Development, Implementation, and Experimentation of Genetic Algorithms

Shwet Vasudevan
Computer Science Student
Brock University
St. Catharines, Ontario, Canada
sv18fi@brocku.ca

I. Introduction

Genetic Algorithms (GAs) are a searching technique that mimic evolution by incorporating similar methodologies found in biological evolution in order to derive a solution to a problem. The solution set for a GA can be represented as a sequence of values, known as a chromosome. A subset of a chromosome would be a smaller sequence of values, known as a gene. There can be multiple variations of a chromosome in one instance, some are identical while others are drastically different from one another, these sets of chromosomes as a whole can be considered a population. The size of a population is one of the many parameters in a GA. When a chromosome has genes that are similar to the optimal solution set it is considered "fitter" than a chromosome whose genes are not similar to the optimal solution set. The objective of a GA is to utilize the chromosomes in a population to create the fittest chromosome, this can be done by taking the current population and creating a new population. The differences between populations is known as a generation. The number of generations is another parameter in a GA. In order for a GA to create a new generation it must first identify the fittest chromosomes in the current generation, a process known as tournament selection. The number of chromosomes selected in a tournament is known as the reproduction value, another parameter used by the GA. Once these chromosomes have been identified some of their genes and some of the genes from a normal chromosome in the current population will be utilized to create a new generation in a process known as crossover. During a crossover the two chromosomes from the current generation are referred to as the parents, while the resulting new chromosomes are referred to as the children. The probability of a crossover occurring is known as the crossover rate, which is another parameter of the GA. Once the crossover is completed, there is a chance that the resulting children will undergo a mutation in which a section of their genes might be altered, allowing the offspring to evolve beyond their parents. The probability of a mutation occurring is known as the mutation rate, which is another parameter of the GA. The importance of crossovers and mutations is to ensure that future generations continuously evolve towards the solution set. In this paper, three experiments were conducted by developing and implementing a GA that would read a shredded document, represented as a text file, and derive the contents of the document in a legible format by utilizing tournament selection, reciprocal exchange mutation, ordered crossover, or uniformed ordered crossover. The objective of these experiments was to compare the performance of the GA's two crossovers and its ability to derive the solution set when given multiple types of shredded documents, and a variety of parameters.

II. Background

The GA for each experiment is considered to be an evolutionary algorithm [1] that has the ability to perform crossovers and mutation to evolve the solution set towards the optimal solution set. The basic structure for this GA is when the text file containing the shredded document data is selected and the user provides the necessary parameters the program commences. In the beginning state the GA produces a randomized initial population of chromosomes contained in an array. For these experiments each chromosome is represented as an integer array containing every number from 0 to 14. Once a population is generated, the fitness of each chromosome is evaluated and the fittest are selected using tournament selection. Next the GA evaluates the probability of preforming a crossover to generate the children of the succeeding generation. Once the children have been created, the GA evaluates the probability of preforming a mutation on each child. When the evaluation, crossover, and mutation is

completed the GA checks if the final generation parameter provided by the user is reached, if so, the fittest chromosome is displayed to the user and the current state of the document is displayed. If further generations are required, then the GA proceeds to continue the process of evaluation, crossing over, mutating, and adding new children to the succeeding generations until completion. The structure of this GA is illustrated in Figure 1.

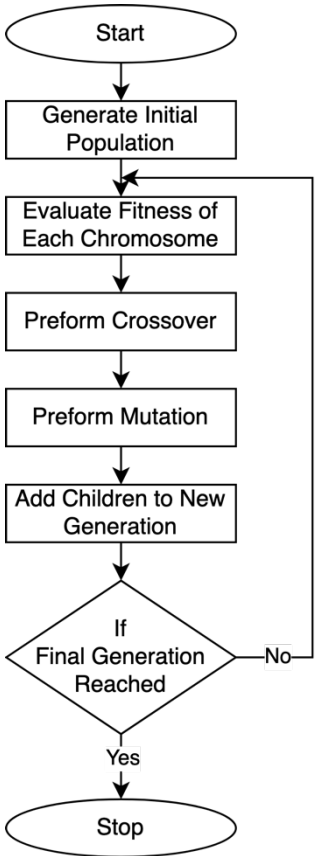

Figure 1. Illustration of the Genetic Algorithm used in an experiment

During execution of this program the GA is capable of preforming two forms of crossovers, an Ordered Crossover (OX), or an Uniformed Ordered Crossover (UOX). The choice of which crossover is determined during start phase when the program prompts the user for required parameters. During an OX, a chromosome from the tournament and a chromosome from the current population are chosen as the parents. Two random positions within the chromosome are chosen to represent the genes that each child is inheriting from their respective parents, (Child 1 inherits genes from Parent 1). Once inheritance is completed, the process of crossing over the genes from the other parent begins. This is done by ensuring that any values that are already contained within the opposite child are not copied over, only the remaining values. The process of an OX is illustrated in Figure 2.

Step 1	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
Step 2	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	C1: 3 4 5	C2: 7 3 6
Step 3	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	C1: 2 5 3 4 5	C2: 1 2 7 3 6
Step 4	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	C1: 2 5 3 4 5 1 7	C2: 1 2 7 3 6 4 5

Figure 2. Illustration of an Ordered Crossover with P representing Parent and C representing Child

The size and location of the section of genes in an OX is random, but once the section is determined the equivalent size and location is used for both children. In an OX the section contains every element from the start to the end point of the section. However, during a UOX the inheritance method differs. During an UOX, the method of selecting parents is similar, by selecting a chromosome from the tournament and a chromosome from the current population to be used as parents. Next, a mask is generated by creating a random binary array of equivalent size to that of each chromosome. Wherever there is a 1 in the mask, the genes and position of the genes are inherited by the child from their respective parent. Once inheritance is completed, all remaining genes are crossed over from the other parent in order from left to right, skipping any duplicate values and maintaining the position of the inherited genes. The process of an UOX is illustrated in Figure 3.

Step 1	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
Step 2	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	Mask: 0 1 0 0 1 1 0	Mask: 0 1 0 0 1 1 0
Step 3	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	Mask: 0 1 0 0 1 1 0	Mask: 0 1 0 0 1 1 0
	C1: 2 5 6	C1: 5 6 1
Step 4	P1: 1 2 3 4 5 6 7	P2: 2 5 7 3 6 1 4
	Mask: 0 1 0 0 1 1 0	Mask: 0 1 0 0 1 1 0
	C1: 7 2 3 1 5 6 4	C1: 2 5 3 4 6 1 7

Figure 3. Illustration of an Uniformed Ordered Crossover with P representing Parent and C representing Child

Once the crossover phase is completed the GA is capable of applying a Reciprocal Exchange Mutation (REM) on the new chromosomes. This is done by selecting two random genes in the chromosome and exchanging their positions. The process of a REM is illustrated in Figure 5.

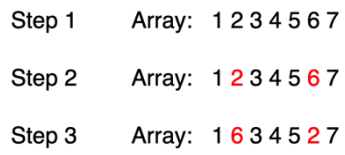

Figure 4. Illustration of a Reciprocal Exchange Mutation

The pseudocode for all algorithms can be found in the following sections.

Algorithm 1: *Genetic Algorithm*

```
1: Read problem instance data
2: Set GA parameters
3: Generate a random initial population
4: For gen=1 to MAXGEN do
5:       Evaluate fitness of each individual in
         population

6:       Select a new population using a
         selection strategy

7:       Apply crossover and mutation

8: End for
```

Algorithm 2: *Ordered Crossover Algorithm*

```
1: Create random double
2: Check if double < crossover rate
3: If true
4:       Select random fittest individual,
         parent1

5:       Select current individual from
         population, parent2

6:       Create child1 and child2 data structures

7:       Copy Section from parent1 to child1, and
         parent2 to child2

8:       Copy remaining values from parent2 to
         child2, and parent1 to child2

9:       Check if child1 or child2 mutates

10:      Add children to the new population
11: Else
12:      Select random fittest individual,
         parent1

13:      Select current individual from
         population, parent2

14:      Check if parent1 or parent2 mutates

15:      Add parent1 and parent2 to new
         population
```

Algorithm 3: *Uniformed Crossover Algorithm*

```
1: Create random double
2: Check if double < crossover rate
3: If true
4:       Select random fittest individual,
         parent1

5:       Select current individual from
         population, parent2

6:       Create random mask

7:       Create child1 and child2 data structures

8:       If position in mask is 1
9:                Copy Section from parent1 to
                  child1, and parent2 to child2

10:      If position in mask is 0
11:               Copy remaining values from
                  parent2 to child2, and parent1
                  to child2
12: Else
13:      Select random fittest individual,
         parent1

14:      Select current individual from
         population, parent2

15:      Check if parent1 or parent2 mutates

16:      Add parent1 and parent2 to new
         population
```

Algorithm 4: *Reciprocal Exchange Mutation Algorithm*

```
1: Create random double
2: Check if double < crossover rate
3: If true
4:       Select two random points in the array

5:       Swap values at the two points

6:       Return new array
7: Else
8:       Return array
```

III. Enperiment Setup

The purpose of these experiments was to compare the performance measures of the GA's implemented crossovers, and their ability to derive the solution set when given a text file containing the data of a shredded document. The GA developed for these experiments was written in Java 8 and developed using IntelliJ. The program can be run by compiling the Assign2.java file located in the src folder. When the program is compiled and ran, a file select window will be displayed to the user. Navigate from the home directory to the necessary text file to test the performance of the GA. In these experiments the documents that were used are located in the Examples folder located in the same file system as the src folder. Once the required text file is selected, the user will be

required to fill in the compulsory parameters into the console to perform the experiment. The parameters for this GA are listed in Table 1.

Table 1. Parameters used by Genetic Algorithm

Parameter	Definition
Text File	File containing the data representing the shredded document.
Population Size	Determines the number of chromosomes in each generation.
Number of Generations	Determines the number of generations for the GA to perform.
Crossover Rate	The probability of a crossover occurring. (Represented as a decimal)
Mutation Rate	The probability of a mutation occurring. (Represented as a decimal)
Reproduction Value	The number of fittest chromosomes from the current population to be used for creating the next generation by using tournament selection.
Crossover Type	Type of crossover to be used during execution. Choose between Ordered Crossover or Uniformed Crossover by entering 1 or 2, respectively.
Seed	Value used by the random number generator for the purpose of replicating results for future analysis.

For each experiment five trials were conducted per crossover. During the trials the population size (PS) was fixed at 200. Likewise, the Number of Generations (NG) was fixed at 200. As for the Crossover Rate (CR) they would vary from 1.0 to 0.8, and the Mutation Rate (MR) would also vary from 0.0 to 0.1. The Reproduction Value (K) was fixed at 5. The seeds used were randomly chosen for each trail. The details of these parameters used during a trial is displayed in Table 2.

Table 2. Parameters used during each trial (Excluding Crossover Type and Text File)

Trials	PS	NG	CR	MR	K	Seed
1	200	200	1.0	0.0	5	98165196516
2	200	200	1.0	0.1	5	756165131
3	200	200	0.9	0.0	5	32146563251632
4	200	200	0.9	0.1	5	654322165162
5	200	200	0.8	0.1	5	685431651544

The objective of each experiment was to determine the performance between OX and UOX. Therefore, all five trials were replicated using both OX and UOX as the Crossover Type (CT) parameter. Each experiment was repeated using different Text Files (TS), each containing a different shredded document (DS). This resulted in the generation of 15 graphs and tables detailing the OX and UOX performances when given various parameters.

IV. Results

As stated in the Experiment Setup section, there were three experiments conducted each with the objective to observe the performances of the OX and UOX when given various GA parameters. As a result, 15 graphs and tables were generated detailing the performances during each trial. Due to the NG size used in each experiment the raw data files of each trial will not be presented in this paper; they have been included in a separate folder called Experiment. Due to the format of this paper, the graphs of each of these experiments have been compressed. The original graphs are included and can be found under the Experiments folder.

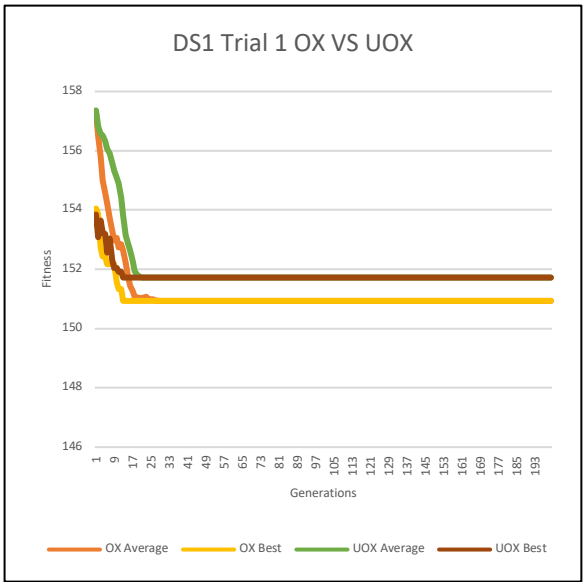

Figure 5. Graph 1

Table 3. Results for Graph 1

	OX Average	OX Best	UOX Average	UOX Best
Min	150.93	150.93	151.72	151.72
Max	157.30	154.05	157.36	153.85
Median	150.93	150.93	151.72	151.72
Standard Deviation	0.89	0.40	1.03	0.29

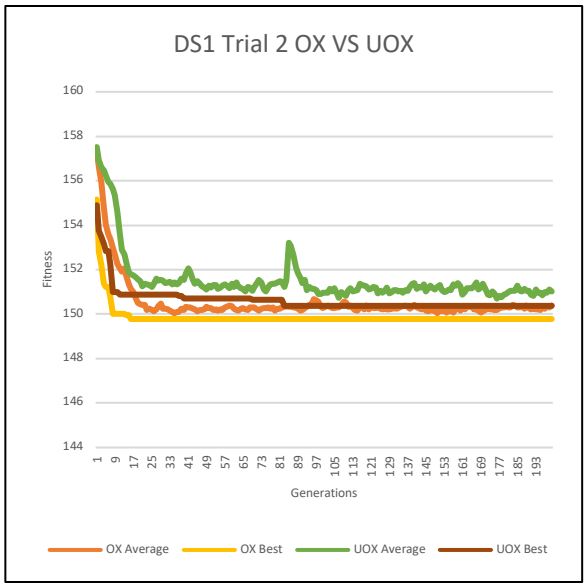

Figure 6. Graph 2

Table 4. Results for Graph 2

	OX Average	OX Best	UOX Average	UOX Best
Min	150.04	149.78	150.70	150.37
Max	157.36	155.16	157.52	154.90
Median	150.27	149.78	151.21	150.37
Standard Deviation	0.99	0.50	1.14	0.57

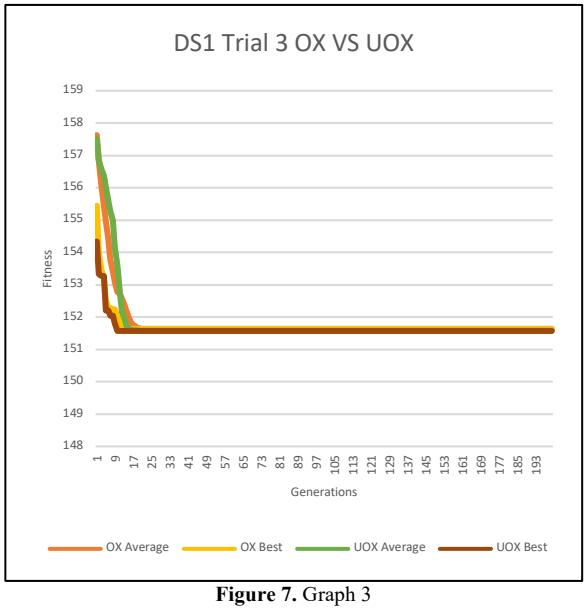

Figure 7. Graph 3

Table 5. Results for Graph 3

	OX Average	OX Best	UOX Average	UOX Best
Min	151.64	151.64	151.57	151.57
Max	157.63	155.44	157.51	154.34
Median	151.64	151.64	151.57	151.57
Standard Deviation	0.79	0.37	0.94	0.29

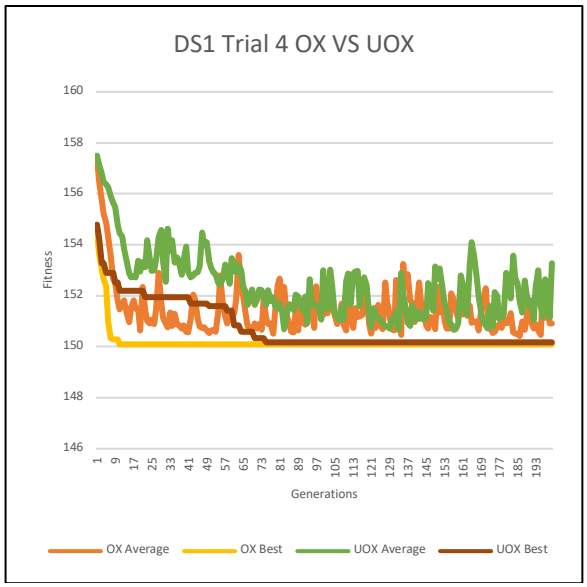

Figure 8. Graph 4

Table 6. Results for Graph 4

	OX Average	OX Best	UOX Average	UOX Best
Min	150.42	150.10	150.67	150.17
Max	157.42	154.51	157.50	154.79
Median	151.18	150.10	152.26	150.17
Standard Deviation	0.99	0.51	1.31	0.94

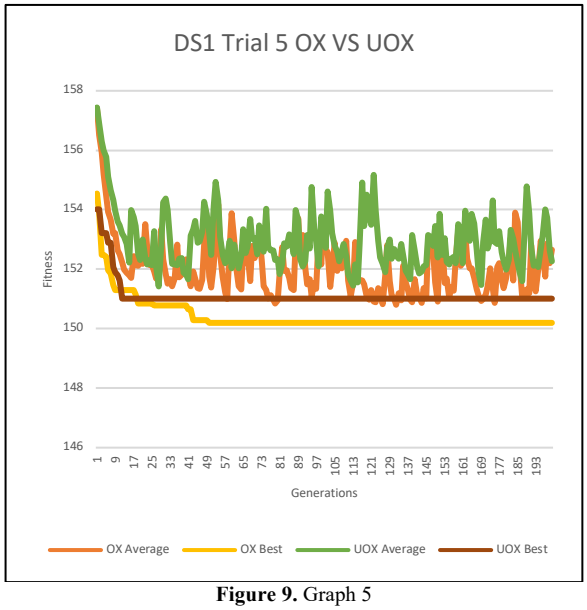

Figure 9. Graph 5

Table 7. Results for Graph 5

	OX Average	OX Best	UOX Average	UOX Best
Min	150.80	150.20	151.42	151.01
Max	157.41	154.54	157.44	154.01
Median	151.85	150.20	152.79	151.01
Standard Deviation	0.99	0.57	0.97	0.45

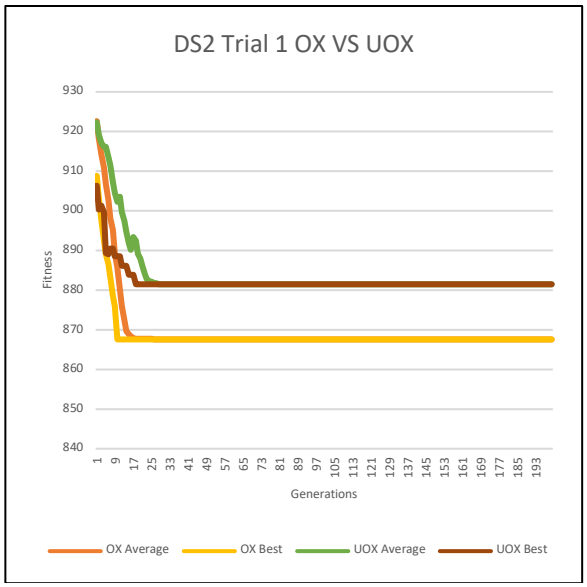

Figure 10. Graph 6

Table 8. Results for Graph 6

	OX Average	OX Best	UOX Average	UOX Best
Min	867.57	867.57	881.47	881.47
Max	922.62	908.82	922.32	906.34
Median	867.57	867.57	881.47	881.47
Standard Deviation	8.48	5.27	7.46	3.23

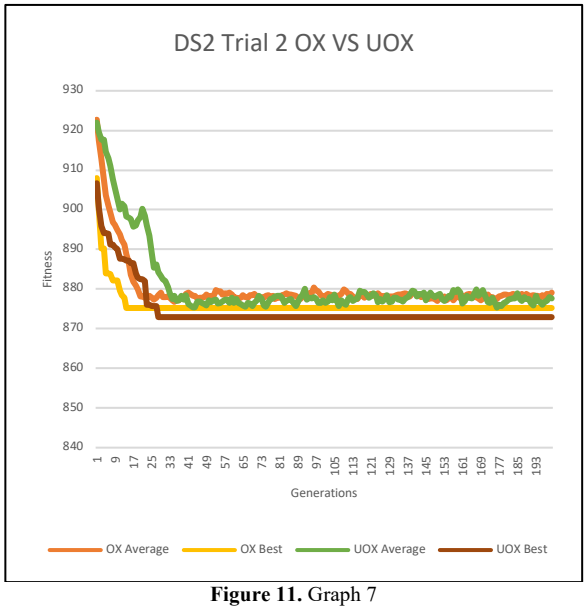

Figure 11. Graph 7

Table 9. Results for Graph 7

	OX Average	OX Best	UOX Average	UOX Best
Min	876.77	875.22	875.32	872.92
Max	922.70	907.99	921.98	906.68
Median	878.31	875.22	877.64	872.92
Standard Deviation	6.49	3.50	9.28	5.60

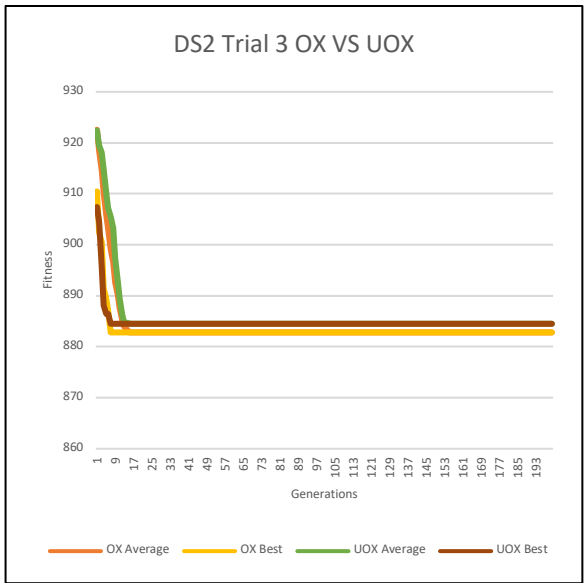

Figure 12. Graph 8

Table 10. Results for Graph 8

	OX Average	OX Best	UOX Average	UOX Best
Min	882.81	882.81	884.51	884.51
Max	922.57	910.48	922.41	907.41
Median	882.81	882.81	884.51	884.51
Standard Deviation	5.43	2.80	5.75	2.29

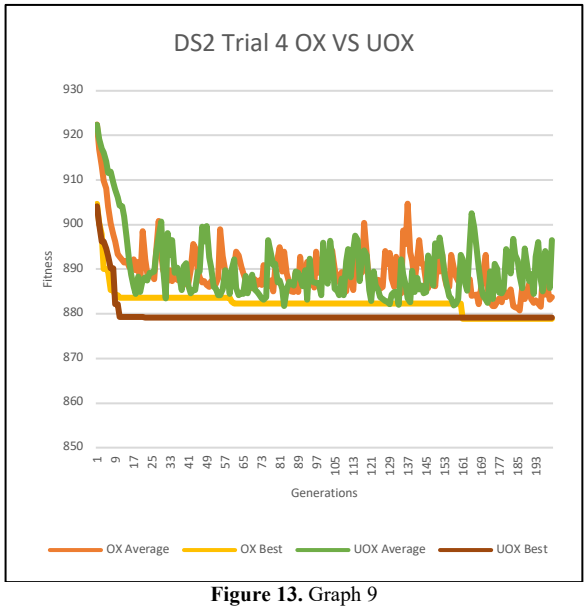

Figure 13. Graph 9

Table 11. Results for Graph 9

	OX Average	OX Best	UOX Average	UOX Best
Min	880.75	878.88	881.75	879.16
Max	922.44	904.65	922.37	904.10
Median	888.20	882.31	888.29	879.16
Standard Deviation	5.82	3.01	7.26	3.35

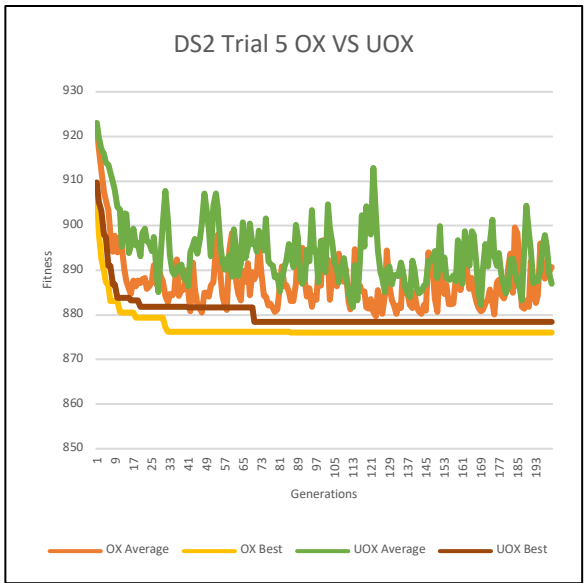

Figure 14. Graph 10

Table 12. Results for Graph 10

	OX Average	OX Best	UOX Average	UOX Best
Min	879.70	876.05	881.72	878.44
Max	922.50	908.57	923.04	909.70
Median	885.98	876.05	892.96	878.44
Standard Deviation	6.32	3.53	7.27	4.25

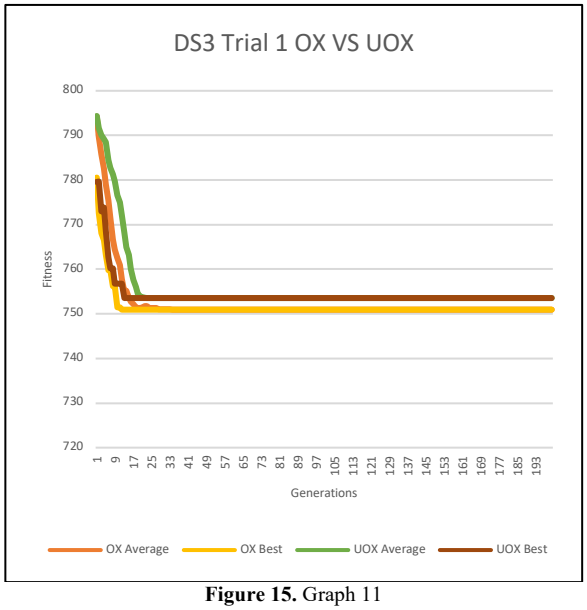

Figure 15. Graph 11

Table 13. Results for Graph 11

	OX Average	OX Best	UOX Average	UOX Best
Min	750.93	750.93	753.54	753.54
Max	794.23	780.55	794.35	779.66
Median	750.93	750.93	753.54	753.54
Standard Deviation	6.20	3.29	7.47	3.53

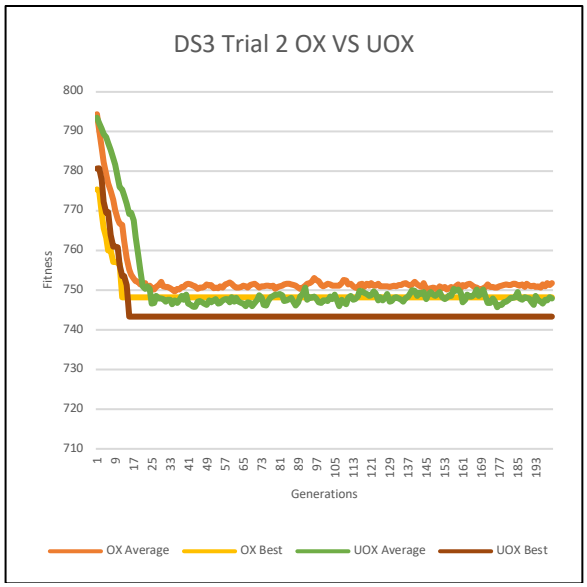

Figure 16. Graph 12

Table 14. Results for Graph 12

	OX Average	OX Best	UOX Average	UOX Best
Min	749.67	748.20	745.71	743.34
Max	794.36	775.41	793.49	780.73
Median	751.13	748.20	748.03	743.34
Standard Deviation	6.67	3.80	9.60	6.13

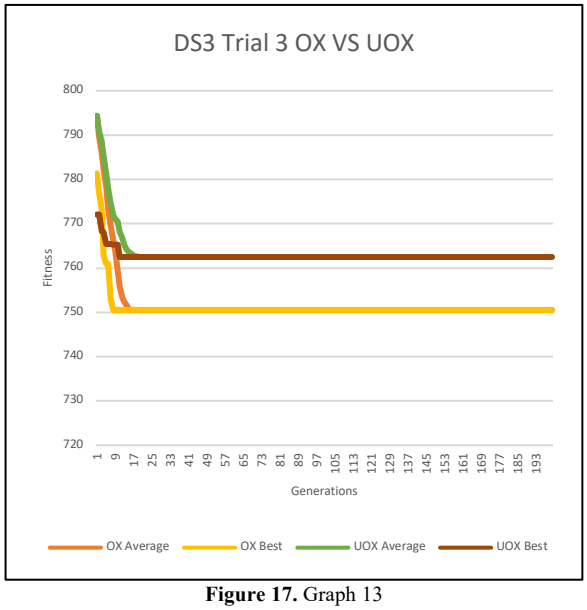

Figure 17. Graph 13

Table 15. Results for Graph 13

	OX Average	OX Best	UOX Average	UOX Best
Min	750.53	750.53	762.49	762.49
Max	794.37	781.35	794.34	772.07
Median	750.53	750.53	762.49	762.49
Standard Deviation	6.15	3.51	4.38	1.20

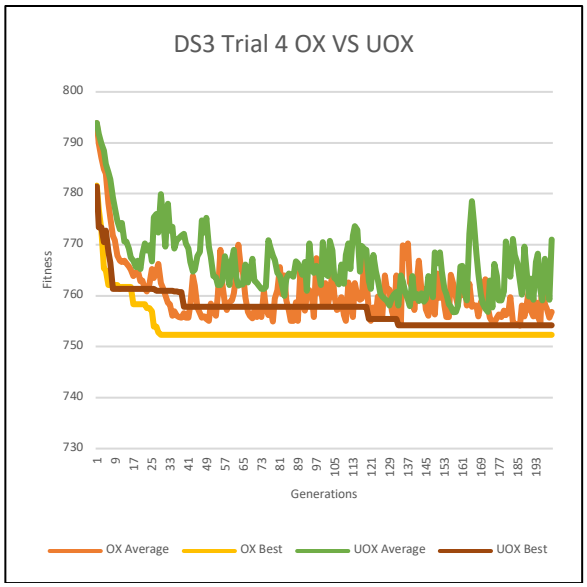

Figure 18. Graph 14

Table 16. Results for Graph 14

	OX Average	OX Best	UOX Average	UOX Best
Min	754.14	752.32	756.75	754.20
Max	793.86	781.61	793.90	781.42
Median	758.98	752.32	765.24	757.87
Standard Deviation	6.13	3.99	6.52	3.81

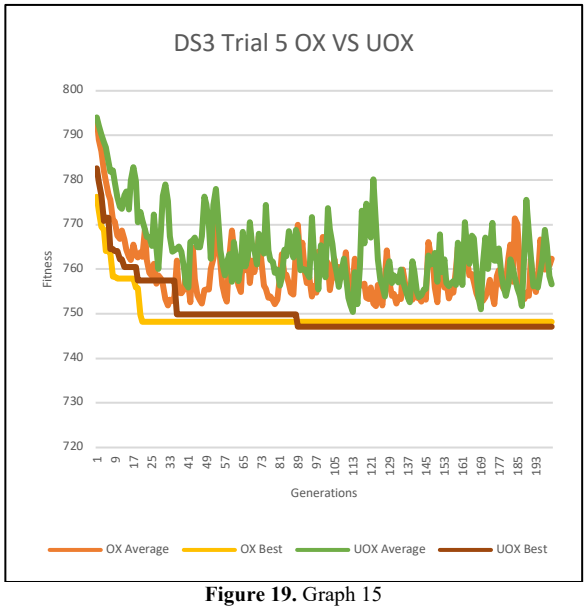

Figure 19. Graph 15

Table 17. Results for Graph 15

	OX Average	OX Best	UOX Average	UOX Best
Min	751.71	748.20	750.40	747.10
Max	793.69	776.23	794.03	782.61
Median	757.87	748.20	762.90	747.10
Standard Deviation	6.66	4.34	8.23	6.18

A. Analysis

As seen in the graphs there is a direct correlation between the MR and the performance of each crossover. Trials (T) completed with a MR of 0.0 (T1 and T3), resulted in both crossovers arriving at their maximum performance at an earlier generation, resulting in stagnation for every generation afterwards, however never reaching the optimal solution. There are exceptions, as seen in DS2 T1, where the OX arrived at the optimum. This is possible due to the seed and DS used in the trial since DS1 T1 and DS3 T1 both have OX performances that did not reach the optimum. The Ts completed with a MR of 0.1(T2, T4, and T5) have shown to continuously improve given a sufficiently large population and generation size. They have also out preformed all normal Ts with MRs of 0.0. This is due to the increase in variance, preventing the crossovers to stagnate. There is also a direct correlation between CR and the performance of each crossover. When comparing T1when the CR is 1.0 for all experiments it is evident that OX preforms better that UOX. However, in T3 when the CR is 0.9 the performance difference is comparatively smaller, excluding DS3. When comparing T2 and T4 where the CRs are different but the MRs are set at 0.1, the difference between OX and UOX is even smaller and in certain instances UOX outperforms OX as seen in DS2. T5 indicates that when the CR is 0.8 and the MR is 0.1 the performance difference between OX and UOX varies, but the overall performance of both crossovers is better than when the MR is 0.0. When comparing the averages of each crossover when the MR is 0.0, it is clear to see that the average line converges with the best line at an earlier generation, past the point where either crossover has reached their maximum performance, indicating stagnation. However, when the MR is greater than 0.0 the average lines continuously oscillate up and down, indicating that there is variation in some chromosomes in each generation. The greatest indicator of this is during each T5 experiment, the average lines display a greater magnitude when the CR is lower than in T4.

V. Conclusion

This paper depicted a Genetic Algorithm written in Java 8 and developed using IntelliJ that incorporates similar methodologies found in evolutionary biology by implementing a sequence of processes that derive an optimal solution to a problem by continuously evolving the chromosomes containing the solution sets towards the optimal solution set. These processes are conducted and completed based on the GA's parameters, which are set by the user. To get to the

solution set, the GA generates an initial random population of chromosomes. Then the GA applies tournament selection, crossovers, and mutations to determine the best solution set. The crossovers implemented in this GA are the Ordered Crossover, and the Uniformed Ordered Crossover. The mutation implemented in this GA is the Reciprocal Exchange Mutation. The purpose of this paper is to examine the GA's performance when given the task of reading a text file containing the data of a shredded document and creating the best solution set based on a variety of parameters. This was done by conducting three experiments. Each experiment compared the performances of the OX and UOX when the Crossover Rate and Mutation Rate are changed. Once the experiment was completed, graphs and tables were generated to visualize the performance measurements. Based on the data retrieved we can conclude that when the MR is 0.0 the performance of each crossover stagnates at a lower generation due to the lack of variation. This can be seen in the trials of T1, T3 for all DS graphs. When the MR is greater than 0.0, it introduces variation, thus improving the performances of both crossovers, as seen in T2, T4, and T5 for most of the DS graphs. When the CR is 0.9 both crossovers perform better than when the CR is 1.0, this consistent for all T3 and T1 trials. This is because more parents from the previous generation are passed to the new generation without being altered, this is allowing for more variation in the next generation. Overall, the performance of OX is usually better than UOX this can be attributed to how the algorithm preforms is crossovers. Since a section of consecutive fit genes in a chromosome can result in a better offspring compared to a random assortment of genes from random positions within a chromosome. However, with the right CR and MR and seed, there are instances where one crossover can out preform and even derive the optimal solution set in under 100 generations, though this is highly unlikely and cannot be consistently replicated using different parameters. We can assume based off of the data provided by T5 that when the CR decreases from 1.0 and the MR increases from 0.0 the amount of variance in each generation increases. This can be beneficial and a hindrance, since it is entirely possible for a sufficiently small CR and a sufficiently large MR to cause an entire generation to deviate from the optimal solution set and get worse due to too much variance.

REFERENCES

[1] B. Ombuki-Berman, "Genetic Algorithms: An introductory Overview".