Name: Mehboob Ali

Roll Number: 17L-4316

Course: Artificial Intelligence

Section: E

Assignment: Exam Schedule Generation by Searching (Local and Evolutionary)

Introduction

Genetic algorithms are widely used for solving scheduling problems. In this particular assignment, it has been used along with Hill Climbing to reduce the likelihood of premature convergence or local optima. This modified version of Genetic algorithm is often known as **memetic algorithm** (MA). The local search could be used at any stage of the genetic algorithm. It has been used after Genetic Algorithm as instructed in the problem statement. However, other methods were also tested.

Methodology

General

Several parameters are passed to a Genetic Algorithm, e.g. population size, crossover rate, mutation rate, elitism rate, tournament size (if tournament selection used) and maximum number of generations allowed. The optimal value for these parameters is determined by experimentation as they vary from problem to problem.

Genetic Algorithms must have a stopping criteria based on which the search should be stopped. There exist multiple stopping criteria's. In this problem several of them have been used, e.g.

- Current generation count becomes greater than maximum generations allowed.
- The fitness value of the fittest individual does not improve for the last say 100 generations. It could be any arbitrary number.
- If fittest individual of a generation reaches the maximum fitness value, i.e. zero.
- If the fittest individual does not violate any of the hard constraints. In that case, fitness value might not be zero.

Chromosome Representation

A chromosome is an Array of integers whose size is equal to the total number of courses which have to be scheduled. Each index in the Array contains a value which is basically the slot number. These slot numbers are randomly generated by keeping total number of slots as the upper bound. Total number of slots is equal to total exam days multiplied by total slots available.

This representation is quite simple and effective for crossover and mutation operations. Additionally, it ensures that there are no duplicate exams which could have been the case if a different representation was used. However, for fitness evaluation it is converted to a 2D-Array where rows are the days and columns are the slots.

Population

When the Genetic Algorithm commences, a population is created with random individuals. Each random individual is created by assigning each course a random slot number. Slots are assigned until no course is left. Then, fitness of the population is calculated and the program enters the main loop which runs until one of the termination condition is met.

Selection

During selection phase, two individuals are selected to produce children. There are several selection methods, e.g. Roulette Wheel Selection, Tournament Selection, Random Selection, Elitism and Ranked Based Selection. In this problem, two selection methods have been used. Those include Tournament Selection and Elitism.

In Tournament Selection, random individuals from the population are added to the tournament after shuffling. The amount of individuals added depend on the tournament size which is passed as a parameter. The fittest individual from the tournament is selected as the parent for breeding. The tournament is run twice to get two parents for crossover.

Elitism has been used to make sure that the fittest individuals are retained for the creation of next generation. It provides a means for reducing genetic drift. It will be applied to prevent losing the best chromosome because it significantly improves the Genetic Algorithm’s performance.

In this problem, these two methods have been used in combination and separately as well to produce better results. Elitism was used with Random Selection as well.

Crossover

A crossover operator is used to recombine two chromosome which were selected during the selection phase in the hope of getting a better one. The process of recombining the genes can performed in multiple ways. In this problem, three different types of crossover techniques were used based on some probability, e.g. Uniform Crossover, One-Point Crossover and Fixed-Point Uniform Crossover.

- **Uniform Crossover:** Child chromosome receives genes from both parents based on equal probability, i.e. it has fifty percent (50%) chance of receiving gene from parent one or parent two.

- **One-Point Crossover:** It is the simplest crossover technique according to [1]. A random crossover point is chosen. All the genes before the crossover point are taken from parent one and the remaining are taken from parent two.
- **Fixed-Uniform Crossover (fpu):** In fixed-point crossover, a fixed number **m** genes are randomly chosen from parent one. The remaining genes are inherited from the other parent. In this problem, **m** is equal to half the number of courses which have to be scheduled which is basically length of the chromosome.

Mutation

Mutation is the process of randomly distributing genetic information. It introduces diversity in the population which helps to avoid getting stuck at a local optima. It depends on the mutation rate. For the sake of experimentation, different mutation operators were used with different probabilities, e.g. Uniform Mutation, Interchanging Mutation and Reverse Mutation.

- **Uniform Mutation:** Randomly select a course and assign it a new random slot.
- **Interchanging Mutation:** Randomly select two courses and swap their slots. It is also known as Swap Mutation.
- **Reverse Mutation:** Randomly select a course and inverts the courses ahead of it.

Note: Since gene is basically an Array of integers (course numbers), randomly selecting a course means randomly selecting an index.

Evaluation (Fitness Function)

The fitness function evaluates a schedule based on the soft constraints and hard constraints. Hard constraints must be followed by an individual otherwise it would be invalid and infeasible. However, an individual can violate soft constraints and based on that penalties are given to it. A violation of a hard constraint has more penalty value than a soft constraint. To compute the fitness value of individuals, frequency of each violation is multiplied with the respective penalty.

	Hard Constraint	**Penalty**
1	Not even a single student can have more than two exams in one slot.	-1000
2	Not even a single student can have more than two exams in consecutive slots.	-100
3	Not even a single student can have more than three exams in one given day.	-200
4	All exams must be scheduled within the given number of days	-2000
5	Total students taking exam in one given slot must be less than the total room capacity.	-3000

	Soft Constraint	Penalty
1	Number students having two exams in one given slot must be minimized	Count of students is subtracted from fitness value
2	Number of students having exams on two consecutive slots must be minimized.	Count of students is subtracted from fitness value

In both scenarios, number of students violating the constraints are determined and subtracted from the fitness value.

Local Search

During Local Search, a mutation operator is applied on an individual in the population. In the problem statement, it was asked to use local search to refine the solution given by the Genetic Algorithm. For this purpose, Simple Hill Climbing was used. The fittest individual is taken from the population. If its fitness is 0 (best one), then we have reached our goal, i.e. no need to perform the search. For the local search to work successors need to be generated. In the solution, it was done by change the slot of courses which still had clashes after GA. Individual with maximum fitness value is chosen from the successors and compared with our fittest individual from the GA. If it is better, then it swapped with the fittest one and the search goes on. Otherwise, the local search halts as we have no operators left. It will return the refined individual which might be even better than the produced by GA.

In another approach, local search could be used during the mutation phase as mentioned in [2]. In this method, mutation operator, based on some probability, is applied on each individual in the population until an individual's fitness value improves. If the fitness has improved, the mutated individual is added to the new generation otherwise, the previous individual is mutated again until its fitness value improves. This search could be time consuming as fitness function is called many times during it. An improved version of the above approach is given in [3] which says that local search should be applied on the fittest individual in the offspring population only if its fitness value is greater than the fittest individual of the parent population.

Experiments Results

Several Experiments were conducted by varying parameters which include population size, crossover rate, mutation rate, elite individuals, tournament size and maximum generations allowed. The time it takes depends on factors such as number of exam days and exam slots. Some parameters were not changed such as Max Generations was set to 10000, Elite Individuals was set to 10 percent (10 %) of the population size and the Tournament Size was set to 5.

The performance of Genetic Algorithm with local search was tested with such parameters settings as shown in the table below.

Parameter	Value
Population Size	10, 20, 30
Maximum Generations Allowed	10000
Crossover Rate	0.50, 0.75, 0.95
Mutation Rate	0.50, 0.25, 0.05
Tournament Size	5
Elites	10 %

Population Size	Selection Method	Crossover Method / Rate	Mutation Method / Rate	Local Search Used/ Mutation Rate	Solution Found In Generation	Perfect Solution Found	Fitness Value After GA	Fitness Value After Local Search	Time Taken (minutes)	Constraints Violated (After GA)
10	Elitism + Random	Uniform, 0.9	Reversing, 0.1	0.3	199	No	-4921	-1300	2.47	-
10	Elitism + Random	Uniform, 0.7	Reversing, 0.3	0.3	99	No	-4999	-1650	2.745	-
20	Elitism + Random	Uniform, 0.8	Uniform, 0.2	1	649	No	-1256	-1240	49.02	-
10	Elitism + Random	Uniform, 0.95	Uniform, 0.05	0.50	204	Yes	-737	-724	7.33	None of the hard constraints were violated.
20	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	0.50	599	No	-1173	-1142	24.95	-
10	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	0.50	999	No	-361	-344	10.13	-
20	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	0.50	699	No	-607	-572	32.39	Hard Constraint no.2 was

										violated only once.
20	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	Not Used	1899	No	-779	-680	32.32	Hard Constraint no.2 and no.3 were violated only once.
20	Tournament + Elitism	Fpu, 0.95	Uniform, 0.50	0.50	649	No	-657	-652	28.76	Hard Constraint no. 2 was violated 2 times.
20	Elitism + Tournament	Fpu, 0.90	Uniform, 0.50	0.50	799	No	-349	-330	55.85	Hard Constraint no.2 was violated only once.
10	Elitism + Tournament	Fpu, 0.75	Uniform, 0.05	Not Used	1999	No	-9551	-	11.53	All Hard Constraints violated.
10	Elitism + Tournament	Fpu, 0.75	Uniform, 0.50	Not Used	1799	No	-1997	-	16.86	Hard Constraint no.2 and 3 were violated.
10	Elitism + Tournament	Fpu, 0.75	Uniform, 0.75	Not Used	2000	No	-1080	-	24.27	Hard Constraint no.2 was violated 7 times.
10	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	Not Used	2399	No	-1011	-	14.99	Hard Constraint no.2 was violated 6 times.

20	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	Not Used	1599	No	-392	-	21.02	Hard Constraint no.2 was violated once.
30	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	Not Used	1599	No	-627	-	30.57	Hard Constraint no.2 was violated once.
20	Elitism + Tournament	Fpu, 0.95	Uniform, 0.50	Used, 100%	1599	No	-716	-710	48.02	Hard Constraint no.2 was violated once.
10	Tournament + Elitism	Fpu, 0.95	Uniform, 0.50	Uniform, 0.50	7099	Yes	-232	-220	54.21	No Hard Constraint Violated.
5	Tournament + Elitism	Fpu, 0.95	Uniform, 0.50	Uniform, 0.50	3620	Yes	-588	-400	10.965	No Hard Constraint Violated.

Note:

- **“No”** means that the perfect solution with fitness value of 0 was not found. It could have several causes, e.g. any of the stopping criteria for this problem which have been mentioned above.
- The column “**Local Search Used/Mutation Rate”** indicates the values when local search was used during the mutation phase. As I’ve already mentioned, I have tested local search in multiple ways, i.e. during mutation and after genetic algorithm ends.
- The column “**Fitness Value after Local Search”** indicates the approach where Simple Hill Climbing was used on the best solution returned by the Genetic Algorithm, i.e. after genetic algorithm ended. This is different from local search done in the above statement. Due to this approach, fitness value improved afterwards, as depicted in the above table.

Conclusion

A smaller population size leads to a quicker convergence speed but the algorithm might get trapped in a local optima. The reverse thing applies to a large population size. Higher crossover (0.95) rates helps the solution converge on one of the good solutions found so far. It was found that higher mutation rate (0.50) produced better solutions while keeping all other parameters the same. With higher mutation rate, the search space is exploited more. It also prevents getting stuck at a local optimum. The algorithm produced schedules with no hard constraint violation when population size of 10, 20 and 30 were used. There is a chance of getting stuck at a local optima but higher mutation rate and local search avoided that. Higher population sizes were also tested which result in longer computation time due to nature of the problem.

Crossover and Mutation operators also have an impact on the effectiveness of the Genetic Algorithm. It was found that Fixed-Point Uniform Crossover (fpu) operator performed best. Amongst the mutation operators used, Uniform Mutation operator was found out to be the most effective whereas Reversing Mutation operator performed the worst. Different Selection mechanisms were tried to solve this problem. Tournament Selection with Elitism proved to be the best.

Local Search was used in three days different ways as already mentioned. All of them significantly improved the fitness value of an individual and number of clashes were reduced. However, it takes some time as fitness function is called almost every time in these methods.

As shown in the above Experiment Results table, in most of the solutions only Hard Constraint no.2 was violated which means that only a few students would have to give more than 2 consecutive exams. Moreover, perfect schedules were created as well. It is to be noted that only violations of hard constraints were mentioned as they determine the feasibility and validity of the solution. In almost all of the schedules generated, soft constraints were violated.

After doing experiments using different population sizes, crossover rates, mutation rates and different combination of selection, crossover and mutation operators, the best set of parameters were found out to be:

Selection Operator	Elitism + Tournament
Crossover Operator	Fixed-Point Uniform Crossover
Mutation Operator	Uniform Mutation
Crossover Rate	0.95
Mutation Rate	0.50
Elite Allowed	10 %
Tournament Size	5
Population Size	20

References

[1] Davis L., Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold, 1991

[2] M. Yusoff and N. Roslan. "Evaluation of Genetic Algorithm and Hybrid Genetic Algorithm-Hill Climbing with Elitist," presented at the 10th Int. Conf. Advances in Swarm Intelligence, Chiang Mai, Thailand, 2019

[3] Wen Wan and Jeffrey B. Birch. "An Improved Hybrid Genetic Algorithm with a New Local Search Procedure." Vol. 233, pp. 2-4, Oct. 2013.