Tutorial on NSGA-II (Non-Dominated Sorting Genetic Algorithm – II)

1. Motivation and Background

Traditional Genetic Algorithms (GAs) are designed for **single-objective optimization**. However, many real-world problems involve multiple conflicting goals — for example:

- Minimize cost while maximizing performance.
- Minimize fuel use while maximizing power output.
- Minimize latency while maximizing throughput.

The challenge: there is no single "best" solution, but rather a set of trade-off solutions forming the **Pareto front**.

NSGA-II is a popular evolutionary algorithm developed by Deb et al. (2002) to efficiently find and maintain a diverse set of Pareto-optimal solutions. It is fast, elitist, and maintains diversity using the **crowding distance** concept.

—

2. Intuitive Analogy

Imagine you are selecting a team of players:

- You want players who are both strong and fast.
- Some players are strong but slow; others are fast but weaker.

A good selection keeps:

- Players who represent the full range of trade-offs (some very strong, some very fast, some balanced).
- The best ones within each category.

That's what NSGA-II does — it maintains the **best trade-offs** (non-dominated solutions) and keeps them **well spread out** using **crowding distance**.

—

3. Key Ideas Behind NSGA-II

1. **Fast Non-Dominated Sorting:** Classify solutions into different Pareto fronts (ranks) — Front 1 (best), Front 2 (next best), etc.

2. **Elitism:** Always keep the best solutions (the non-dominated ones) in the next generation.

3. **Crowding Distance:** Maintain diversity within each front by preferring more isolated solutions.

—

4. Steps of the NSGA-II Algorithm

1. **Initialization:** Generate an initial population P_0 of N random solutions.

2. **Evaluation:** Compute objective values $f_1(x), f_2(x), \ldots, f_M(x)$ for each solution.

3. **Non-Dominated Sorting:** Divide the population into levels (Front 1, Front 2, etc.). - Front 1: all solutions not dominated by any other. - Front 2: solutions dominated only by those in Front 1. Continue until all solutions are assigned a front number (rank).

4. **Crowding Distance Calculation:** For each front, calculate the crowding distance as discussed earlier to preserve spread.

5. **Selection:** Use a **binary tournament selection** based on two criteria:

   (a) Lower rank (better Pareto front).

   (b) Higher crowding distance (more isolated).

6. **Crossover and Mutation:** Create offspring population Q_t from selected parents using GA operators.

7. **Combine Populations:** Merge parent and offspring populations: $R_t = P_t \cup Q_t$.

8. **Elitist Sorting:** Sort R_t using non-dominated sorting again. Fill the next generation P_{t+1} with best fronts until you reach size N. If a front cannot fit completely, choose the solutions with the highest crowding distance.

9. **Termination:** Repeat from Step 3 until the maximum number of generations is reached.

—

5. Mathematical View (Simplified)

Each individual has two key attributes:

Rank = Level of non-domination (1 = best), Crowding Distance = Measure of isolation in objective

The selection rule is:

$$x_i \text{ is preferred to } x_j \text{ if } \begin{cases} \text{Rank}(x_i) < \text{Rank}(x_j), & \text{or} \\ \text{Rank}(x_i) = \text{Rank}(x_j) \text{ and } d_i > d_j. \end{cases}$$

This ensures both convergence (better rank) and diversity (larger distance).

—

6. Illustrative Example (Two Objectives)

Suppose we are optimizing a design for:

- f_1: *Cost* () – minimize.
- f_2: *Emission* (kg) – minimize.

You start with a random population of designs.

Step 1: Evaluate cost and emission for each design. **Step 2:** Sort into Pareto fronts — the best trade-offs (low cost and low emission) form Front 1. **Step 3:** Compute crowding distances — keep solutions that spread evenly along the front. **Step 4:** Generate new designs (offspring) using crossover and mutation. **Step 5:** Combine parent + offspring $\rightarrow$ keep best and most diverse solutions.

After several generations, the Pareto front converges to a smooth curve of trade-offs between cost and emission.

—

7. Intuitive Visualization

You can visualize NSGA-II as a gardener maintaining a flower bed:

- He removes weeds (dominated solutions).
- Keeps the strongest plants (non-dominated ones).
- Ensures plants are evenly spaced (using crowding distance).

Over generations, the garden (Pareto front) becomes both healthy and beautiful — this is convergence + diversity in optimization terms.

—

8. Pseudocode (Simplified)

```
Initialize population P0
Evaluate objectives for all individuals
t = 0
while (termination condition not met):
    Rt = Pt  Qt # Combine parent and offspring
    Perform fast non-dominated sorting on Rt
    Pt+1 = empty
    for each front Fi in Rt:
        if |Pt+1| + |Fi| <= N:
            calculate crowding distance for Fi
            add Fi to Pt+1
        else:
            sort Fi by descending crowding distance
            fill Pt+1 until it reaches N
            break
    Apply binary tournament selection using rank + distance
    Generate offspring Qt+1 using crossover and mutation
    t = t + 1
return final Pareto front (best non-dominated set)
```

—

9. Advantages of NSGA-II

- Maintains a diverse set of Pareto-optimal solutions.
- Uses elitism — best solutions are always preserved.
- Fast non-dominated sorting reduces complexity to $O(MN^2)$.
- Works well for two or three objectives (NSGA-III extends it for many).

—

10. Limitations and Extensions

- May struggle when the number of objectives is very high (> 3).
- Requires careful tuning of population size and mutation rate.
- NSGA-III and MOEA/D are improved versions for many-objective problems.

—

11. Key Takeaways

- NSGA-II finds a set of trade-off solutions (Pareto front), not a single best.
- Uses