Formulating the Optimal School Schedule Parameter Sets Using Genetic Algorithms

Mitchell Adamson
COSC 3P71
Brock University
St. Catharines, Canada
Email: ma18hr@brocku.ca

***Abstract*—With the increased applicability of genetic algorithm implementations over the last 50 years, we have come up with numerous applications for the algorithms such as the design of aircraft which utilizing information such as atmospheric pressure and chemical properties from the environment the aircraft operates in. A new problem that is needed to be solved is the class scheduling problem. This paper seeks to investigate this problem, and see how different constraints and parameters of a genetic algorithm can affect its performance in the solving of the Class Scheduling Problem. In order to answer this question, this paper employed many different parameters in order to compare the fitness values and possible school schedule solutions generated by the genetic algorithm. The results encountered showed that is is possible to design a genetic algorithm with sufficient parameters resulting in optimal solutions of a no conflict schedule.**

I. Introduction

The issue of scheduling classes is an intriguing topic discussed in Artificial Intelligence classes. This task requires matching courses with available Rooms and Timeslots adhering to limitations such as Room size limits, the inability to have two classes in the same Room simultaneously, and the inability for teachers to be scheduled multiple classes simultaneously. Genetic algorithms, which draw inspiration from natural selection, present a strong solution by developing a group of tentative schedules over time, aiming for the best possible arrangement. The research completed delves into how well genetic algorithms can handle this scheduling challenge, focusing on how different settings like crossover methods (specifically, uniform versus one-point) and mutation rates(randomizing a gene) affect the final schedule quality. Through this work, I intend to demonstrate empirically how genetic algorithms can be fine-tuned for real-world educational scheduling, highlighting their capability to effectively navigate the complexities of such tasks.

II. Background

Using the university class scheduling problem described in the introduction, this research project covers important concepts concerning the evolution of a set of solutions using both uniform(50 percent chance to copy genes from parent 1 or parent 2) and one-point(select a portion of the solution to swap genes) crossover methods. Different parameters such as the crossover and mutation rates are used in order to control the intensity of gene swaps and creations. The concept of elitism, where the elitism rate is used to select the 'x' the best solutions for generation x, whereupon they are carried over to generation 'x + 1'. Each solution of a course, Room and Timeslot were encoded into a chromosome, allowing the manipulation of the solution across generations.

III. Experimental Setup

In order to come up with solutions to the class scheduling problem, a population size of 100 solutions was used, where each solution is a tuple consisting of a course index, a Room index and a Timeslot index. These solutions were evolved across 150 generations, where an evolutionary algorithm was applied to the population each generation. This evolutionary algorithm sorts the population in descending order of fitness values which are calculated to be the number of conflicts within a solution, where 1.0 means no conflicts. Next the elitism rate is sued to preserve the 'x percent' of elites to retain for the next generation. Following this, a tournament selection heuristic is implemented to allow 4 individuals from the population to compete by comparing their fitness values. The two top fitness value solutions are then bred using crossover methods to create two children for the next generation. The children are then mutated, where there is a 50 percent chance of randomly mutating either the Room or Timeslot index. The new children are added to the population and the evolutionary process is applied again until the max generations of 150. For each parameter set, 5 experiments were run with a calculation of the statistical summary over the five different seeded runs.

IV. Results

The results have shown that for both datasets (t1, t2) the uniform crossover method with a crossover rate of 1.0 and a mutation rate of 0.1 was the most successful parameter set with a mean best fitness of 0.5676 per generation over both datasets as well as a Median Fitness value of 1, indicating over half the generations evolved were optimal solutions, the highest rate out of any parameter set tested. Across 20 different parameter sets, 7 of them reached a max fitness value of one, indicating effectiveness a somewhat general effectiveness. The custom parameter set of crossover rate equal to 0.75 and mutation rate equal to 0.25. This custom parameter set did not perform well scoring the lowest in mean best fitness among the parameter

set ratings under uniform and one point crossover methods respectively.

Configuration	Mean Best Fitness	Max Fitness	Min Fitness	Std. Dev. Fitness	Median Fitness
uniform, 1.0, 0.1	0.4555	1.0000	0.0435	0.3049	0.5000
uniform, 0.9, 0.1	0.3733	1.0000	0.0435	0.2858	0.3333
uniform, 1.0, 0.0	0.2865	0.5000	0.0476	0.1158	0.2500
uniform, 0.9, 0.0	0.1630	0.2500	0.0476	0.0521	0.1667
uniform, 0.75, 0.25	0.0861	0.1429	0.0455	0.0195	0.0833
one_point, 1.0, 0.1	0.2834	0.5000	0.0455	0.1497	0.2500
one_point, 0.9, 0.1	0.2295	0.5000	0.0476	0.1126	0.2000
one_point, 1.0, 0.0	0.1130	0.1429	0.0455	0.0194	0.1250
one_point, 0.9, 0.0	0.1053	0.1250	0.0476	0.0184	0.1111
one_point, 0.75, 0.25	0.0882	0.1250	0.0435	0.0196	0.0833

TABLE I
SUMMARY STATISTICS OF FITNESS ACROSS DIFFERENT GA CONFIGURATIONS, SORTED BY CROSSOVER TYPE - T1 PROBLEM SET

Configuration	Mean Best Fitness	Max Fitness	Min Fitness	Std. Dev. Fitness	Median Fitness
uniform, 1.0, 0.1	0.6797	1.0000	0.0500	0.3744	1.0000
uniform, 0.9, 0.1	0.5964	1.0000	0.0556	0.3498	0.5000
uniform, 1.0, 0.0	0.4281	1.0000	0.0526	0.2725	0.3333
uniform, 0.9, 0.0	0.2190	0.3333	0.0556	0.0653	0.2000
uniform, 0.75, 0.25	0.1173	0.1667	0.0526	0.0292	0.1111
one_point, 1.0, 0.1	0.5562	1.0000	0.0526	0.3232	0.5000
one_point, 0.9, 0.1	0.4627	1.0000	0.0526	0.3185	0.3333
one_point, 1.0, 0.0	0.1661	0.2000	0.0500	0.0309	0.1667
one_point, 0.9, 0.0	0.1422	0.1667	0.0476	0.0242	0.1429
one_point, 0.75, 0.25	0.1151	0.2000	0.0500	0.0294	0.1111

TABLE II
SUMMARY STATISTICS OF FITNESS ACROSS DIFFERENT GA CONFIGURATIONS, SORTED BY CROSSOVER TYPE - T2 PROBLEM SET

V. DISCUSSIONS AND CONCLUSIONS

To conclude, the heuristic methods attempted to optimize the class scheduling problem were fairly effective overall with the most effective solution being the combination of the lowest crossover rate attempted(0.9) along with the highest official mutation rate attempted (0.1) using the uniform crossover method. Typically the parameter sets with zero mutation applied performed worse than those with mutation applied. This highlights the significance of randomness in the selection of a new Room or Timeslot index. Overall it seems the uniform crossover method was more effective than the one point crossover method. Although graphs of the parameter sets are not included in the LaTex report, there is a file in the submission which contains a set of graphs modeling each generation among the five alongside one another.

REFERENCES

NA