ARTIFICIAL INTELLIGENCE

PROGRAMMING ASSIGNMENT 1

RAJVI SAMPAT

2018B4A70820G

GENETIC ALGORITHM

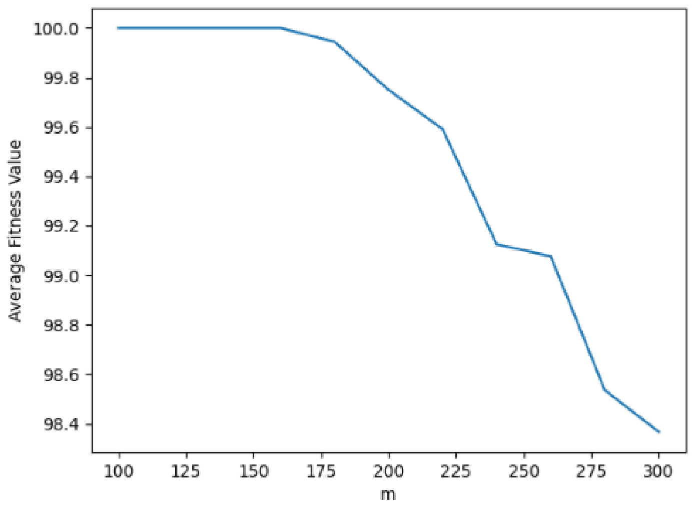
The Genetic Algorithm is used to maximise the number of satisfied clauses in a given 3-CNF sentence. The given sentence has 50 variables. Hence, there are 2^{50} models in the state space. An individual is a model represented as an array of length 50 where the value at each index represents the True/False value assigned to the corresponding literal in the model.

A population consisting of 50 randomly generated individuals is used as the starting population for this algorithm. A corresponding fitness value is found for each individual, which is the number of clauses it satisfies in the given sentence. The individual with the best fitness value in a given generation is found. To create the next generation, two parents are stochastically chosen from the population, with weights corresponding to the fitness value of an individual. In this model, two methods of reproduction are used to create the children. The first involves randomly choosing a crossover point such that the values of the child before the crossover point are taken from one parent and the values after it are taken from the other parent. The second method involves creating a random sequence of values to be taken from one parent and taking the rest from the other parent. Since two children are created during reproduction, the child with the better fitness value is added to the new generation. The child is then mutated with a small probability by negating the value at a random index. In this model, elitism is used to retain 10% of the individuals in the new population and culling is used to stochastically choose 90% of the new generation in the new population.

Random restart is used by restarting the algorithm with a new population of randomly generated individuals if the maximum fitness value of a generation remains constant for 100 consecutive generations. The algorithm is terminated if a model satisfying all the clauses is found, or if the algorithm exceeds 44 seconds (ensuring the program terminates within 45 seconds).

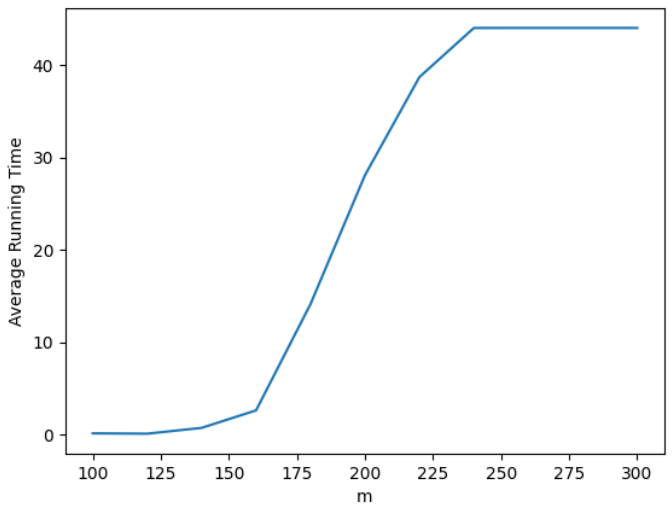
1. AVERAGE FITNESS FUNCTION

It is observed that the average fitness function is more for smaller values of m and decreases for higher values of m. As the sentence size increases, it becomes more difficult to satisfy it.



2.AVERAGE RUNNING TIME

It is observed that the algorithm is faster for smaller values of m and the average running time increases for larger values of m. The program runs for 45 seconds unless all the clauses are satisfied earlier. Since random restart is used, once the program converges to a fitness value, random restart is done instead of terminating it.



3. IMPROVING THE GENETIC ALGORITHM

A. REPRODUCTION:

The standard method for reproduction involves randomly choosing a crossover point and creating a child model of the new generation by selecting the values of one parent up to the

crossover point and the values of the other parent after the crossover point. This method helps to preserve a block, or a subarray of literals of a parent with a good fitness value in the next generation. However, it can be observed that a specific sequence of literals does not impose any structure on the model. Hence, if the sequence is randomly generated instead of splitting the array using a crossover point, there is an equal probability of choosing a certain number of literals from a parent with a good fitness value. This method does not restrict the algorithm to choosing either the beginning or ending subarray of the parent, and helps to widen the search space. The information from the parents is combined and preserved. Since two children are obtained in this process, taking values of each literal from opposite parents, the child with the better fitness function is added to the new generation.
To test these methods, each of the sentences from the CSV files were run 5 times against the two methods and a combination of the two. The average time and percentage of clauses satisfied over the 5 iterations was recorded. The other parameters were kept constant.
Mutation rate – 10%
Population size – 50
Elitism and Culling – 10% old generation(best), 90% new generation(stochastically)

Reproduction Method	CNF1		CNF2		CNF3		CNF4		CNF5	
Crossover : Random Sequence	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time
100 : 0	100%	0.45s	99.6%	36.5s	99.92%	24.12s	98.4%	44.017s	98%	44.016s
90 : 10	100%	0.134s	99.5%	44.01s	99.92%	14.45s	98.667%	44.01s	98.057%	44.013s
50 : 50	100%	0.057s	99.9%	20.349s	100%	9.985s	98.6%	44.013s	98.228%	44.016s
0 : 100	100%	0.0647s	99.9%	20.12s	100%	7.125s	98.667%	44.012s	97.228%	44.012s

Since adding the random sequence method is seen to improve the average number of clauses and decrease the average time, a combination of both the methods is chosen. The algorithm randomly chooses which of the two methods to follow in each reproduction step, with equal probability (50:50).

B. <u>MUTATION:</u>
Mutating some of the children helps to increase the diversity of the population and hence widen the search space. It helps the algorithm escape the local maxima. To mutate a child, an index is randomly chosen and the value at that index (i.e. the Boolean value of that literal) is flipped. It is observed that increasing the mutation rate improves the algorithm. This indicates the abundance of local maxima. However, the mutation rate cannot be too high as it will make the algorithm resemble random search; the information of the parents will not be preserved.
To find the best mutation rate, each of the sentences from the CSV files were tested with different mutation rates. The average time and percentage of clauses satisfied over 5 iterations against each rate was recorded. The other parameters were kept constant.
Reproduction method – 50% crossover, 50% random sequence
Population size – 50
Elitism and Culling – 10% old generation(best), 90% new generation(stochastically)

Mutation Rate	CNF1		CNF2		CNF3		CNF4		CNF5	
	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time
1%	100%	0.055s	99.6%	43.09s	99.76%	32.59s	98.53%	44.017s	97.82%	44.011s
5%	100%	0.067s	99.7%	38.76s	100%	6.36s	98.53%	44.013s	98.17%	44.017s
10%	100%	0.057s	99.9%	20.349s	100%	9.985s	98.6%	44.013s	98.228%	44.016s

C. <u>RANDOM RESTART:</u>

The method of random restart is used if the maximum fitness value of a generation remains constant for 100 consecutive generations. This indicates the algorithm is likely to have reached a local maximum. A new population is created which is independent of the previous generations. It provides a fresh starting point for the algorithm and increases the chances of finding the global maximum. The best model over all the iterations is stored so that when the program terminates the best model will be printed. Applying random restart significantly improved the algorithm as shown below.

Comparison of the average percentage of clauses satisfied (over 5 runs) with and without random restart

	CNF1	CNF2	CNF3	CNF4	CNF5
With Random restart	100%	99.9%	100%	98.67%	98.228%
Without Random restart	100%	99.8%	99.67%	98.33%	97.94%

D. <u>ELITISM AND CULLING:</u>

A combination of elitism and culling has been used to generate the new population for the next iteration. Since elitism is used, the individuals with the best fitness value are retained in the next population, which enables us to use the information from the good models in many more generations. Culling is used to reduce the size of the new population. However, this is done stochastically. Since a majority of the new generation is being chosen, most of the good individuals produced will be passed on. However, if only the best individuals are chosen, the individuals having low fitness value will be left out, making the approach greedy. The individuals with low fitness value may eventually reproduce to produce individuals with higher fitness value, and can widen the search space. Hence, they are chosen stochastically, with weights proportional to their fitness values. 10% of the older generation is chosen and 90% of the new generation.

E. <u>POPULATION SIZE:</u>

If the population size is very small, the search space also becomes small as the initial diversity is low. Hence it is more likely for an algorithm to get stuck at a local maximum and more difficult to find the solution. If the size is too high, the algorithm runs slowly and very few generations are iterated over. A population size of 50 is chosen

Popln_size	CNF1		CNF2		CNF3		CNF4		CNF5	
	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time	Average Clauses	Average Time
20	100%	0.069s	99.7%	31.53s	99.67%	35.0499s	98.4%	44.006s	97.88%	44.005s
50	100%	0.057s	99.9%	20.349s	100%	9.985s	98.6%	44.013s	98.228%	44.016s

4. LIMITATIONS OF GENETIC ALGORITHM

From the above graphs, we can see that as the size of the sentence (number of clauses) increases, the performance of the algorithm goes down. When the number of clauses is large, the number of local minima also increase. Hence it is more difficult for the algorithm to find the global maxima as it is more likely to get stuck at a local maximum. Thus, the performance of the genetic algorithm decreases with increase in number of local maxima.

5. SATISFIABILITY OF A 3 CNF SENTENCE

As the number of clauses in a 3CNF sentence increases, the number of clauses satisfied by the genetic algorithm decreases. This is also due to the lower satisfiability of the sentence with increased number of clauses, as there is a higher probability for a group of clauses to be conflicting, i.e. it is impossible for them to be satisfied at the same time. For example,

$(a \lor \neg b \lor c) \land (a \lor b \lor \neg c) \land (a \lor b \lor c) \land (a \lor \neg b \lor \neg c) \land (\neg a \lor b \lor c) \land (\neg a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor c) \land (\neg a \lor \neg b \lor \neg c)$ cannot be satisfied for any assignment of a, b, c