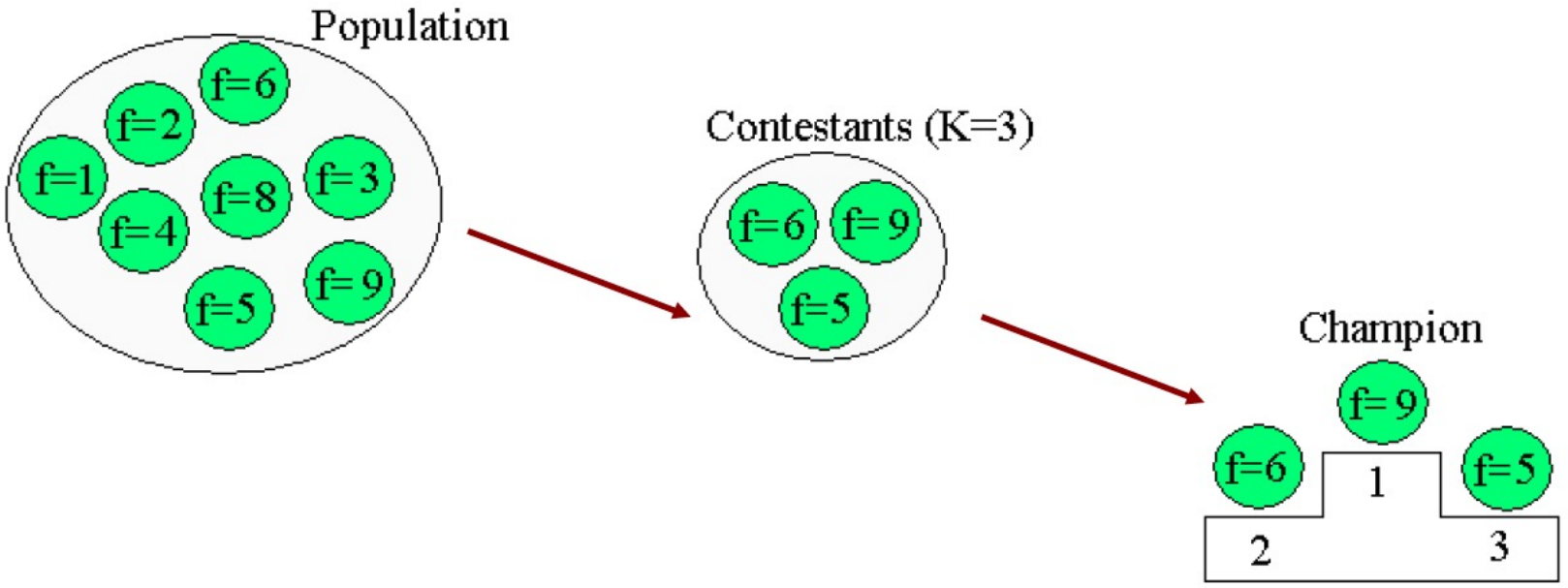
Fitness Proportionate Selection

Disadvantages:

- Danger of premature convergence because outstanding individuals take over the entire population very quickly.
- Low selection pressure when fitness values are near each other.

Tournament Selection

- Select k random individuals, without replacement
- Take the best
  - K is called the size of the tournament



Crossover vs. Mutation

- Crossover
  - modifications depend on the whole population
  - decreasing effects with convergence
  - exploitation operator
- Mutation
  - mandatory to escape local optima
  - exploration operator
- GA emphasizes crossover

Exploration vs. Exploitation

- Exploration
  - = sample unknown regions
  - Too much exploration = random search, no convergence
- Exploitation
  - = try to improve the best-so-far individuals
  - Too much exploitation = local search only convergence to a local optimum

Replacement Strategy

- The selection pressure is also affected by the way in which we decide which members of the population to eliminate in order to make space for the new individuals.
- We can use the stochastic selection methods in reverse, or there are some deterministic replacement strategies.
- We can decide never to replace the best in the population: elitism.

Stopping Criteria

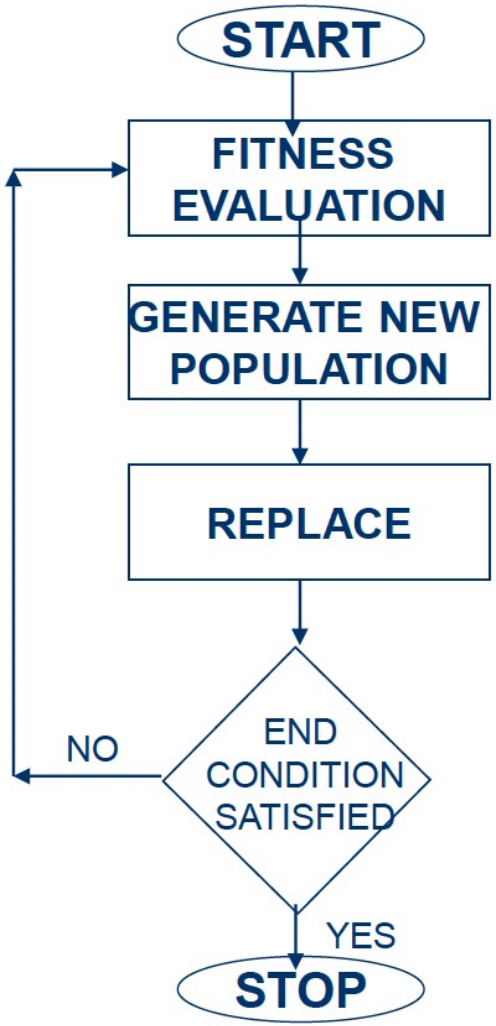
- The optimum is reached !!!!!
- Limit on CPU resources
- Maximum number of evolution generations
- Maximum number of fitness evaluations
- Limit on the user's patience
- After some generations without improvement

Performance

- Never draw any conclusion from a single run
  - Use statistical measure (average, median) (Box plot)
  - From a sufficient number of independent runs (30-50 minimum)
- From the application point of view
  - Design perspective
    - ➢ Find a very good solution at least once
  - Production perspective
    - ➢ Find a good solution at almost every run
- "What you test is what you get", don't tune algorithm performance on toy data and expect it to work with real data

Flowchart

1. Generate random population of N chromosomes (feasible solutions for the problem)
2. Evaluate the fitness f(x) of each chromosome x in the population
3. Create a new population by repeating following steps until the new population is complete by means of selection and crossover or mutation
4. Replace unfit individuals in old population by new off springs
5. If the end condition is satisfied, stop, and return the best solution in current population
6. else Go to step 2



- Applications in
  - Traffic control
  - VLSI design
  - Deadlock prevention
  - Parallel memory storage schemes
  - Sensor deployment
  - And etc.

Problem Issues in GA

- Population size
- Binary representation vs. real representation
- Population initialization
- Noisy fitness function
- Stochastic fitness function (or dynamic environment)
- Fitness inheritance and fitness approximation
- Selection/ranking strategy
- Crossover/Recombination operator
- Mutation operator
- Replacement strategy
- Stopping criteria
- Elitism strategy
- Benchmark test functions
- Exploration vs. exploitation dilemma
- Constraint handling
- Diversity promotion
- Population management