Pseudocode explanations and Steps

The algorithm is based on the article (Beasley, Bull and Martin, n.d.)) which was a general-purpose summary and a view of the Genetic Algorithm.

As a first step, elaborating on the user input was necessary and eventually converting it into a way that a GA could understand. Given a sequence coming from the user, if the matches the requirements in section 3, we could proceed.

According to (Vijini Mallawaarachchi, 2017) a full solution to the problem is represented by a chromosome, a string of 1 and 0 where each bit is a gene.

Instead of using straight the binary representation of the words, a dictionary has been created, hashing each letter into a couple of integers, since with two bytes we could represent 4 different symbols. The user input must be part of the solution and we consider them as genes that have reached their convergence, converting to the respective representation in integers couple. The chromosome ( a possible solution representation) must be complete, thus, randomly a letter will be set avoiding the creation of an impossible sequence.

The third step is to create a population to start with. Trying to avoid premature convergence, a small number is provided.

Is important to recall that the algorithm is not determinist and therefore is recommended to set the number of generations as the parameter.

Give that, in step four we start a loop:

UNTIL WE HAVE NOT REACHED MAXIMUM GENERATION:

1. EVALUATE THE CURRENT FITNESS VALUE OF THE WHOLE POPULATION.
2. BASED ON THE PREVIOUS STEP, CREATE NEW COUPLES
3. APPLY THE CROSSOVER OPERATOR
4. CALL MUTATION OPERATOR
5. INCREASING THE NUMBER OF GENERATION
6. DELETE BETWEEN 5 and 10 solutions with the lowest score.
7. ADDS 5 NEW RANDOM INDIVIDUALS TO THE POPULATION
8. REPEAT

FITNESS VALUE:

Each chromosome if does fit guides the entire algorithm. The fitness value is the sum of each rule violated by this instance. This function scans the matrix by row, column, and submatrix (for the respective scanning technique, consult the appendix). A sum of all violations is returned. At the end of this step, we have the population classified according to their violations.

SELECTION\_OF\_NEW\_COUPLES:

Here, we select the couple for the operator. A chromosome could not be coupled with himself, and we would like to promote the early indexed as they are more likely to succeed. However, the number of couples would be restricted to 10 per generation.

APPLY THE CROSSOVER OPERATOR:

Given the list of couples, this function chooses a random number between 1 and 14. The number is equal for this generation, more on this later. The next step would be looping between couples and combining them.

UNTIL THERE IS A COUPLE, REPEAT:

1. SPLIT first chromosome in the random index chosen in the previous step
2. SPIT the second chromosome in the random index chosen in the previous step.
3. JOIN the first part of the first chromosome with the end part of the second chromosome.
4. JOIN the first part of the second chromosome with the end part of the first chromosome.
5. ADD the new individual to the population

CALL MUTATION OPERATOR:

Here we do select randomly a number, if this number is greater than 0.23, we apply the operator.

This operator changes a bit into the chromosome. Therefore, a gene must be swap its own content with something else in the list of words.

In doing so, it’s important to check we don’t alternate any of the user input since are constraints for us.

Because there is no use to create a chromosome that would lead us to an impossible scenario, this step, it’s generated only a word that does not appear in that row and that column.

DELETE BETWEEN 5 and 10 solutions with the lowest score.

There is no use to keep tracking for solutions with the high number of errors, so we clean up every 5,10 cycles of this individual in the population.

DDS 5 NEW RANDOM INDIVIDUALS TO THE POPULATION

Replaying with new data the old in the previous step. This is particularly useful to avoid premature convergence of the population to a local maximum (Beasley, Bull and Martin, n.d.).

References:

Beasley, D., Bull, D. and Martin, R. (n.d.). *An Overview of Genetic Algorithms : Part 1, Fundamentals*. [online] Available at: <https://mat.uab.cat/~alseda/MasterOpt/Beasley93GA1.pdf>.

Vijini Mallawaarachchi (2017). *Introduction to Genetic Algorithms — Including Example Code*. [online] Towards Data Science. Available at: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3.