The five crossover operators mentioned—SBX (Simulated Binary Crossover), PCX (Parent Centric Crossover), SPX (Simplex Crossover), BLX-α (Blend Crossover Alpha), and DE/rand/1 (Differential Evolution/rand/1)—are techniques used in genetic algorithms and evolutionary computation to combine the genetic information of parent solutions to produce new offspring. Each operator has its unique mechanism and application scenarios, particularly in handling real-valued optimization problems. Here's a brief overview of each:

### 1. Simulated Binary Crossover (SBX)

- \*\*Description\*\*: SBX is designed to simulate the single-point crossover found in binary genetic algorithms but applied to real-valued parameters. It uses a probability distribution to decide how far the offspring will be from the parents in the parameter space, allowing for both exploration and exploitation.

The Simulated Binary Crossover (SBX) algorithm for continuous search spaces involves the following simplified steps:

1. \*\*Select Parents\*\*: Two parent solutions are chosen from

2. \*\*Determine Crossover Probability\*\*: Check if crossover will occur based on a predefined probability. If not, offspring are exact copies of parents.

3. \*\*Calculate Spread Factor (β)\*\*: For each pair of corresponding genes in the parent solutions, calculate a spread factor that determines how far apart the offspring will be from their parents. (how?)

4. \*\*Generate Offspring\*\*: Using the spread factor, generate two offspring solutions. The calculation ensures that offspring values are spread around the parent values, simulating the binary crossover effect.

To calculate the spread factor (β) in Simulated Binary Crossover (SBX) for each pair of corresponding genes in the parent solutions, follow these steps:

1. \*\*Calculate the Difference\*\*: Find the absolute difference between the two parent genes.

2. \*\*Determine Random Value\*\*: Generate a random number \(u\) between 0 and 1.

3. \*\*Calculate βq\*\*: Use the random value \(u\) to calculate the spread factor (\(β\_q\)) based on a predefined distribution and parameters. This involves specific equations that consider whether \(u\) is less than 0.5, leading to a calculation that ensures diversity in the offspring by controlling how much they deviate from their parents.

4. \*\*Use βq to Generate Offspring\*\*: Apply \(β\_q\) to interpolate (or extrapolate) between the parent genes to generate offspring genes. This is done by calculating the average value of the two parent genes and then adding or subtracting half of the product of \(β\_q\) and the difference between the parent genes.

These steps ensure that the offspring are a blend of their parents but with variations that allow for exploration of the search space in continuous domains.

- \*\*Application\*\*: SBX is widely used in genetic algorithms for problems where the decision variables are real numbers. It's effective in maintaining a balance between exploration of the search space and exploitation of the current solutions.

Interval schemata refer to patterns or structures within a solution space that are defined by specific ranges (intervals) for each variable or gene in the context of genetic algorithms. These patterns represent common traits or characteristics shared among potential solutions. The concept of respecting interval schemata, particularly in the context of the SBX (Simulated Binary Crossover) operator, means that when creating offspring, the crossover operation preserves these common ranges or characteristics found in the parent solutions. This approach ensures that the offspring inherit significant traits from their parents, contributing to the efficiency of the search process in evolutionary algorithms by maintaining useful genetic information across generations.

### 2. Parent Centric Crossover (PCX)

- \*\*Description\*\*: PCX is a real-coded crossover technique where the new offspring are generated closer to the parent solutions, with a focus on the centroid of the parents. It uses Gaussian distribution around the centroid to create offspring, promoting fine-grained search and exploitation.

The Parent-Centric Crossover (PCX) algorithm is a technique used in genetic algorithms to generate new offspring solutions from a group of parent solutions. Here's a simplified step-by-step explanation of the PCX algorithm, followed by a basic Python implementation:

### Step-by-Step Explanation

1. \*\*Selection of Parents\*\*: Choose a set of parent solutions from the current population based on their fitness scores.

2. \*\*Determine the Center of Mass\*\*: Calculate the center of mass (mean) of the selected parents to understand the central tendency of the solutions.

3. \*\*Selection of a Target Parent\*\*: Randomly select one parent from the chosen set to act as the target for generating offspring.

4. \*\*Calculation of Displacement Direction\*\*: Compute the direction vector from the center of mass towards the target parent. This vector guides the generation of offspring towards promising regions in the search space.

5. \*\*Generation of Offspring\*\*: Create new solutions (offspring) by perturbing the target parent along the calculated direction vector. The perturbation involves adding a scaled version of the direction vector and a random component orthogonal to it to maintain diversity.

- \*\*Application\*\*: PCX is particularly useful in evolutionary strategies where a more localized search is beneficial, especially in densely populated regions of the solution space, aiding in fine-tuning solutions.

### 3. Simplex Crossover (SPX)

- \*\*Description\*\*: SPX generates offspring by creating a simplex (a geometric figure) using parent solutions and then randomly sampling points within this simplex. This approach allows offspring to inherit features from all parents, encouraging diversity.

The Simplex Crossover (SPX) algorithm is a multi-parent recombination operator designed for real-coded genetic algorithms. It generates offspring by uniformly sampling from a simplex formed by \(m\) parent vectors. The key steps of the SPX algorithm can be summarized as follows:

1. \*\*Formation of a Simplex:\*\* A simplex in an \(n\)-dimensional space is formed by \(n+1\) points (parent vectors) that are not co-linear. For simplicity, consider a 3-parent SPX in a two-dimensional space, where the parents form a triangle (simplex).

2. \*\*Center of Mass Calculation:\*\* Calculate the center of mass (\(O\)) of the simplex formed by the parent vectors.

3. \*\*Expansion:\*\* Expand the simplex by a factor of \((1+\varepsilon)\) for each direction (\(X(j) - O\)), where \(\varepsilon\) is a control parameter for expansion, making the simplex larger or smaller.

4. \*\*Offspring Generation:\*\* Generate \(m\) offspring by uniformly picking vector values from the expanded simplex. This allows for a balance between exploration and exploitation in the search space.

The SPX algorithm is characterized by its independence from coordinate systems and its effectiveness on functions with multimodality and/or epistasis. It's simple and not time-consuming, making it a powerful tool for optimization in genetic algorithms.

- \*\*Application\*\*: SPX is effective in exploratory phases of the search, especially in multi-modal landscapes, as it allows offspring to inherit diverse traits from a group of parents, enhancing the algorithm's ability to explore various niches.

### 4. Blend Crossover Alpha (BLX-α)

- \*\*Description\*\*: BLX-α generates offspring by blending the genes of two parents with a specific parameter α that controls the extent of exploration beyond the parents' traits. Offspring can lie within a range that extends beyond the parents' values, determined by α.

The BLX-alpha (Blend Crossover Alpha) algorithm is a method used in genetic algorithms for real-coded genetic representations. It combines the features of parents to generate offspring by blending their characteristics. Here's a simplified step-by-step explanation:

1. \*\*Select Two Parents\*\*: Choose two parent solutions from the population that you want to combine.

2. \*\*Determine the Interval\*\*: For each parameter, identify the range (interval) between the corresponding values of the two parents.

3. \*\*Extend the Interval\*\*: Based on a user-defined parameter, alpha (α), extend the interval on both sides. For example, if alpha is 0.5, and the interval between the parents is [a, b], then the extended interval would be [a - 0.5\*(b-a), b + 0.5\*(b-a)].

4. \*\*Generate Offspring\*\*: For each parameter, randomly select a value within the extended interval to create a new offspring. This random selection ensures a blend of parental features with a chance of exploring new solutions outside the direct lineage.

5. \*\*Repeat\*\*: Generate as many offspring as needed by repeating the process with different pairs of parents.

This approach allows the algorithm to explore the search space more thoroughly by not only combining the exact traits of the parents but also considering the surrounding area in the parameter space, potentially leading to better solutions.

- \*\*Application\*\*: BLX-α is used for its simplicity and ability to explore both within and beyond the region defined by the parent solutions. It's suitable for real-valued problems where a balance between local search and exploration is needed.

### 5. Differential Evolution/rand/1 (DE/rand/1)

- \*\*Description\*\*: This operator is part of the Differential Evolution algorithm family, where a trial vector is generated by adding the weighted difference between two randomly selected individuals to a third individual. The "/rand/1" notation indicates that one difference vector is used, and the base vector is chosen randomly.

- \*\*Application\*\*: DE/rand/1 is adept at global optimization, especially in continuous, high-dimensional spaces. It's known for its simplicity and effectiveness in exploring the search space and exploiting the available genetic material.

Each of these crossover operators has unique characteristics that make them suitable for different kinds of optimization problems, especially in the context of multi-objective optimization where the diversity of solutions and convergence to the Pareto front are critical considerations.

The provided screenshots describe the adaptive crossover selection mechanism used in the Hybrid Multiobjective Evolutionary Algorithm (HMOEA). This mechanism dynamically selects one of multiple crossover operators during the evolutionary process to generate new solutions. Here’s a detailed breakdown of the process:

### Step 1: Update Selection Probability

- \*\*Evolving Extended Archive (EXA):\*\* Each time the Extended Archive (EXA), which stores nondominated solutions, is updated, the algorithm recalculates the selection probability for each crossover operator. This archive update signifies a change in the solution space, prompting a re-evaluation of operator effectiveness.

- \*\*Operator Index:\*\* Each crossover operator is assigned an index (for instance, `1` for BLX-α, `2` for SBX, etc.).

- \*\*Calculation of Selection Probability (`pi`):\*\* The selection probability for an operator is calculated as the number of solutions in the EXA that were generated using that operator, divided by the total number of solutions in the EXA (`|EXA|`). Formally, it is given by `pi = |{solutions in EXA generated by operator i}| / |EXA|`.

This selection probability is a measure of how much each operator has contributed to the current set of nondominated solutions, under the assumption that operators that have generated more solutions in the EXA may be more effective for the current state of the evolutionary process.

### Step 2: Select Crossover Operator using Roulette-Wheel Method

- \*\*Roulette-Wheel Selection:\*\* The algorithm uses a roulette-wheel selection method to choose an operator based on the calculated probabilities. This is a stochastic process where the chance of selecting an operator is proportional to its selection probability.

- \*\*Minimum Probability (`pmin`):\*\* To ensure diversity and prevent the algorithm from converging prematurely on one operator, each operator is given a minimum selection probability (`pmin`). If the calculated probability for an operator falls below this minimum, it is set to `pmin`, and the excess probability is redistributed among other operators, ensuring that all operators have at least a chance `pmin` of being selected.

- \*\*Adjusting Probabilities:\*\* If after setting an operator's probability to `pmin` it causes another operator's probability to exceed `1`, the excess is subtracted from the latter's probability. This ensures that the total probability across all operators sums to `1`.

- \*\*Ignoring Probability Calculation:\*\* If the EXA is not updated in a given generation, the selection probability calculation is ignored, presumably because no new information has been gained that would influence the effectiveness of the crossover operators.

### Summary of the Adaptive Mechanism

This adaptive mechanism ensures that the crossover operators used to create new solutions are selected based on their past performance, with a bias towards those that have recently contributed to improving the nondominated set of solutions. The use of a minimum selection probability guarantees that all operators continue to be used to some extent, maintaining diversity and preventing the stagnation of the search process. The roulette-wheel method provides a stochastic yet biased way to choose operators, ensuring that the selection is not entirely deterministic and allows for exploration.

Polynomial mutation is a technique used in evolutionary computing to introduce small, random changes in the offspring's genes to maintain genetic diversity in the population. It is particularly useful in real-valued genetic algorithms where each gene represents a real number. Polynomial mutation is favored because it allows for a more controlled and less disruptive mutation process compared to other methods, helping to explore the solution space more effectively.

### How Polynomial Mutation Works

1. \*\*Selection of Mutation Probability:\*\* Each gene has a predefined probability of undergoing mutation. This probability is typically low to ensure that mutations do not happen too often, preserving the integrity of promising solutions.

2. \*\*Mutation Distribution Index (\(η\_m\)):\*\* This parameter controls the shape of the distribution used to generate the mutation. A higher value for \(η\_m\) makes the distribution more peaky, concentrating mutations closer to the parent value, whereas a lower value spreads the mutations out, allowing for more significant changes.

3. \*\*Mutation Operation:\*\* For a selected gene to mutate, a random number \(δ\) is generated for that gene using a polynomial probability distribution. This \(δ\) is then used to calculate the mutated value of the gene. The calculation ensures that the mutated value is within the bounds of the gene's possible values and that mutations are more likely to be small changes, with larger changes being increasingly rare.