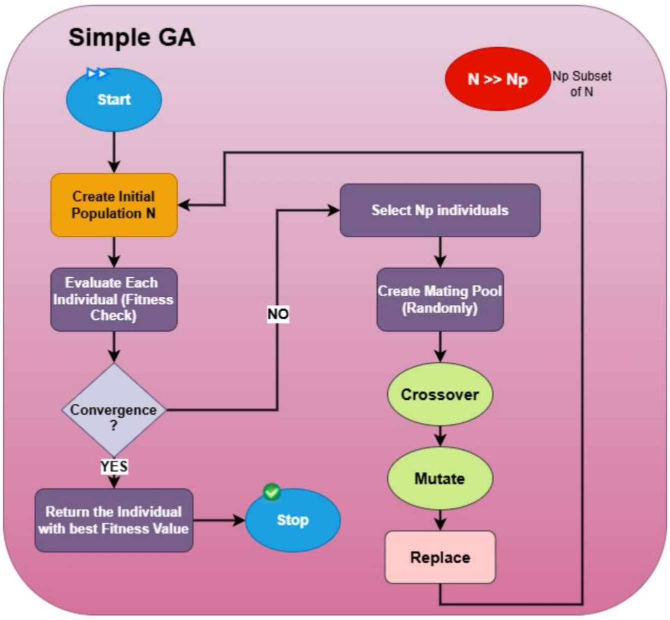
03. Simple Genetic Algorithm (GA)

Saturday, September 13, 2025 3:53 PM



- Initialize the Population
- Calculate the Fitness of each individual in population

Driving force of Evolutionary Algorithms just like, driving force for Neural networks (error), driving force of reinforcement learning (rewards, and punishment)

- While stopping criterion not satisfied
  - Select parents
  - Perform crossover ---> offsprings
  - Apply mutation
  - Calculate fitness

? **What are the shortcomings of GAs?**

- They are slow (test **many solutions over many generations**, which takes time.)
- Fitness function may not be easily designed

? **What is the population size? (or how many chromosomes in one generation/iteration.)**

1. Too many: GA extremely sluggish (becomes **slow** because it has to test a huge number of solutions.)
2. Too few: not many possibilities for mating (crossover)

Population size is like the number of students in a class.
- If the class is **too big** → teaching is **slow and inefficient**.
- If the class is **too small** → there's **little diversity of ideas** when students work together.

The best learning happens with a **balanced class size** — just like GA works best with a balanced population.

- **Small population** = fast but may miss good answers.
- **Large population** = thorough but very slow.
- **Balanced size** gives the best trade-off.

? **What about crossover frequency? (**how often we mix parents to create new offspring.**)**

- **100% (always crossover):** Every child is made by mixing parents → lots of new solutions, but risk of losing good ones.
- **0% (never crossover):** Just copy parents → no new ideas, population may get stuck.
- **Middle ground (60–90%):** Most children are new (from crossover), but some good parents are kept unchanged → balance between **exploring new solutions** and **keeping the best ones**.

That's why many use around **80–90%** crossover rate.

? **What about mutation frequency?** (how often we should apply mutation)

- **Too often (50%)**: Every solution keeps changing randomly → it becomes almost random search, evolution can't "learn" properly.
  (Imagine you have a candidate solution that is almost perfect, but every time you try to improve it, you randomly change 50% of it. That's like trying to solve a puzzle but shuffling half of the pieces randomly every time — you **lose the progress you had made**.)
- **0% (never):** no change in offsprings
- **Rarely (1%):** just enough to explore new possibilities without destroying the good solution.

For instance, ---> Probability of mutation = P_mutation = 1 / 1000, ... typically [0.5% - 1%]

? How to select parents?

- Roulette Wheel Selection
- Tournament Selection
- Rank-Based Selection
- Random Selection

★ ▶ **Roulette Wheel Selection**

The idea is that individuals with **higher fitness** should have a **greater chance of being selected as parents**, but every individual still has **some chance**.

Mathematical Formulation
Suppose we have a population of N individuals.

- Fitness of individual i: f_i
- Total fitness of population: $F = \sum_{i=1}^{N} f_i$
- Selection probability of individual i: $p_i = \frac{f_i}{F}$

This means the probability of selecting an individual is proportional to its fitness relative to the whole population.

Cumulative probability distribution (for the "roulette wheel"): $C_i = \sum_{k=1}^{i} p_i$

To **select one parent**:
1. Generate a random number **r ⍰ [0,1].**
2. Select the first individual iii such that **Ci ≥ r**.

Example

Suppose we have **5 individuals**:

Individual	**Fitness (f_i)**
A	10
B	30
C	20

Step 1: Compute total fitness $F = \sum_{i=1}^{N} f_i$ = 10 + 30 + 20 + 25 + 15 = 100

Step 2: Compute probabilities

$$p_i = \frac{f_i}{F}$$

$$p_A = \frac{10}{100} = 0.10, \quad p_B = \frac{30}{100} = 0.30, \quad p_C = \frac{20}{100} = 0.20, \quad p_D = \frac{25}{100} = 0.25, \quad p_E = \frac{15}{100} = 0.15$$

	Fitness f_i
A	10
B	30
C	20
D	25
E	15

$$p_i = \frac{f_i}{F}$$

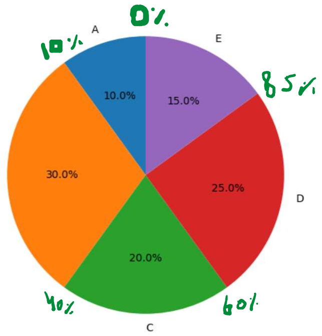
$$p_A = \frac{10}{100} = 0.10,\quad p_B = \frac{30}{100} = 0.30,\quad p_C = \frac{20}{100} = 0.20,\quad p_D = \frac{25}{100} = 0.25,\quad p_E = \frac{15}{100} = 0.15$$

So the probabilities are: A: 10%, B: 30%, C: 20%, D: 25%, E: 15%

Step 3: Build cumulative distribution $C_i = \sum_{k=1}^{i} p_i$

$C_A = 0.10,$
$C_B = 0.10 + 0.30 = 0.40$
$C_C = 0.40 + 0.20 = 0.60$
$C_D = 0.60 + 0.25 = 0.85$
$C_E = 0.85 + 0.15 = 1.00$

So intervals are:
- A: [0.00, 0.10)
- B: [0.10, 0.40)
- C: [0.40, 0.60)
- D: [0.60, 0.85)
- E: [0.85, 1.00]



Step 4: Simulate a spin
Suppose we generate random number **r = 0.73**.
- 0.73 falls in interval [0.60, 0.85], so **D is selected**.

If r = 0.05, we'd select **A**.
If r = 0.37, we'd select **B**.

Pros

1. **Simple and intuitive**
   - Just proportional to fitness → easy to understand.
   - Example: In our case, **B (fitness 30)** has **3× more chance** to be selected than **A (fitness 10)**.
2. **Every individual has a chance**
   - Even the weakest (A with 10 fitness) still has **10% chance**.
   - This helps keep **diversity** alive.

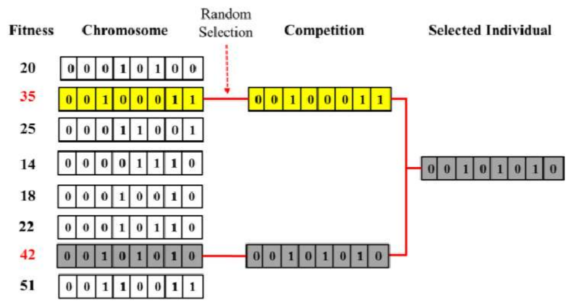
Cons

1. **Premature convergence if one fitness dominates**
   - Suppose one individual has fitness 95 and others share the remaining 5.
   - Then:
     - $p_{elite} = \frac{95}{100} = 0.95$
     - Almost always selects the same parent → loss of diversity → GA may get stuck in a local optimum.
2. **Scaling issues**
   - If fitness values are very large, selection becomes biased.
   - Example: With values [1000, 2, 1, 3], almost always select 1000 → weak exploration.

★ Tournament Selection

Instead of using probabilities directly (like roulette), tournament selection works by **competition**:
- Pick k individuals **at random** from the population.
- The **fittest** among those k wins and becomes a parent.
- Repeat as many times as needed (for multiple parents).

The parameter k controls **selection pressure**:
- Small k (e.g., 2) → more randomness, more diversity.
- Large k (close to population size) → almost always picks the global best, less diversity.



Example: Tournament Selection with N = 8 individuals
Step 1. Define Population & Fitness
Let's take **8 individuals** with these fitness values:

Individual	Fitness
A	12
B	25
C	7
D	30
E	18
F	40
G	22
H	15

Step 2. Rank Individuals
Sort by fitness (worst → best):

Rank	Individual	Fitness
1 (worst)	C	7
2	A	12
3	H	15
4	E	18
5	G	22
6	B	25
7	D	30
8 (best)	F	40

Step 3. Tournament Selection Process
Suppose **tournament size k = 3**.
- Pick 3 individuals randomly from the population.
- Select the one with **highest fitness** among them.
- Repeat for second parent.

Step 4. Mathematical Probability of Winning
The **probability** that an individual of rank r wins is:

$$P(r) = \left(\frac{r}{N}\right)^k - \left(\frac{r-1}{N}\right)^k$$

where:
- N = 8
- k = 3
- r=1…8 (1 = worst, 8 = best)

Step 5. Compute Probabilities
For N = 8, k = 3:
- $P(1) = \left(\frac{1}{8}\right)^3 - \left(\frac{0}{8}\right)^3 = 0.002$
- $P(2) = \left(\frac{2}{8}\right)^3 - \left(\frac{1}{8}\right)^3 = 0.015$
- $P(3) = \left(\frac{3}{8}\right)^3 - \left(\frac{2}{8}\right)^3 = 0.037$
- $P(4) = \left(\frac{4}{8}\right)^3 - \left(\frac{3}{8}\right)^3 = 0.059$
- $P(5) = \left(\frac{5}{8}\right)^3 - \left(\frac{4}{8}\right)^3 = 0.084$
- $P(6) = \left(\frac{6}{8}\right)^3 - \left(\frac{5}{8}\right)^3 = 0.103$
- $P(7) = \left(\frac{7}{8}\right)^3 - \left(\frac{6}{8}\right)^3 = 0.114$
- $P(8) = \left(\frac{8}{8}\right)^3 - \left(\frac{7}{8}\right)^3 = 0.118$

Pros

- **Easy to implement**: just pick k at random, choose max.
- **Adjustable pressure**: larger k → stronger bias toward best.

Cons

- If k is too small (e.g., k=2), weak individuals win too often → randomness.
- If k is too large (close to N), best always wins → loss of diversity.

- The **best individual (F, rank 8)** has ~11.8% chance of winning any tournament.
- Even weaker individuals still have a chance — e.g., rank 4 (E) has ~5.9%.
- This balances **selection pressure**: fitter individuals are more likely to win, but weaker ones are not completely excluded.