Artificial Intelligence: Genetic Algorithms

Yelyzaveta Denysova
Brock University
St. Catharines , Canada
ed18xy@brocku.ca 6667596

I. Introduction

Evolutionary algorithms were inspired by the metaphor of natural selection in biology [1]. Individuals produces offspring that populate the next generation. In genetic algorithms (GA) individual is represented as a string over a finite alphabet just as DNA is string over the alphabet ACGT.There are other important parameters defining the evolutionary algorithm: population size, selection process, recombination procedure, mutation rate, make up of the new generation and more.

The way creatures are changing from generation to generation becoming more adaptable to their environment was observed by people and used to create solution by drawing the parallel between the problem and real life. This paper will examine how good Artificial Intelligence can solve complex problems without knowing the outcome beforehand. We will learn how can different parameters impact the performance of GA.

We will look at the problem that is unrealistically time consuming to be done by hand so the assistance of a computer power was needed. A computer program was required to help recover the information stored in the shredded documents. Although the documents were recovered, the text is very distorted and unreadable. However, there was a pattern in the way lines were cut, so we just needed to rearrange them to get some real text.

II. Background

I have developed a computer program that uses GA to come up with the solution for the rearrangement of cut lines of the documents to retrieve the original text.

It accepts the file name in input folder as a command line argument for the file with all the parameters: Population size, Random seed, Maximum number of generation, Input file name, Crossover rate, Mutation rate, Tournament selection k, Crossover type: O for order crossover, U for uniform order crossover, Number of individuals selected for elitism.

The main class GA implements the genetic algorithm and contains the main running method. It accepts the input and writes the output. Crossover is a recombination procedure to get new individuals that has common features to their parents but also are different. In this assignment I used Order crossover and Uniform order crossover. The order crossover procedure is this (pseudo code):

- Get 2 parents
- Randomly select a set of 8 genes (I picked 8 because it is around the half of the full length of DNA)
- Copy the set from parent 1 to child 1, and from parent 2 to child 2.
- Go through all empty genes and fill with the values from the other parent avoiding duplicates

The Uniform order crossover procedure is very similar. The only difference is the set not random but copied according to the mask that is randomly generated containing 1's and 0's. 1 - keep the value, 0 - leave empty. Mutation procedure: For the mutation I didn't want to alter the DNA too much so I'm just swapping 2 different random genes

The FitnessCalculator class was provided by the instructor and has function to evaluate the fitness of the individual for this specific problem and has other helpful functions.

Population class imitates the population of the individuals as an array. The constructor initializes the population with specified number of individuals.

Innovative idea: As my personal idea, I have added a special method in initializing the populations. Since the text starts with capital letters, the initial chromosomes contain the position of a capital as their first gene. The program reads specified shredded file and identifies the capital letters in the first line to be used at the start of DNA of individuals.

Individual class represents individuals that have chromosomes encoding the solution to the problem. In this case chromosome is an array of integers representing the sequence in which the cut pieces needs to be rearranged to retrieve readable text. It also has a fitness as a numeric evaluation of the how good individuals fits the solution.

Program execution can be described as follows(pseudo code):

- Read and set the parameters
- Read input file
- Initialize the first generation population
- Create output file and record all GA parameters
- for generation=0 to MAXGEN do:
- Evaluate fitness of each individual
- Select n best individuals to pass selection: elitism
- Select more individuals via tournament selection
- Apply Order or Uniform order crossover to create children for new generation
- Apply mutation
- Move to the next generation
- endfor
- Print final results and unshredded document using the chromosome with the best fitness

In addition to this, to make the experiments easier I have developed test class that creates all the parameter files to test 3 text files for 5 crossover and mutation variations with order and uniform crossover for 5 different random seeds. All of them are included in the input folder. It also prints to the console names of all of the created files so you can use that as a command line parameter when running GA

A. Compile and run instructions

First you need to create a txt file for all the GA parameters in input folder. In that folder you can find example.txt and descriptionOfIntput.txt to see what the file should look like.

The output file will be created in the output folder with a timestamp for easier navigation.

Open the terminal and navigate to the folder GA. It contains ScamTest which is the name of our program. Run javac ScamTest/*.java command to compile. Run main class GA and specify your parameter file like: java ScamTest.GA ScamTest/input/example.txt. Output files are generated with a timestamp to insure unique names.

III. Experimental Setup

To test and experiment with implemented GA I have prepared 150 input files (available in the input folder).

Different parameters were:

- Crossover rate
- Mutation rate
- Input shredded file (3 provided files)
- Crossover type: order or uniform order
- Random seed (5 different seeds: 0,1,2,3,4)

Same parameters:

- Population size: 200
- Maximum generation (number of iterations): 30
- Tournament k: 5

For the sake of experiment I decided to keep some of the parameters the same to see the difference of changes on other parameters. I chose 200 for the population size to insure wide diversity. According to few test runs, the average and best fitness value plot would flatten around after 25th generation, so I set the maximum generation value to 30 so that flattening is visible so we know we've reached the best solution for this run.

The crossover and mutation rate variations were:

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) Crossover rate = 95%, mutation 15%

IV. Results

All the files with the output are stored in the output folder, each file has:

- all the parameters used
- Average and Best fitness values per generation
- Best fitness value and corresponding chromosome
- Unshredded text using best chromosome
- Organized fitness values for statistics

After running the algorithm and outputing the results I organized them in excel file statisticsDataAndMore.xlsx. Excel file contains all the data in a large table organized by parameters. Under the table you will find a statistic summary, t-test, graphs, and other tables used in this report. Some columns highlighted in colors. Colors represent how readable text was. Colors do not correspond to the fitness value but rather my opinion on how easy you can recognize English in the text. Light beige - few letters are off but mostly recognizable, Bright yellow - only 1 or 2 letters are out of order but the text is very readable, Green - text was unshredded perfectly.

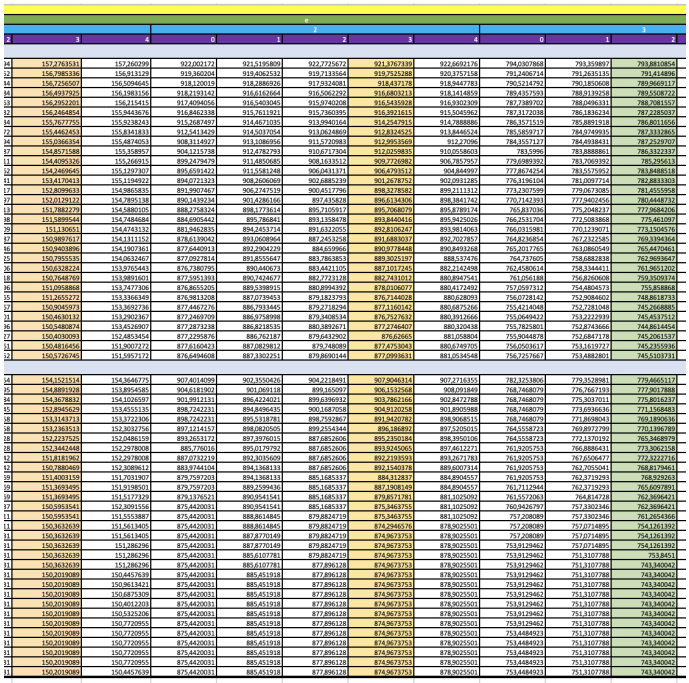

Fig. 1. An example of color code: light beige, bright yellow, green

Mostly color code is proportional to lower fitness values but not always. For some parameters, the final solution might be almost perfect text but for other parameters text will be less readable even though the fitness value is lower.

On figure 2 perfectly unshredded document 3 was generated with the lowest fitness value chromosome. However, the bright yellow column is very close to solution even though it does not have the lowest value. The lowest value for documents 2 is depicted on the figure 3.

Top part are average fitness values per run and the bottom are the best values per run. I gathered the following statistics from all the best solutions for all runs per document. Since numbers vary quite a lot between documents it is better to examine them separately. Figure 4 is the table with the total statistics

From my analysis, the options where crossover rate and mutation rate together is equal to or above 100% gave the best results. I believe that is because such parameters ensure wide

782,3253806	779,3528981	779,4665117	780,6467923
777,5664788	774,0991885	775,7781206	775,7413604
774,8421278	773,6073097	776,6565084	772,6794469
773,2817004	775,6916634	774,9689935	769,5068239
773,5649283	775,7782979	775,2264544	772,2880319
770,5300479	770,716725	771,7106198	768,8712518
769,9808198	766,1222949	768,2677543	768,5604098
766,5277546	766,1222949	768,3969363	761,2822761
768,4071546	765,4116563	761,5533046	761,2822761
766,2807845	764,3755692	769,8510411	761,2822761
768,4020342	758,0087486	764,2690444	760,0201309
769,43016	755,8504761	764,2690444	761,5131795
766,1667767	755,8391642	761,6263351	755,7021642
769,112224	755,8391642	759,3866728	754,1601644
769,0206267	755,8391642	757,4033466	754,1601644
768,3646591	755,8391642	757,4033466	750,536835
765,4312843	754,0884114	757,1681765	750,5335417
762,8633622	754,0884114	755,9532476	743,340042
757,819223	750,6744018	752,2282717	743,340042
757,819223	750,6744018	753,1084848	743,340042
755,1376866	750,6744018	752,2282717	743,340042
747,1630431	750,6744018	752,2282717	743,340042
747,1630431	750,6744018	752,2282717	743,340042
747,1630431	750,6744018	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042
747,1630431	750,6455947	750,8824759	743,340042

Fig. 2. An example of color code VS highlighted in red lowest values

907,4014099	902,3550426
904,2119509	897,7029549
896,4438844	889,6442763
898,8340552	884,5560741
887,4885224	884,5560741
890,2358493	879,4063528
888,6559863	879,4063528
887,7782932	879,4063528
886,5981175	871,1662193
879,7005913	871,1662193
871,1662193	874,3868372
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193
871,1662193	871,1662193

Fig. 3. An example of color code VS highlighted in red lowest values

	Document 1	Document 2	Document 3
Minimum	149,56353	871,1662193	743,340042
Maximum	152,4999564	886,3762225	761,1266236
Median	150,420762	876,6186365	753,3969281
Mean	150,5135677	877,2966524	752,6081081
Standard Deviation	0,284114343	2,781094616	1,831816389

Fig. 4. Summary statistics

diversity increasing the possibility of finding better solution. However, there are downsides to it as well. Consider the graph on figure 5.

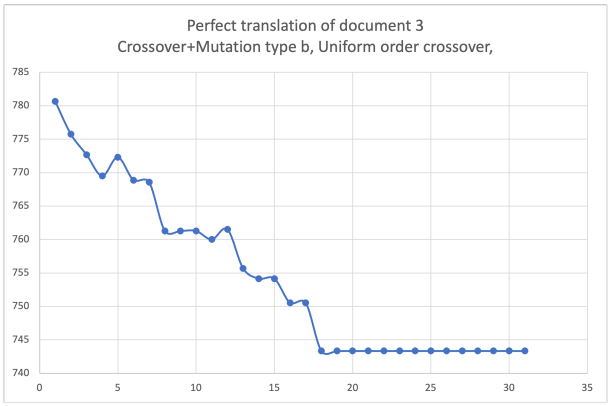

Fig. 5. Best fitness values per generation

For this graph the crossover rate is 100% and mutation rate is 10%. As we can see there are few points where graph goes up. This is because there is a very high chance that all of the DNA will be changed, even elite members which might negatively reflect on accuracy. However, if the maximum number of generations is large enough, the curve reaches the absolute minimum and flattens out.

Therefore, I picked my own parameters of 95% crossover rate and 15% mutation rate, which turned out to be pretty good for this problem. On figure 6 you can see the graph of the best fitness values per generation for document 3 and on figure 7 - document 2.

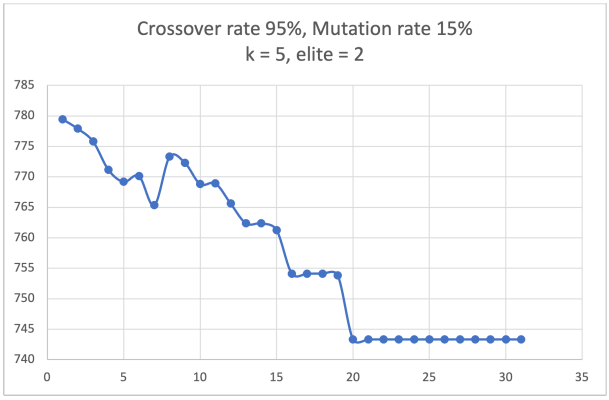

Fig. 6. Best fitness values per generation: Doc 3. Final solution restored th original document perfectly

So far we were looking at the graphs for uniform order crossover, which in fact did certainly better for document 3. However, the absolute minimum fitness value for the final solution was with the same parameters but order crossover. Consider the graph on figure 8.

There are still ups and downs due to high crossover and mutation rate, but it is more consistent and does flatten quicker. There were multiple best solutions equal to the absolute

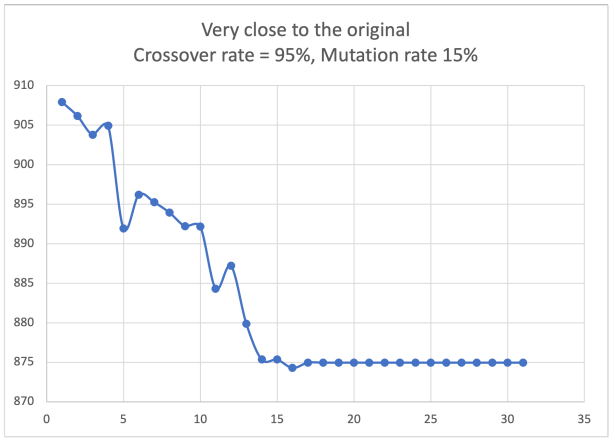

Fig. 7. Best fitness values per generation: Doc 2. Final solution had only 1 position misplaced

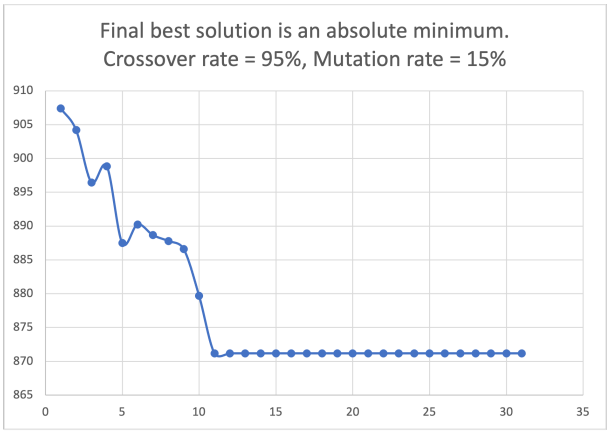

Fig. 8. Best fitness values per generation: Doc 2. Order crossover

minimum. Consider graph on figure 9: Order crossover type with rates variant a.

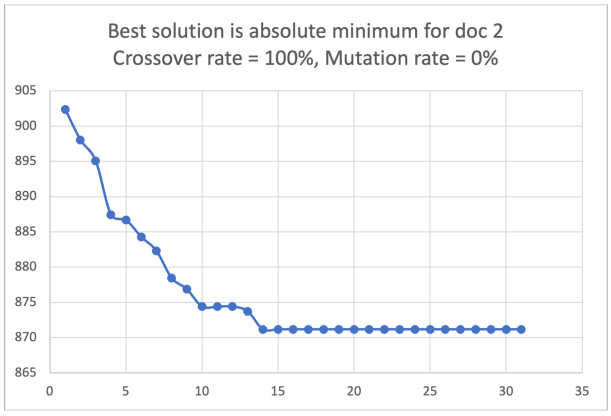

Fig. 9. Best fitness values per generation: Doc 2. Order crossover

This graph is noticeably more consistent and does not go up at all, meaning the best fitness values always improving. That is most likely due to absence of mutation that can modify DNA in unpredictable ways. Of course, the random seed also plays its role. GA in programming, just like in nature relies on randomness. However, impact of that is not as significant as changes on the parameters. Let's examine few graphs based on the average best fitness values among all 5 runs.

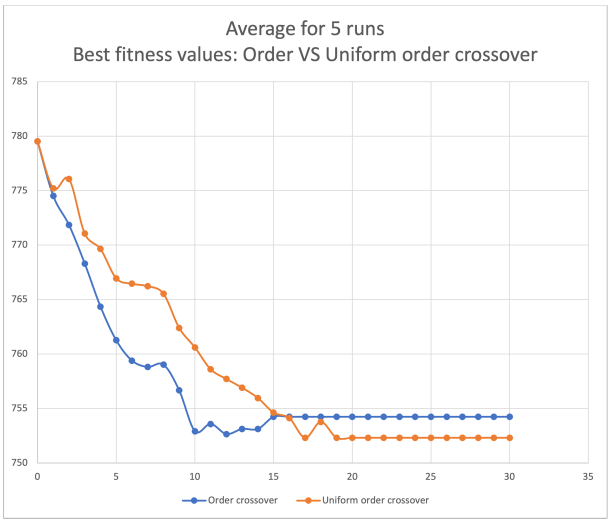

Fig. 10. Best fitness values average for 5 runs

On figure 10 we can see order and uniform order crossover with rate 100% and no mutation for the 3rd document. With order crossover GA never reached as low values as with uniform order. Another interesting observation is how context of a problem influences the results. Figure 11 shows average for 5 runs of best fitness values for 3 documents with the same parameters (Order crossover with the rate 95%, Mutation rate 15%). We can see a similar trend but the actual values vary significantly. It is important to say that actual readability was not affected, like unshredded document 1 was not much better that unshredded document 2.

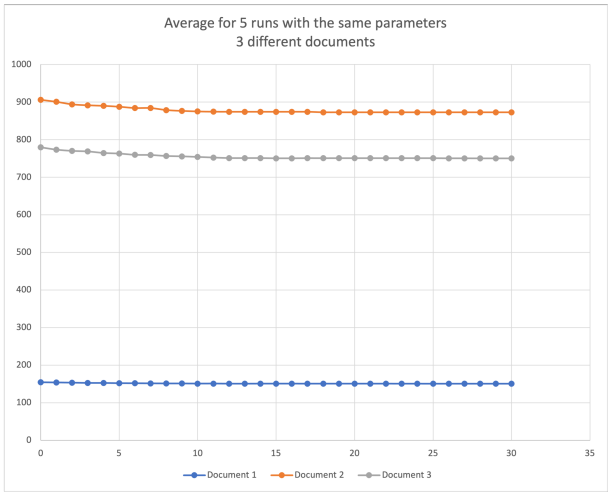

Fig. 11. Best fitness values average for 5 runs: Different context

We have established that crossover procedure and rate have impact on the GA results so now we will focus on the

crossover. We will test thee hypothesis that with mutation GA produces lower final fitness value therefore it is better to include mutation. The data I have gathered are best fitness values from 5 runs per document and we will compare a and c crossover+mutation parameters against b and d parameters. Original data and results are summarized in the table on figure 12.

	Best fitness values for each of 5 runs (a,c VS b,d)					
T-Test	Doc1: without mutation	Doc1: with mutation	Doc2: without mutation	Doc2: with mutation	Doc3: without mutatio	Doc3: with mutation
	150,7650856	150,5511888	878,7212432	871,873678	754,5734846	757,8496273
	150,5605397	150,6261648	871,1662193	878,7699926	750,1811452	752,7563914
	151,3312585	150,5871207	871,933192	871,933192	757,3235575	751,1223917
	150,3945339	150,2919704	880,7294943	875,2043418	750,7680512	747,1386566
	151,3387241	151,0421021	874,3163436	874,3163436	758,3280531	754,0282645
	151,3764826	149,56353	871,873678	877,5401718	754,5734846	754,5734846
	150,6798945	150,3610462	871,1662193	879,3133407	747,8804112	747,8780711
	150,5871207	149,6746758	874,6671396	874,7143876	753,7924477	752,2282717
	150,2919704	150,2919704	881,9286323	879,1498641	757,1108489	750,7680512
	151,4575207	150,3288418	886,3762225	874,3163436	758,5261267	756,8406861
p=	0,006550764		0,383475168		0,139833118	

Fig. 12. T-TEST

P-value for document 1 is insignificant, but for document 2 and 3 it is greater. We can conclude that mutation helped improving the results for document 2 and 3, but what about 1? I believe that can be explained by the fact that the text in the first document is much shorter that in the other two, earlier we saw that the fitness values for each document is much different. The standard deviation is much smaller two, so such results are expected.

So far we have observed the best fitness values which is evaluation of chromosome belonging to one individual. Now, let's consider the population in general. I believe it is important to aim not only for 1 best solution but for the improvement of the population as a whole. The better population is, the higher chances of a successful crossover. This is influenced not only by the parameters like crossover rate and type but also things like by tournament k. For the next demonstration I used data gather from 5 runs on uniform order crossover rate 95%, mutation rate 15%, and k = 5. Consider the graph on figure 13 showing the trends of the average population and best values.

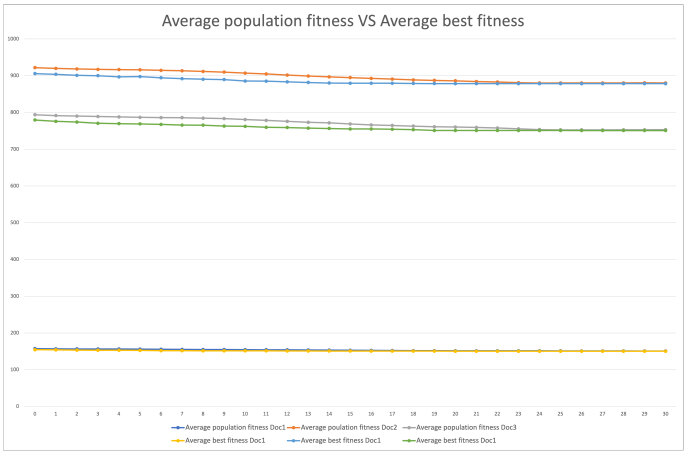

Fig. 13. Average trend population VS best fitness. Per 3 documents

As you can see this was a successful iteration because both population and best individuals improved over time. We can also see that eventually the population fitness caught up to the best fitness. Such information can be used for selecting parameters like population size and maximum generation size. I didn't use too large value for maximum number of generation so it didn't not affected the performance greatly but as you can see I could have used a smaller number of generations if needed without losing the best solution.

V. Conclusions

In conclusion, Genetic Algorithm is randomized but a good choice of parameters can greatly increase your chance of success. In this paper we have discussed program that included genetic algorithm based on fitness function that was evaluating the readability of the text or in other words tries to recognize English. We had the following parameters: population size, maximum number of generations, crossover procedure, crossover rate, mutation rate, tournament k. We ran the program 5 times for each set of parameters per 3 different input texts.

Statistics and graphs showed that context of the problem, in this case input text, have an impact on the outcome so there is no 'one fits all' when it comes to parameters.

Overall, I came to the conclusion that it is best to include mutation according to the T-test, and make sure that together crossover rate and mutation rate sums up to at least 100%. Most of the absolute lowest fitness values (per document) were obtained in variants a, d, and e.

Crossover procedure parameter brought controversial results so I believe it mostly depends on context of a problem. The absolute lowest values for document 1 and 2 were obtained through order crossover but there were no perfect results. However, with uniform order crossover we got perfect restoration of the document 3 with 3 different sets of parameters. Perhaps, my personal addition to creation of initial population also helped. It made sure that all initial chromosomes were created based on the fact that the first index should correspond to the index of the capital letter. Even though, it most likely did not improve the fitness value but increased the chance of having final solution corresponding to the appropriate beginning of a document.

References

[1] S. Russel, P. Norvig, "Artificial Intelligence A Modern Approach" 4th edition, Pearson Education Inc., 221 River Street, Hoboken, NJ07030