Mateo W. Racca

UNIVERSITAT ROVIRA i VIRGILI
Escola Tècnica
Superior d'Enginyeria

NEURAL AND EVOLUTIONARY COMPUTING

(MESIIA): Assignment #4: Optimization with Genetic Algorithms

Abstract:

This report explores the application of genetic algorithms (GAs) to the Traveling Salesperson Problem (TSP), a classical problem in combinatorial optimization. The objective is to find the shortest possible route that visits a set of cities and returns to the origin point. Our methodology involves translating the TSP into a GA framework, where routes are chromosomes and the total travel distance is the fitness value to be minimized. We experiment with various selection, crossover, and mutation techniques to optimize performance.

The script, datasets and results are available at: https://github.com/raccamateo/NEC_A4

Introduction:

The Traveling Salesperson Problem (TSP) is a well-known problem in the field of optimization and computational mathematics. It involves finding the most efficient route for a salesperson who must visit a list of cities and return to the starting city, with the constraint that each city is visited exactly once. This problem has significant applications in logistics, planning, and the organization of networks. The genetic algorithm approach to the TSP is motivated by the need for heuristic methods that can provide good solutions within reasonable timeframes, especially for large datasets where exact algorithms become impractical. By simulating the process of natural evolution, GAs adaptively search through the space of potential solutions, leveraging mechanisms such as selection, crossover, and mutation to evolve routes that approximate the shortest possible path. The objective of this report is to demonstrate the effectiveness of GAs in finding near-optimal solutions to the TSP and to analyze the impact of different algorithmic parameters on the quality of the solutions obtained.

2. Problem Representation:

In the context of genetic algorithms, the TSP is represented as a chromosome, which is essentially a specific sequence of genes. Here, each gene represents a city, and the chromosome as a whole represents a complete tour. The permutation encoding is crucial because it naturally represents the path through each city. In this encoding scheme, each city is listed once, and the order of cities represents the sequence in which they are visited. This direct approach maintains the integrity of the problem by ensuring that each city is visited exactly once before returning to the starting point.

3. Genetic Algorithm Components:

A genetic algorithm (GA) mimics the process of natural selection by creating a population of solutions and iteratively applying selection, crossover, and mutation to evolve towards the best solution.

3.1 Selection Techniques:

Selection methods determine how individuals from the population are chosen to create new offspring. Tournament selection involves randomly selecting a subset of the population and choosing the best among them. Roulette wheel selection assigns selection probability proportional to fitness. Rank-based selection involves ranking individuals and assigning selection probability based on this ranking. These methods were chosen to balance exploitation and exploration in the genetic search process.

3.2 Crossover Methods:

Partially Mapped Crossover (PMX) and Order Crossover (OX) are two techniques that respect the permutation encoding. PMX ensures that offspring inherit a mixture of parents' genes with a specific mapping to prevent duplicate cities in the tour. OX, on the other hand, maintains the relative order of cities from one parent and fills the remaining cities from the other parent. These crossovers are used to produce new offspring that combine traits from both parents while still representing valid TSP solutions.

3.3 Mutation Techniques:

Scramble mutation randomly shuffles a subset of genes, and displacement mutation moves a subset of genes to a different position in the chromosome. These mutations introduce variability into the population, helping to explore new areas of the solution space and prevent premature convergence. The choice of these mutation methods is to maintain genetic diversity and allow the algorithm to escape local optima.

4, Population Size and Stationary State:

The population size in a genetic algorithm is a crucial parameter that affects both the search diversity and the computational resources required. A larger population size offers a more diverse genetic pool which can explore the solution space more thoroughly, but it also demands more computational power. Conversely, a smaller population size can lead to quicker convergence but might get trapped in local optima.

A stationary state is typically identified when there is little to no improvement in the population's fitness over a number of generations. This indicates that the algorithm has likely converged to a near-optimal solution, or that it is stuck in
In the context of the genetic algorithm for the TSP, the population size is pivotal. It dictates the algorithm's ability to navigate the solution space and impacts computational efficiency. A carefully balanced population size can ensure diverse genetic material to avoid premature convergence while maintaining a manageable computational load.

A stationary state is detected when successive generations do not yield significant improvements in the best solution found. This plateau suggests that the algorithm has either converged on a solution or is stuck in a local optimum. The identification of such a state often involves monitoring the best fitness value or employing a convergence criterion, such as a threshold below which the variation in fitness is considered negligible.

For the genetic algorithm's implementation, the chosen parameters aim to strike a balance between exploration and exploitation. A diverse initial population, robust selection methods that pressure towards fitness while preserving diversity, and carefully tuned crossover and mutation operations that introduce new genetic variations without disrupting advantageous traits were the key. Varying these parameters and observing their effects on the convergence pattern provided insights into the algorithm's behavior and efficiency. The visual representation of the algorithm's progress through generational minimum distance plots is a powerful tool to analyze performance and convergence visually.