TP4: Evolutionary Algorithms

Goals

The objectives of this exercise sheet are to:

- Develop a practical understanding of genetic algorithms through implementation;
- Compare different selection, crossover, and mutation operators on various problems;
- Analyze the performance and behavior of evolutionary algorithms on different problems.

Some other information:

- The coding language for this course is Python.
- Always write tests for each exercise to assure yourselves that your code works as expected.
- You can discuss in groups, but the code and tests should be written individually.
- Some example instances are contained in the zip archive `bioalg-tp04.zip`.

1 The Knapsack Problem

The Knapsack Problem is a classic optimization problem where we are given a set of n items, each with a weight w_i and a value v_i, along with a knapsack capacity W. The goal is to select items to maximize the total value while keeping the total weight within the knapsack capacity.

Exercise 1.1. Implement a genetic algorithm for the 0-1 Knapsack Problem using the following components:

- Representation: binary string encoding;
- Fitness: total value of selected items, with a penalty if the weight constraint is violated;
- Crossover: single-point crossover;
- Mutation: bit-flip mutation with probability $p_m = 0.05$;
- Parent selection: tournament selection of size $k = 3$;
- Survivor selection: best overall (also known as $(\mu + \lambda)$-selection)

Test your algorithm using handmade examples and the example provided.

Note: Some instances for this problem are provided, obtained from the work of Xiang et al. [XAL+14]. The first line of the file has two integers n and W separated by a space, and then one line for each time, containing its value v_i and weight w_i separated by a space.

In the file `KP7.txt`, the optimal solution has a value of 107 and includes items 1 and 4.

Exercise 1.2. Extend your genetic algorithm by implementing and comparing different parent selection methods, namely roulette selection, rank selection, and different tournament sizes

Analyze how each method affects the solution quality, genetic diversity of the population, but also the selection pressure (how diverse is the set of parents).

Exercise 1.3. Implement and compare the two-point and uniform crossover operators.

Analyze how each crossover operator affects solution quality, genetic diversity of the population and convergence speed.

2 Traveling Salesperson Problem

The Traveling Salesperson Problem (TSP) involves finding the shortest possible route that visits each vertex of a graph exactly once and returns to the origin vertex.

Exercise 2.1. Implement a genetic algorithm for the TSP with the following components:

- Representation: permutation;
- Fitness: Total distance of the tour (smaller is better);
- Crossover: permutation insertion with probability $p_c = 0.8$;
- Mutation: Exchange with probability $p_m = 0.1$;
- Parent selection: Tournament selection;
- Survivor selection: generational.

Test your implementation on the provided TSP instances.

Exercise 2.2. Implement and compare the following crossover operators specific to permutation representations:

1. Partially mapped crossover (PMX): as in permutation insertion, we choose a segment of parent Y to copy to the child, but now we "replace" the elements from this segment in parent X by the elements in the segment.

   Example: $(2, 3, 4, 5, 6, 1) + (2, 4, 1, \mathbf{6}, \mathbf{3}, 5) = (2, 5, 4, \mathbf{6}, \mathbf{3}, 1)$

2. Edge Recombination Crossover (ERX): make a list of the neighbors of each vertex in either parent, then build a new permutation by starting at a random position and picking as the next one its neighbor on the list with the fewest unused neighbors.

   Example: $(2, 3, 4, 5, 6, 1) + (2, 4, 1, 6, 3, 5)$
   The neighbor sets are: $\{2, 6, 3\}$, $\{3, 1, 4\}$, $\{2, 4, 1, 5\}$, $\{5, 3, 6, 2\}$, $\{6, 4, 3\}$, $\{1, 5, 4\}$.
   Possible solution $2 \to 1 \to 6 \to 4 \to 5 \to 3 \to 2$, or $(6, 1, 2, 5, 3, 4)$.

Which operator performs best for TSP?

Exercise 2.3. Enhance your TSP genetic algorithm with local search by implementing a 2-opt local improvement procedure and applying it to the best individuals in each generation.

Compare the performance of the standard genetic algorithm versus the improved version, and analyze the trade-offs between computational cost and solution quality.

References

[XAL+14] Wan-li Xiang, Mei-qing An, Yin-zhen Li, Rui-chun He, and Jing-fang Zhang. A novel discrete global-best harmony search algorithm for solving 0-1 knapsack problems. *Discrete Dynamics in Nature and Society*, 2014(1):573731, 2014.