# Estimation of distribution algorithm

Estimation of Distribution Algorithms (EDAs), also known as Probabilistic Model-Building Genetic Algorithms (PMBGAs), are stochastic optimization methods that utilize probabilistic models to guide the search for optimal solutions. These algorithms start with a population of potential solutions, often generated randomly, and evaluate them using a fitness function to determine their quality.

The best-performing solutions are selected to build a probabilistic model that captures the distribution of these promising candidates. This model is then used to generate new solutions through sampling, allowing the algorithm to focus on areas of the solution space that are more likely to yield high-quality results.

Unlike traditional evolutionary algorithms that rely on variation operators like mutation and crossover to create new candidates, EDAs explicitly model the distribution of successful solutions. By continuously refining their probabilistic models based on successful candidates, EDAs improve their ability to find optimal or near-optimal solutions effectively.

## Univariate EDAs

Univariate EDAs assume that each variable (or parameter) in the problem is independent of others. They model the distribution of each variable separately, often using simple probability distributions like Gaussian or Bernoulli.

### Steps:

1. **Initialize** - Generate an initial random population of solutions.
2. **Select** - Identify a subset of high-quality solutions (based on fitness).
3. **Estimate** - For each variable, estimate its probability distribution from the selected subset.
4. **Sample** - Generate a new population by sampling from the updated distributions.
5. **Iterate** - repeat steps 2-4 until convergence.

### Pseudocode:

```
Initialize population P with random solutions
Repeat until termination criterion is met:
  Select subset S of best solutions from P
  For each variable X_i:
    Estimate distribution p(X_i) from S
  Generate new population P by sampling p(X_1), p(X_2), ..., p(X_n)
```

## Multivariate EDAs

Multivariate EDAs account for dependencies between variables. Instead of modeling each variable independently, they capture relationships using multivariate probability models such as Bayesian networks or Gaussian multivariate distributions.

**Steps:**

1. **Initialize** - Generate an initial random population.
2. **Select** - Identify the top-performing solutions.
3. **Estimate** - Learn a multivariate model (e.g., covariance matrix) that captures variable relationships.
4. **Sample** - Generate new solutions by sampling from the multivariate model.
5. **Iterate** - Repeat until a satisfactory solution is found.

**Pseudocode:**

```
Initialize population P with random solutions
Repeat until termination criterion is met:
    Select subset S of best solutions from P
    Estimate joint distribution p(X_1, X_2, ..., X_n) from S
    Generate new population P by sampling p(X_1, ..., X_n)
```

**Key Differences**

| Feature | Univariate EDAs | Multivariate EDAs |
|---|---|---|
| Assumptions | Variables are independent. | Accounts for variable dependencies. |
| Complexity | Simpler to implement. | Computationally intensive. |
| Distribution | Separate for each variable. | Joint distribution over all variables. |
| Use Cases | Suitable for simpler problems. | Handles complex, dependent variables. |

**Applications in Real-World Problems**

Using EDAs, **personalized meal plans** can be optimized to ensure a balance of macronutrients (carbohydrates, proteins, and fats). The algorithm identifies patterns from individuals' dietary preferences and health data to recommend combinations that suit their lifestyle while maintaining nutritional balance.

EDAs can help optimize **traffic flow** across a city. By analyzing patterns in vehicle movement and road usage, the algorithm recommends optimal traffic light timings or alternate routes to reduce congestion, saving time and fuel for commuters.

In an **e-commerce setting**, EDAs can be used to create product bundles that maximize sales. The algorithm identifies purchase patterns (e.g., customers buying laptops often buy accessories like mice and keyboards) and designs bundles that appeal to customer preferences, increasing revenue.