

Fall 2022 COSC 3P71 Artificial Intelligence: Assignment 2

Instructor: Prof. B. Ombuki-Berman

Course Coordinator: Alanna McNulty (amcnulty@brocku.ca)

Tutorial Leader: Nicholas Aksamit (na16dg@brocku.ca)

Consultation Hours TAs: Liam McDevitt, Alanna McNulty & Tom Wallace

Teaching Assistants: Nicholas Aksamit, Liam McDevitt, Alanna McNulty & Tom Wallace

Available Date: October 19th, 2022

Due Date: 8:00 AM November 11, 2022 (NO LATES)

Recently the world has been shocked by new accusations against the totally real company We're Not a Scam Inc. A few clients have come forward with suspicious sounding stories and the authorities have taken interest. It looks like the people over at We're Not a Scam Inc., may have, in fact, actually been scamming people. Recently a large collection of shredded documents has been recovered from We're Not a Scam Inc. and the authorities are extremely interested in the contents of these documents. Since re-assembling shredded documents by hand is a time consuming and tedious task, the authorities are asking for assistance. A computer program that can help recover shredded documents is required.

Goal: Showcase your understanding of Genetic Algorithms (GA) and prepare a technical report examining the algorithm's performance in a given problem.

Task: This assignment has 3 parts. First, you must implement a GA system as described in lecture. Next, you must perform a number of experiments with your GA system and collect data regarding its performance. The final step will be preparing a technical report to present your findings. Specific details regarding the requirements of the GA implementation, the experiments to be performed, and the format and content of the report follow.

As a reminder, the basic procedure of a GA is as follows:

```
Read problem instance data
Set GA parameters
Generate a random initial population, POP, of size popSize

for gen=1 to MAXGEN do:
    evaluate fitness of each individual in POP
    select a new population using a selection strategy
    apply crossover and mutation
end for
```

A GA could have the following components:

- Initial Population Initializer: Creates a population of size popSize of randomized individuals as described in class
- Chromosome: A chromosome encodes a solution to the problem being solved. In our case each chromosome must represent a sequence of distinct integers. The simplest representation to use is an integer array, mutation and crossover can then be performed on the elements of the array, although other representations are possible. The array should contain each of the integers 0,1,2,...,n-1 in some order. No integer should be present more than once, and all integers from 0 up to and including n - 1 should be included, where n is the size of the array.
- Reproduction: Use Tournament Selection (remember $K = 2, 3, 4$ or 5)
- Crossover: Given two individuals, a crossover creates two offspring. Implement your GA using the following crossover strategies independently:
 - Order Crossover
 - A crossover of your choice for example: Uniform Order Crossover or Partially Mapped Crossover
- Mutator: Given an individual, a mutator creates a mutated individual. Implement your GA using a mutation operator of your choice (from those discussed in class)
- Fitness evaluation function: The fitness function should take an individual and produce a real number describing the suitability of the solution encoded in the individual. In our case the fitness function will attempt to describe how much an unshredded document looks like English. One possible fitness evaluation function is provided, but feel free to experiment with other options. With the provided fitness function smaller numbers indicate a more fit solution.
- Genetic algorithm system: The implementation of the GA system. This file should glue together the various components of your system.
- User parameters: Population size, maximum generation span, probability of (crossover, mutation, etc.). Your GA program should permit the user to easily define his/her own genetic parameters and data (e.g., crossover rate, mutation rate, population size, maximum generation span etc. . . .).

Using your GA implementation perform the following experimentation:

- Run your GA to compare the performance of the two crossover operators mentioned above by using the following parameters (and include elitism in all cases):
 - a) Crossover rate = 100 %, mutation = 0%
 - b) Crossover rate = 100 %, mutation = 10%
 - c) Crossover rate = 90 %, mutation 0%
 - d) Crossover rate = 90 %, mutation 10%
 - e) Determine your own best parameter settings

BONUS (For a bonus of 2% to your total course grade): Incorporate into your experiments your own innovative idea. This could be a different initial population representation and creation strategy, a different selection scheme, a different (third) crossover not discussed in class, etc.

Alternatively, you could introduce a local search into your GA, or evaluate your own fitness function.

- For each experiment mentioned above, run your GA at least 5 times with 5 different random number seeds. Shredded documents will be provided. **You should repeat the experiments above for document1- shredded.txt, document2-shredded.txt, and document3-shredded.txt**
- For all three documents your chromosomes should have length 15.
- For each run, your GA system should output the following to a file or standard out:
 - All GA parameters, including random number seed
 - Per each generation: best fitness value, average population fitness value
 - Per each run: best solution fitness and its corresponding best solution chromosome
- Finally, prepare a summarized report of your findings using IEEE format introduced to you at the tutorial. IEEE format details are found at: <https://www.ieee.org/conferences/publishing/templates.html>. An IEEE report in this format is expected to be 8 pages in length. The report should have the following sections and each section should address the listed points:

Introduction

- * BRIEFLY introduce the concepts and topics discussed in the report.
- * Precisely define the problem and explain why its solution is important.

Background

- * This section should explain the algorithms used in the report (pseudo code is helpful) and may provide other information which you feel will be relevant to someone trying to understand your results.

Experimental Setup

- * This section should provide enough information about your experiments to allow someone else to duplicate your results.
- * This should include algorithm parameters used, the crossover and mutation operators used, and any other relevant implementation details.

Results

- * This section should summarize your findings.
- * For your multiple runs compute the average of best fitness per generation and average population fitness per generation. Using a graph drawing tool such as excel, plot well labeled graphs for your experiments. If you decided to do the bonus, plot graphs for it as well. The types of graphs you plot will depend on your incorporated idea, if any. Feel free to experiment with different crossover and mutation rates. You will set your own Pop-Size and generation size.
- * Also include summary tables describing the fitness of your final solution. Summary statistics such as min, max, mean, median, and standard deviation

should be included in your tables. Tests for statistical significance would also be appropriate, for example T-Tests or Mann- Whitney U tests.

- * Explain your graphs/data in detail and emphasize the similarities and differences between different algorithm configurations.

Discussions and Conclusions.

- * This section should provide a BRIEF summary of what experiments you performed and the results you observed.
- * Following this BRIEF summary, you should discuss your opinions regarding your results and what conclusions you've arrived at.
- * This could include issues like which crossover performed better. If more than one mutation type was tried, which one performed better. If you included local search, did it help? How did the choice of GA parameters affect the final outcome etc.?

References

- * List your sources here. The text of the report should contain references to your sources.

This report is very important, so be sure to include it. Start early, gathering the data and doing the experimental analysis will take much more time than coding the assignment.

NOTE: You are allowed to discuss the assignment with friends, but you must do the work independently.

Implementation Notes:

- For the sake of testing, the original, unshredded documents have also been provided.
- For each shredded document (document1-shredded.txt, document2-shredded.txt, and document3-shredded.txt) the chromosome length should be 15.
- The provided fitness function will accept a shredded document, a permutation, and return a real number, v , indicating how much the de-shredded document looks like English. A smaller number indicates a better solution. The permutation describes the order that the shreds of the document should be placed in.
- The provided code contains a fitness function, a function for reading shredded documents from disk, and a function for printing de-shredded documents in an easy-to-read format. This code will be discussed in tutorial.

Your electronic submission should include:

- Your report
- Your source code
- An executable
- Instructions for running and compiling your program and changing parameters

- The data you've generated for your report

Hand in and submission notes:

- Submission will be done through Sakai.
- Unity is not allowed for this assignment by any students. We recommend using Java 8, Python 3.6, or C#. Your source code should be presented in a format that can be easily compiled and use a programming language that the TAs can access in our COSC labs. All assignments must include detailed instructions for the marker to compile and run the submission. Failure to provide adequate explanation and documentation may result in a submission not being graded.
- For those that wish to, please note that the Brock virtual COMMONS is also available to all students at Brock. If you are unsure of whether or not the markers will be able to test your submission you can test it on the virtual COMMONS. Machines in the virtual COMMONS have IDEs for Java, Python, and C#.
- Any questions/concerns etc., regarding the grading of any assignment MUST be raised within 7 days of graded assignment hand-back. In this case, please send your concerns/questions to Course Coordinator: Alanna McNulty at: "amcnulty@brocku.ca". To better serve you, please don't send multiple queries on the same topic to Alanna, Professor, and other TAs. Alanna will be the point of contact for any such queries and the Professor will receive them all at once from Cody.
- If we deem it necessary, we will be creating a FAQ for this class and adding some of your questions there (while maintaining your privacy). In this case, please send all your email questions to Alanna at "amcnulty@brocku.ca". Alanna will try her best to respond to all your questions as soon as possible, so please wait 72 hours before you contact course instructor if Alanna has not responded. The Tutorial Leader (Nicholas) will discuss this assignment during tutorials starting this coming Thursday (he will also take note of tutorial attendance on Teams). The instructor will have access to all your questions via Alanna and tutorial questions (via Nicholas).