Genetic Algorithms and their application in creating university class schedules

Parise, Nicholas 7242530
dept. of computer Science
Brock university
St. Catharines, Canada
np21ei@brocku.ca

I. Introduction

The university scheduling problem is a classic example of an optimisation problem. This problem is too computational complex to solve with brute force so a different method must be used. That solution being the use of a Genetic Algorithm. The following report shows the application of Genetic Algorithms in solving this optimisation problem, and the parameters chosen to maximise speed of convergence.

This problem is important so solve as hand crafting a schedule is time consuming and could produce results that are less than fit. By using a program to create a schedule, it allows resources to be efficiently allocated.

II. Background

A GA(genetic algorithm) simulates the process of natural selection to create solutions that get better through generations. At the end a solution created by a GA is almost indistinguishable from one created through bruteforce.

There are a lot of specialised terms with genetic algorithms, so the following is a list of terminology, how it's used, and what it does, along with the pseudo code of the algorithm.

1) **Chromosomes**: an array of Genes, in a GA a chromosome is seen as a solution to a problem
2) **Genes**: a collection of parameters. In this problem it is a 3 tuple of (course, room, timeslot)
3) **Population**: an array of chromosomes
4) **Fitness**: a float [0,1] that determines how good a solution is. 1 being a solution, less than 1 meaning an unfit solution. Calculated with the formula is F = 1/1+conflicts. Where a conflict is any time a time slot or professor overlaps in the same room or if the room's capacity is unfit for the students registered.
5) **Selection**: process in which we select 2 individuals to become parents. In this instance we are using tournament selection which is when we get K (4) chromosomes from the population and pick the 2 fittest individuals.
6) **Crossover**: in a GA it is how parents share genetic material to produce offspring. This is done randomly dependent on the crossover rate. When running the program there are two types of crossover that can be used:
   a) **Uniform**: randomly exchange genes between two parents
   b) **2 Point**: split chromosome and swap genes between two random points
7) **Mutation**: Introduce diversity into the chromosome by randomly changing one gene in a chromosome, dependent if the random value is less than the mutation rate
8) **Elitism**: Add 8% of the best individuals to the new population without performing any action on them to ensure they live on in the case of random disruption.

A. Parameters

When running the program there are a number of parameters that can be used to change how the algorithm behaves, the following are the parameters and the data it expects to get.

Population Size: 1...n
Max Generations: 1...n
Crossover Rate: [0,1]
Mutation Rate: [0,1]
Crossover Type: 'Uniform' or 'Two_Point'
Seed: -1 for random or VALUE for selected seed
Directory: 'string' the directory of the data
Example: java -cp build ClassScheduling.Main 1000 500 1.0 0.2 UNIFORM -1 t1

B. Pseudocode

Randomly initialise population P
For generation 0 to Max generation do:
Evaluate fitness of P
Perform Elitism (add best individuals to new population)
While new population ¡ population size
Select individuals using tournament selection
Apply Crossover to parents to get children
Mutate the offspring
Add to new population
End While
P = new population
End For

III. Experimental Setup

There were 30 trials ran for each of all the experiments

The following values were empirically found to result in the fastest convergence

population size = 200

max generations = 500

Elitism = 8 %

1) Crossover rate = 100%, Mutation = 0%
2) Crossover rate = 100%, Mutation = 10%
3) Crossover rate = 90%, Mutation = 0%
4) Crossover rate = 90%, Mutation = 10%
5) Crossover rate = 100%, Mutation = 20%

A. Implementation details

Programming language: Java

Seed: Random number but can be set to a desired number for reproducibility

Output: Per each generation: best fitness value, average population fitness value, Per each run: best solution fitness and its corresponding best solution chromosome

The program outputs .csv files to the data directory, inside this directory it is organised by the following format: data/name of input data/given Parameters/Seed.csv This way the data is nice and orderly, and can easily be found.

B. Notes

Included is also a simple utility to combine the created csv files into one csv per experiment to make it much easier to access the produced data. It can be ran with 'run-combine.bat'. This utility was used in combining all the information in excel.

IV. Results

The following is the table of T-Test values

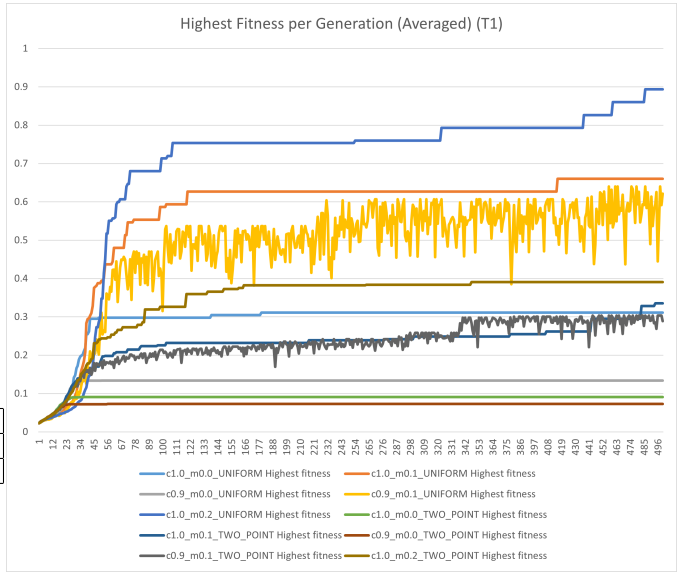
data	uniform vs two point	0 vs 20% mutation	100% vs 90% crossover
T1	3.6411E-169	2.0937E-214	4.70083E-21
T2	0.476192259	1.01372E-21	0.739370759

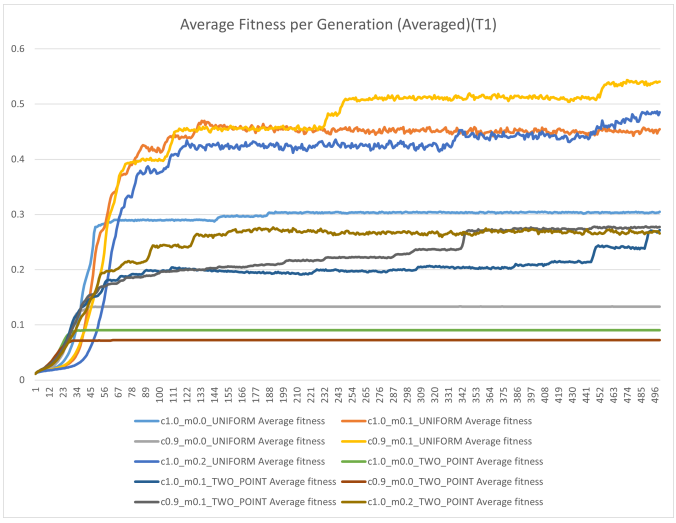
A. T1 Findings

Across the board the findings from this data suggests that:

- Uniform crossover performs significantly better compared to two points.
- A mutation rate of 0 is extremely unfavourable and has very poor performance
- The higher crossover rate of 100% is better than the crossover rate of 90%

Speed of convergence can be measured by the Highest fitness per generation. This is a very important metric to use to understand the differences in our findings, we can see both visually and mathematically the difference between the two methods. Running a TTest on 'C1.0 M0.2' both Uniform and Two Point we get the P-value of 3.6411E-169.This value is satisfactory low and we can reject the null hypothesis, allowing us to state that the difference between Uniform and Two Point are extremely statistically significant. Proving that uniform is better at converging faster. We can also see the importance of mutation in this data, by comparing 'C1.0 M0.0' and 'C1.0 M0.2' we can see a clear difference in our time to converge. The higher mutation in these tests clearly shows a faster speed and higher quality results. Running a TTest on this data we get the P-Value of 2.0937E-214. This value is satisfactorily low and we can reject the null hypothesis, allowing us to state that the difference between A mutation of 0 and a mutation rate of 0.2 are extremely statistically significant. Proving that a higher mutation rate is significant in causing faster convergence. Finally we can see the importance of high crossover values by comparing a crossover of 90% and 100%. We can see the significance by running a TTest on 'C1.0 M0.1' and 'C0.9 M0.1'. Running a TTest on this data we get the P-Value of 4.70083E-21. This value is less than 0.05 and we can therefore be significant in us rejecting the null hypothesis and allows us to conclude that the higher crossover rate does statistically make a difference in convergence. This P value is not as significant as our other P-values so we can conclude that a difference of crossover rate at these small values is harder to see a large difference in significance.





Parameters	type	Min	Max	Mean	Median	st.dev
c1.0 m0.0	High. fit.	0.023	0.311	0.291	0.311	0.061
	Avg fit.	0.012	0.305	0.278	0.303	0.070
c1.0 m0.1	High. fit.	0.023	0.660	0.571	0.627	0.156
	Avg fit.	0.012	0.470	0.404	0.450	0.121
c0.9 m0.0	High. fit.	0.023	0.134	0.129	0.134	0.019
	Avg fit.	0.012	0.133	0.127	0.133	0.024
c0.9 m0.1	High. fit.	0.022	0.640	0.479	0.524	0.145
	Avg fit.	0.012	0.544	0.435	0.505	0.140
c1.0 m0.2	High. fit.	0.023	0.893	0.696	0.753	0.214
	Avg fit.	0.012	0.487	0.380	0.425	0.128

TABLE I
UNIFORM (T1)

Parameters	type	Min	Max	Mean	Median	st.dev
c1.0 m0.0	High. fit.	0.023	0.091	0.089	0.091	0.009
	Avg fit.	0.012	0.091	0.088	0.091	0.013
c1.0 m0.1	High. fit.	0.022	0.335	0.233	0.239	0.053
	Avg fit.	0.012	0.272	0.193	0.199	0.044
c0.9 m0.0	High. fit.	0.022	0.073	0.072	0.073	0.006
	Avg fit.	0.012	0.073	0.071	0.072	0.009
c0.9 m0.1	High. fit.	0.023	0.304	0.231	0.230	0.059
	Avg fit.	0.012	0.278	0.217	0.222	0.058
c1.0 m0.2	High. fit.	0.023	0.390	0.344	0.382	0.088
	Avg fit.	0.012	0.277	0.241	0.266	0.061

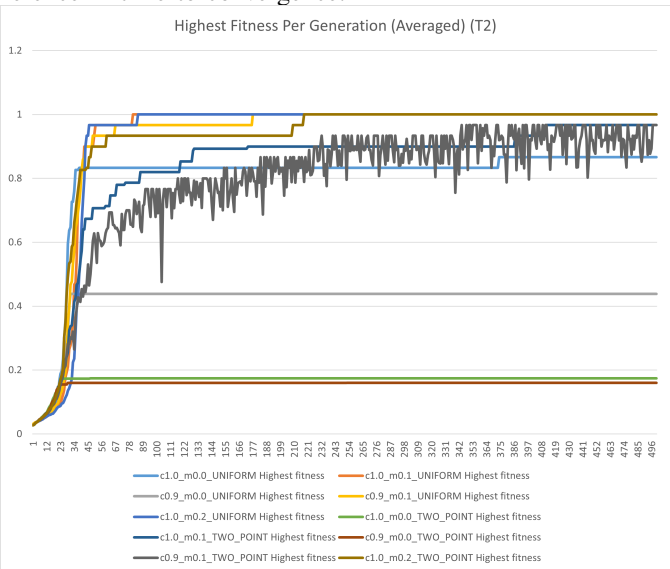
TABLE II
TWO POINT (T1)

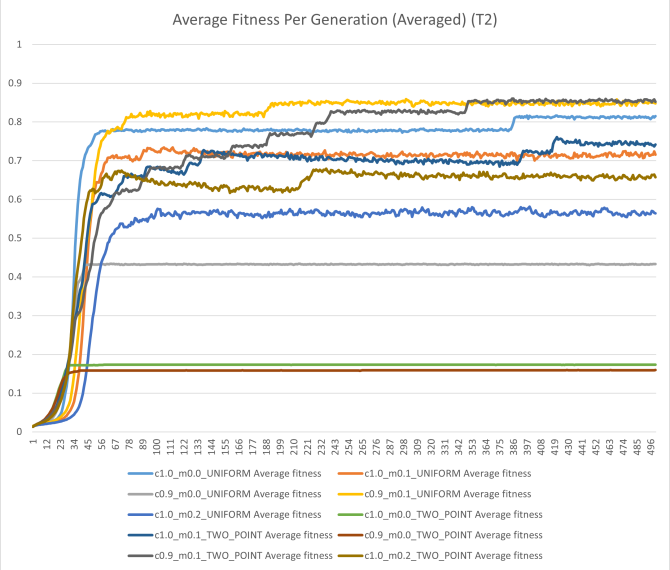
B. T2 Findings

Across the T2 data set the findings from this data suggests that:

- Uniform crossover and two points seem to be similar in terms of convergence rate.
- A mutation rate of 0 is extremely unfavourable and has very poor performance
- The crossover rate of 100% vs 90% is inconclusive and not significant.

Speed of convergence can be measured by the Highest fitness per generation. This is a very important metric to use to understand the differences in our findings, we can see both visually and mathematically the difference between the two methods. Running a TTest on 'C1.0 M0.2' both Uniform and Two Point we get the P-value of 0.4762. This value is unsatisfactory to reject the null hypothesis. The differences between Uniform and Two Point are not statistically significant. We can also see the importance of mutation in this data. By comparing 'C1.0 M0.0' and 'C1.0 M0.2' we can see a clear difference in our time to converge. The higher mutation in these tests clearly shows a faster speeds and higher quality results. Running a TTest on this data we get the P-Value of 1.01372E-21. This value is satisfactorily low and we can reject the null hypothesis, allowing us to state that the difference between A mutation of 0 and a mutation rate of 0.2 are extremely statistically significant. Proving that a higher mutation rate is significant in causing faster convergence. Finally we can discuss the inconclusive test results of the difference between 90% and 100% crossover values. We can see the significance by running a TTest on 'C1.0 M0.1' and 'C0.9 M0.1'. Running a TTest on this data we get the P-Value of 0.7394. This value is greater than 0.05 and therefore we have to conclude that the higher crossover rate does not make a statistically significant difference in time to convergence.





Parameters	type	Min	Max	Mean	Median	st.dev
c1.0 m0.0	High. fit.	0.027	0.867	0.800	0.833	0.171
	Avg fit.	0.013	0.817	0.735	0.779	0.183
c1.0 m0.1	High. fit.	0.027	1.000	0.932	1.000	0.226
	Avg fit.	0.013	0.734	0.656	0.715	0.184
c0.9 m0.0	High. fit.	0.028	0.438	0.419	0.438	0.079
	Avg fit.	0.013	0.434	0.408	0.432	0.091
c0.9 m0.1	High. fit.	0.031	1.000	0.927	1.000	0.217
	Avg fit.	0.015	0.859	0.769	0.845	0.213
c1.0 m0.2	High. fit.	0.027	1.000	0.927	1.000	0.236
	Avg fit.	0.013	0.581	0.510	0.563	0.152

TABLE III
UNIFORM (T2)

V. Discussions and Conclusions

There were a wild array of experiments performed with different parameters to analyse the significance of different parameters in solving this optimisation problem. By only

Parameters	type	Min	Max	Mean	Median	st.dev
c1.0 m0.0	High. fit.	0.027	0.174	0.170	0.174	0.022
	Avg fit.	0.013	0.174	0.167	0.173	0.027
c1.0 m0.1	High. fit.	0.027	0.967	0.836	0.900	0.204
	Avg fit.	0.013	0.761	0.654	0.702	0.165
c0.9 m0.0	High. fit.	0.028	0.160	0.156	0.160	0.019
	Avg fit.	0.013	0.159	0.153	0.159	0.024
c0.9 m0.1	High. fit.	0.027	0.967	0.788	0.854	0.212
	Avg fit.	0.013	0.860	0.723	0.823	0.205
c1.0 m0.2	High. fit.	0.027	1.000	0.917	1.000	0.206

TABLE IV
TWO POINT (T2)

varying one parameter at a time we can see the difference between parameters more accurately. The TTests are run on the averaged values from a larger dataset.

A. Key observations:

1) Crossover methods:: Uniform significantly outperformed Two Point on the first dataset, but on the second dataset uniform and two points were not as conclusive. Its possible crossover wasn't as big of an issue in T2 because the dataset was so much easier to solve than T1. Even graphically T2 is visually so much easier to solve, so perhaps uniform is still better, but only on harder datasets where it takes more generations to solve.

2) Mutation Rate:: A mutation rate of 0% was shown to result in poorer performance across all the configurations. This seems to be due to chromosomes getting 'stuck' genes where having a certain room or timeslot will never succeed and will always cause conflicts. Since these cannot be changed they have a hard time converging if the initial random values are not ideal. This seems to be the reason why both uniform and two point struggle with 0 mutation rate.

3) Crossover Rate:: A crossover rate of 100% provided better results that 90% in the T1 dataset, but was not statistically significant in T2. This shows that crossover rates are very dependent on the dataset than other parameters are and must be tailored specifically.

B. Discussion:

These results really showed how dependent the dataset is when it comes to parameters. Mutation rate was proven to be significant independent of the dataset, but Crossover type and crossover rate were not statistically significant in all datasets. More datasets could have been used to determine if T1 is a statistical anomaly with crossover type and rate mattering as much as it did. Since this paper only deals with two datasets its is hard to conclude if Crossover type and rate matters for more or less than half, but with only three sets it is impossible to tell.

Overall, this experiment shows that Uniform crossover is a good place to start when solving a problem. It also shows that the Crossover rate should remain relatively high between 0.9 - 1 to get decent results.Finally the data suggests that mutation rate must be at least 0.1 and in some cases 0.2 could result in good results too. The main conclusion is that a mutation rate of 0 results in horrible results and should be avoided at all costs.

C. Problems with data and results.

The biggest problem with the algorithm as it stands is that it doesn't take into account courses people are usually required to take for a degree. For instance a philosophy major might need to take philosophy and sociology in a semester but this algorithm could allow for both mandatory courses to be at the same time, making it impossible for a student to successfully take them in one semester.

A fix might be to introduce a mandatory flag to courses and in the fitness function make sure that mandatory courses cannot be at the same time. This might result in longer solution times, and in the worst case if all the courses are mandatory and there aren't enough rooms / timeslots then it will be impossible for the program to make a schedule.

In such cases another solution might be tagging each course with a major and specifically not scheduling mandatory courses with a similar major together. This could be done with a similarity matrix that would solve the issue of people taking minors in fields in completely different disciplines.

REFERENCES

[1] McLean, R. (2024) COSC 3P71 A2: Genetic Algorithm.

[2] DeGregorio, A. (2024) *How to write to a CSV file in Java*, Baeldung. https://www.baeldung.com/java-csv