Initialization

The initialization phase of a genetic algorithm marks the beginning of its process to find solutions. This phase involves creating an initial group of individuals, each representing a potential solution. The population's size and the method of initializing these individuals are critical. Generally, initialization uses randomness to ensure a wide exploration of solutions, but if specific knowledge about the problem is available, a more targeted approach can be taken.

Each individual is assessed for their fitness value after the population is created. This evaluation helps gauge the diversity and quality of solutions, setting the stage for the algorithm's evolution.

For problems where optimal solutions are unclear, like the OMP, initialization might involve generating random binary strings for each individual. For example, in a population of 100 individuals, each might start with a 50-bit string.

The Evolution Engine manages this process, overseeing initialization, evaluation, and evolution. It is configured through parameters like population size and the structure of individuals. Here's a Kotlin code example to illustrate the setup:

```kotlin
// Define the population size.
private const val POPULATION_SIZE = 100

// Configure and instantiate the Evolution Engine.
val engine = evolutionEngine(::count, genotypeOf {
    chromosomeOf {
        booleans {
            size = CHROMOSOME_SIZE  // Length of each individual's chromosome.
            trueRate = TRUE_RATE    // Initial probability of a gene being `true`.
        }
    }
}) {
    populationSize = POPULATION_SIZE  // Set the population size.
    // Additional configurations can be added here.
}
```

This code sets up the Evolution Engine with a fitness evaluation function and the genetic structure of the population, readying the algorithm for its evolutionary journey.