Multi Objective Dung Bettle Optimization algorithm

Step 1:- Initialize Population

* Randomly initialize a population 𝑃 0 P 0 ​ of size 𝑁 N for the dung beetles.
* Initialize an empty archive 𝑃 archive P archive ​ to store non-dominated solutions.
* Each dung beetle 𝑖 i has a position 𝑥 𝑖 x i ​ and will have corresponding fitness values for both objectives 𝑓 1 ( ) f 1 ​ (x i ​ ) and 𝑓 2 ( 𝑖 ) f 2 ​ (x i ​ ).

Step 2. Evaluate Fitness

* For each beetle 𝑖 i, compute the two objective functions: 𝑓 1 ( 𝑥 𝑖 ) , 𝑓 2 ( 𝑥 𝑖 ) f 1 ​ (x i ​ ),f 2 ​ (x i ​ )
* Store these objective values.

Step 3. Non-Dominated Sorting (NSGA-II)

* Perform non-dominated sorting on the population 𝑃 𝑔 P g ​ to classify solutions into Pareto fronts 𝐹 1 , 𝐹 2 , … 𝐹 𝑛 F 1 ​ ,F 2 ​ ,…F n ​ .
* Pareto front 𝐹 1 F 1 ​ contains non-dominated solutions with respect to both objectives.
* Use these Pareto fronts to rank individuals.

Step 4:- Crowding Distance

* Calculation For each front 𝐹 𝑖 F i ​ , calculate the crowding distance (from NSGA-II) to maintain diversity in the Pareto front. Individuals with a higher crowding distance will be given priority for selection to maintain spread across the Pareto front.

Step 5:- Archive Maintenance (SPEA2)

* Maintain an archive 𝑃 archive P archive ​ to store non-dominated solutions.
* If 𝑃 archive P archive ​ becomes too large, apply a truncation mechanism (based on crowding distance) to reduce it back to size 𝑁 N.

Step 6:- Update Position of Dung Beetles

* For each beetle 𝑖 i in the population:
* If 𝑖 i is a ball-rolling dung beetle: Update the position based on its previous and worst-performing solution: 𝑥 𝑖 𝑡 + 1 = 𝑥 𝑖 𝑡 + 𝑎 ⋅ 𝑘 ⋅ ( 𝑥 𝑖 𝑡 − 1 ) + 𝑏 ⋅ Δ 𝑥 x i t+1 ​ =x i t ​ +a⋅k⋅(x i t−1 ​ )+b⋅Δx

Where: Δ 𝑥 = ∣ 𝑥 𝑖 𝑡 − 𝑥 worst ∣ Δx=∣x i t ​ −x worst ​ ∣

The worst solution is determined by the objective functions 𝑓 1 f 1 ​ and 𝑓 2 f 2 ​ .

* If 𝑖 i is a breeding dung beetle:
* Update the position based on the best-performing beetle: 𝑥 𝑖 𝑡 + 1 = 𝑥 pbest + 𝑟 1 ⋅ ( 𝑥 𝑖 𝑡 − 𝐿 𝑏 ∗ ) + 𝑟 2 ⋅ ( 𝑥 𝑖 𝑡 − 𝑈 𝑏 ∗ ) x i t+1 ​ =x pbest ​ +r 1 ​ ⋅(x i t ​ −L b ∗ ​ )+r 2 ​ ⋅(x i t ​ −U b ∗ ​ )

The best solution is found from Pareto front 𝐹 1 F 1 ​ , based on non-dominated sorting.

* If 𝑖 i is a small dung beetle:
* Update the position based on the best and worst solutions from the Pareto front: 𝑥 𝑖 𝑡 + 1 = 𝑥 𝑖 𝑡 + 𝑔 1 ⋅ ( 𝑥 𝑖 𝑡 − 𝐿 𝑏 𝑏 ) + 𝑟 3 ⋅ ( 𝑥 𝑖 𝑡 − 𝑈 𝑏 𝑏 ) x i t+1 ​ =x i t ​ +g 1 ​ ⋅(x i t ​ −L b b ​ )+r 3 ​ ⋅(x i t ​ −U b b ​ )
* If 𝑖 i is a thieving dung beetle:
* Update the position by moving towards the best and personal best solutions: 𝑥 𝑖 𝑡 + 1 = 𝑥 gbest + 𝑆 ⋅ 𝑟 4 ⋅ ( ∣ 𝑥 𝑖 𝑡 − 𝑥 pbest ∣ + ∣ 𝑥 𝑖 𝑡 − 𝑥 gbest ∣ ) x i t+1 ​ =x gbest ​ +S⋅r 4 ​ ⋅(∣x i t ​ −x pbest ​ ∣+∣x i t ​ −x gbest ​ ∣)

Step 7:- Selection for Next Generation (NSGA-II)

* Select individuals for the next generation using tournament selection based on Pareto dominance and crowding distance. The solutions with better ranking in the Pareto fronts and higher crowding distance will have higher selection probability.

Step 8:- Crossover and Mutation (NSGA-II)

* Apply crossover and mutation operators on the selected individuals to generate offspring. This ensures exploration of the search space and variation in the population.

Step 9:- Update the Archive

* After each generation, update 𝑃 archive P archive ​ by adding new non-dominated solutions and removing less desirable ones to keep its size at 𝑁 N.

Step 10:- Termination

* Repeat the process until a stopping criterion is met (e.g., a maximum number of generations 𝑇 max T max ​ , or convergence).
* Once the stopping criterion is reached, return the solutions from the archive 𝑃 archive P archive ​ , representing the Pareto front.

Initialize population P\_g of size N

Initialize archive P\_archive as empty

t = 0

while t < T\_max do:

    # 1. Evaluate Fitness for both objectives f\_1 and f\_2

    for each beetle i in P\_g:

        Evaluate f\_1(x\_i) and f\_2(x\_i)

    # 2. Apply Non-dominated Sorting

    F\_1, F\_2, ..., F\_n = NonDominatedSorting(P\_g)

    # 3. Calculate Crowding Distance for each front

    for each front F\_i:

        CalculateCrowdingDistance(F\_i)

    # 4. Update Position for each dung beetle type

    for each beetle i in P\_g:

        if i is ball-rolling beetle:

            Update position using gbest and gworst from Pareto fronts

        elif i is breeding beetle:

            Update position using personal best and boundaries

        elif i is small beetle:

            Update position based on gbest and crowding distance

        elif i is thieving beetle:

            Update position by stealing from best solutions in F\_1

    # 5. Update Archive

    Update P\_archive with non-dominated solutions

    Truncate archive if size exceeds N

    # 6. Selection (NSGA-II)

    P\_g+1 = TournamentSelection(P\_g, F\_1, ..., F\_n)

    # 7. Crossover and Mutation

    Q\_g+1 = CrossoverAndMutation(P\_g+1)

    # 8. Increment time step

    t = t + 1

# Return the Pareto-optimal front

Return P\_archive

\*\*Input\*\*:

- Problem Definition (Number of Variables, Objective Functions, Variable Ranges)

- Parameters (Population Size, Max Generations, Archive Size, Crossover Probability, Mutation Probability)

\*\*Output\*\*:

- Final Pareto Front (Set of Non-Dominated Solutions)

1. \*\*Initialize Population\*\*:

a. For each individual in the population:

i. Randomly generate position within variable ranges.

ii. Evaluate fitness using the given objective functions.

iii. Assign a random type (ball-rolling, breeding, small, thieving).

2. \*\*Initialize Archive\*\*:

- Set the archive as an empty list.

3. \*\*For\*\* generation = 1 to Max Generations \*\*Do\*\*:

a. \*\*Evaluate Population\*\*:

i. For each individual in the population:

- Re-evaluate fitness based on current position.

b. \*\*Update Archive\*\*:

i. Combine the current population and archive.

ii. Perform non-dominated sorting on the combined set.

iii. Select individuals for the archive using:

- Front-based selection.

- Crowding distance to maintain diversity.

iv. Ensure archive size does not exceed the predefined limit.

c. \*\*Update Positions\*\*:

i. For each individual in the population:

- If type == "ball-rolling":

Update position relative to the worst individual.

- If type == "breeding":

Update position relative to the best individual in the archive.

- If type == "small":

Update position considering both the best and worst individuals in the archive.

- If type == "thieving":

Update position relative to the global best individual in the archive.

- Clip position to remain within variable bounds.

- Recalculate fitness.

d. \*\*Generate Next Population\*\*:

i. Select individuals using non-dominated sorting and crowding distance.

ii. While the next population size is less than the population size:

- Randomly select two parents.

- Perform crossover with a certain probability to create offspring.

- Perform mutation with a certain probability on the offspring.

- Add offspring to the next generation population.

4. \*\*End For\*\*.

5. \*\*Return Pareto Front\*\*:

- Extract fitness values from the final archive.

- Return the set of non-dominated solutions as the Pareto front.