Exam Timetable Problem using Local search and Genetic Algorithm

Hunzlah Malik
Aashar Naseem
Hussain Mohy Ud Din

Local Search

Local Search is an approach that follows a heuristic to solve an optimization problem to achieve a solution with a maximized criteria in a specific problem among a number of other candidate/potential solutions.
In our Local Search approach, we create a chromosome, create N neighbors by randomizing the genes, check for the fitness, if fitter than previous chromosome, consider it best and then again call neighborhood operator on it. The level of iterations are decided by us.

Genetic Algorithm

Genetic Algorithms (GAs) are a specialization of evolution programs, based on the principals of natural selection and random mutation from Darwin biological evolution. They were formalized in 1975 by John Holland and have been growing in popularity since, particularly for solving problems with a large irregular search space of possible solutions described by Colorni et al., Rawat and Rajamani.
In our GA approach, a population of feasible timetables is maintained. To form the basis of next iteration or generation, the fittest timetables are selected. To get the best results, basic operators such as selection, mutation and crossover are applied.

Chromosome

The chromosome is made up of many genes. In our program we are using courses, students, room and slot as our genes. So, when these many genes combine, it will make a chromosome. In other words, you can say this is our representation of the genes.

$$Gene \;=\; (course,\ room,\ slot,\ students)$$

$$Chromosome \;=\; [Gene1, Gene2, Gene3, \ldots\ldots, GeneN]$$

Constraints

In exam timetabling problems, given exams have to be assigned to a number of periods such that there is no violation of hard constraints. Hard constraints are those to which timetable has to adhere in order to be satisfied.
These are following:

- One course exam should be in a slot
- One room should have only one paper at a time
- One student should have one exam at a time
- Every Course should have Exam

- Every Student should have Exam
- Rooms should have enough space for the present Course Students

Violating the above constraints will cause the timetable to be unfeasible.

Fitness Evaluation

The fitness function of each chromosome is evaluated by checking the conflicts in the Constraints. This paper is assigning a penalty value which contributes to the fitness function for each constraint violated.

Our fitness function is made up in the form

$$fitness(chromosome) = 1/(1 + x)$$

where x is a sum of weighted penalty

$$x = \sum wi.ni$$

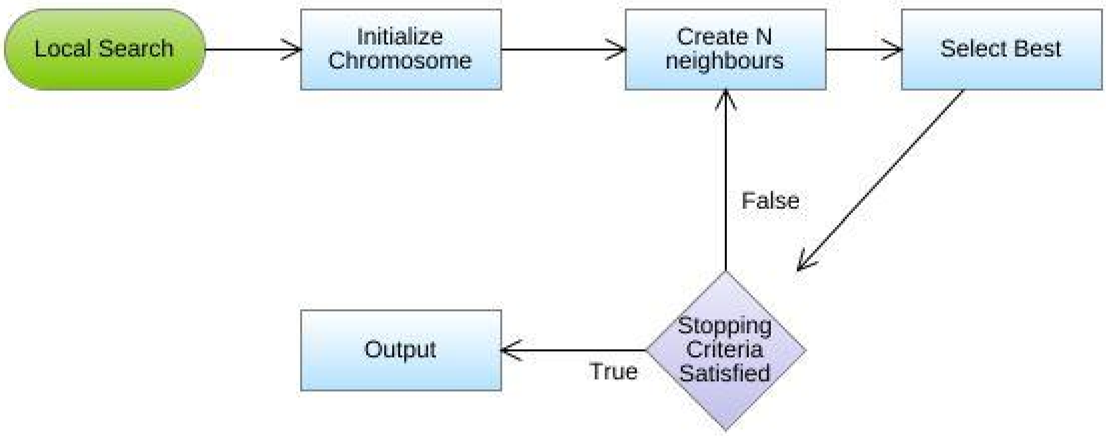
where wi is the weight of constraint and ni is the conflicts of that constraint. In this paper implementation we didn't have enough time to test and decide the weights of the given constraints.

Local Search Implementation

In our Local Search approach, we create a Best chromosome, create N neighbors by randomizing the genes, check for the fitness, if fitter than previous chromosome, consider it best and then again call neighborhood operator on it. The neighbors and level of iterations is decided by us.

The implementation of Local Search is such that we get the best chromosome out of the population of chromosomes on the criteria of fitness. Then in Local Search, we move towards making N Neighbors for that Chromosome using neighbor operator. The neighbor operator creates N neighbors by randomly mixing gene data to create new and different genes for neighbors of the chromosome. Now if the newly created neighbor has a better fitness value, then it is the new best chromosome and then again the neighbor operator is called on this best chromosome. We decide how many times to repeat these iterations and also the N Neighbors are decided/set by us.

The solution is selected if the desired fitness is achieved or until the iterations are completed.



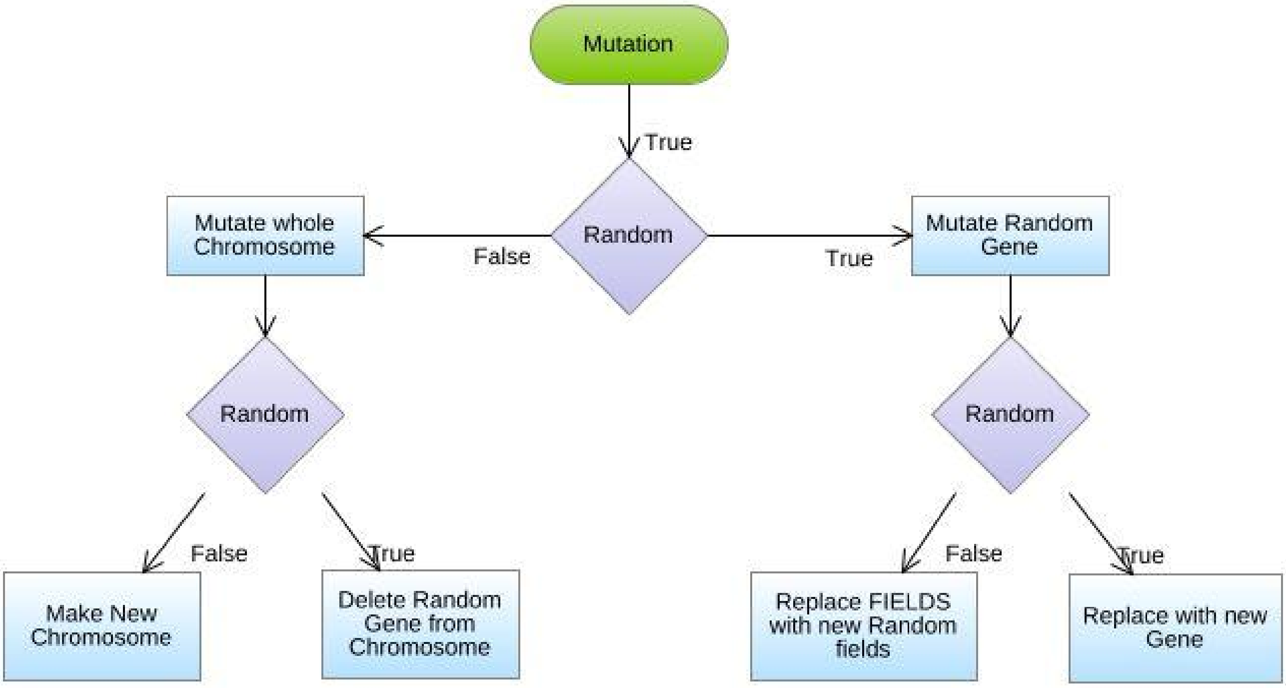
Genetic Algorithm Implementation

Crossover

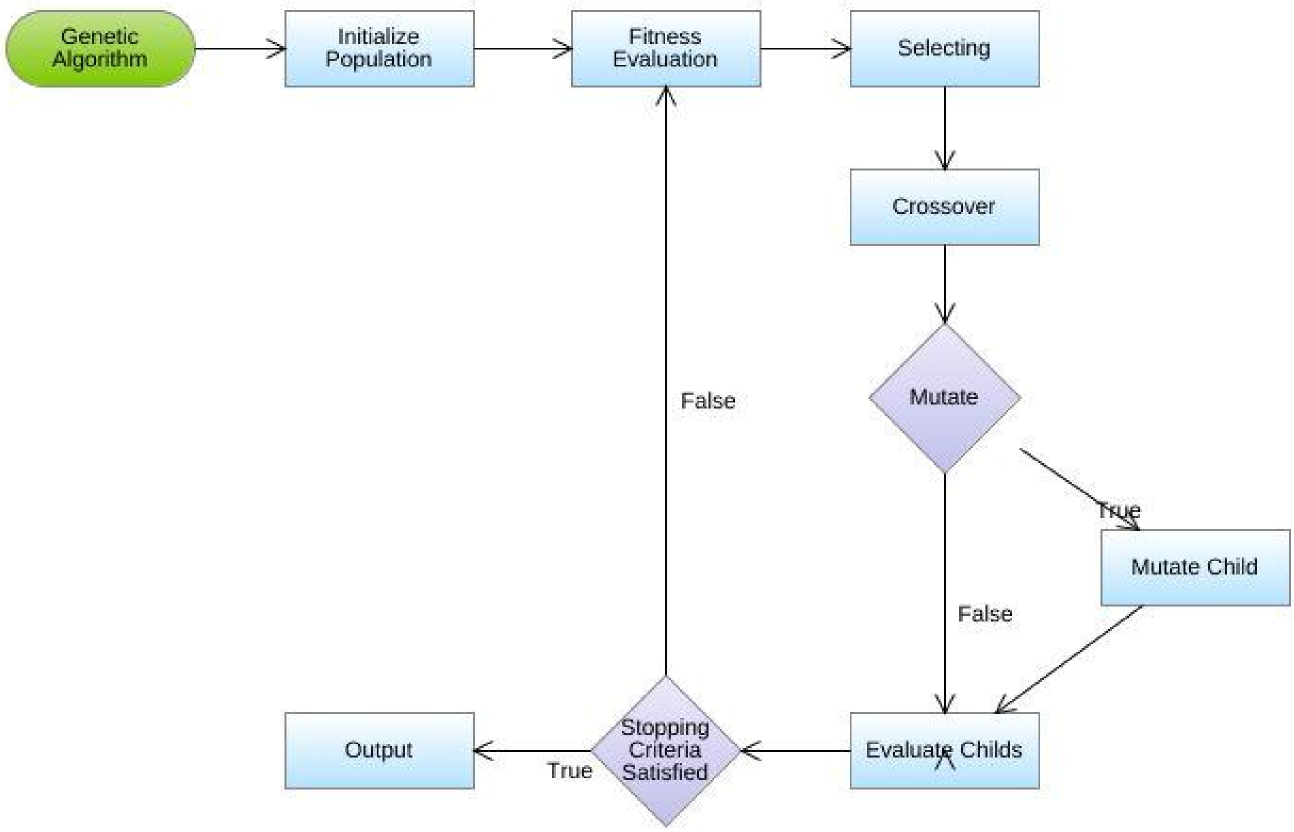
In GA approach, crossover is done such that a pair of parent chromosomes produce a new chromosome(child), in the child chromosome, a random point is selected and the genes prior to that point are selected from Parent1 and the genes after that point are selected from Parent2, just like single point crossover.

Mutation

Mutation is based on the random True/False. And based on that True False the chromosome is mutated. To improve the mutation we are passing the 'FIELDS' that aren't performing well. Where FIELDS are Course, Room, Slot or Students.
The flow chart of our mutation process is as follow.



Implementation



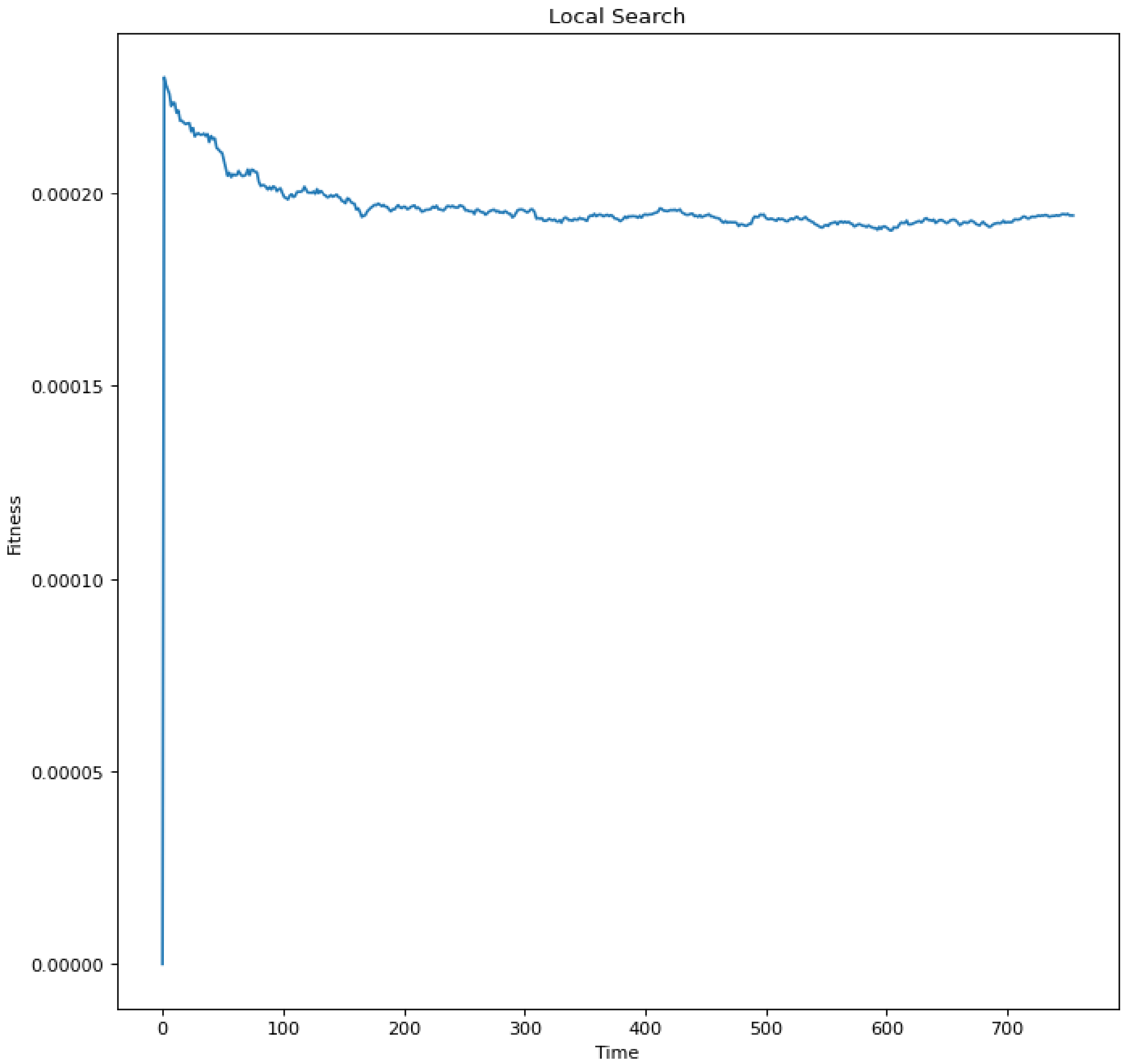
Computational Results

Local Search

Total Iterations: 500
Neighbors: 10
Best Output Conflicts:

```
{'conflicts': 40, 'name': 'one_exam_in_one_slot', 'weight': 10},
{'conflicts': 6, 'name': 'one_room_have_one_exam', 'weight': 10},
{'conflicts': 3178, 'name': 'student_one_exam_at_a_time', 'weight': 1}
{'conflicts': 62, 'name': 'one_exam_per_course', 'weight': 10},
{'conflicts': 87, 'name': 'room_cap_enough_for_students', 'weight': 1}
```

Output with respect to time



Genetic Algorithm

Max Generations= 500
Mutation Probability= 0.5521096046190731
Initial Population= Population[size=10, best=Chrom[Genes=87, Fitness=0]]
Best Output conflicts:

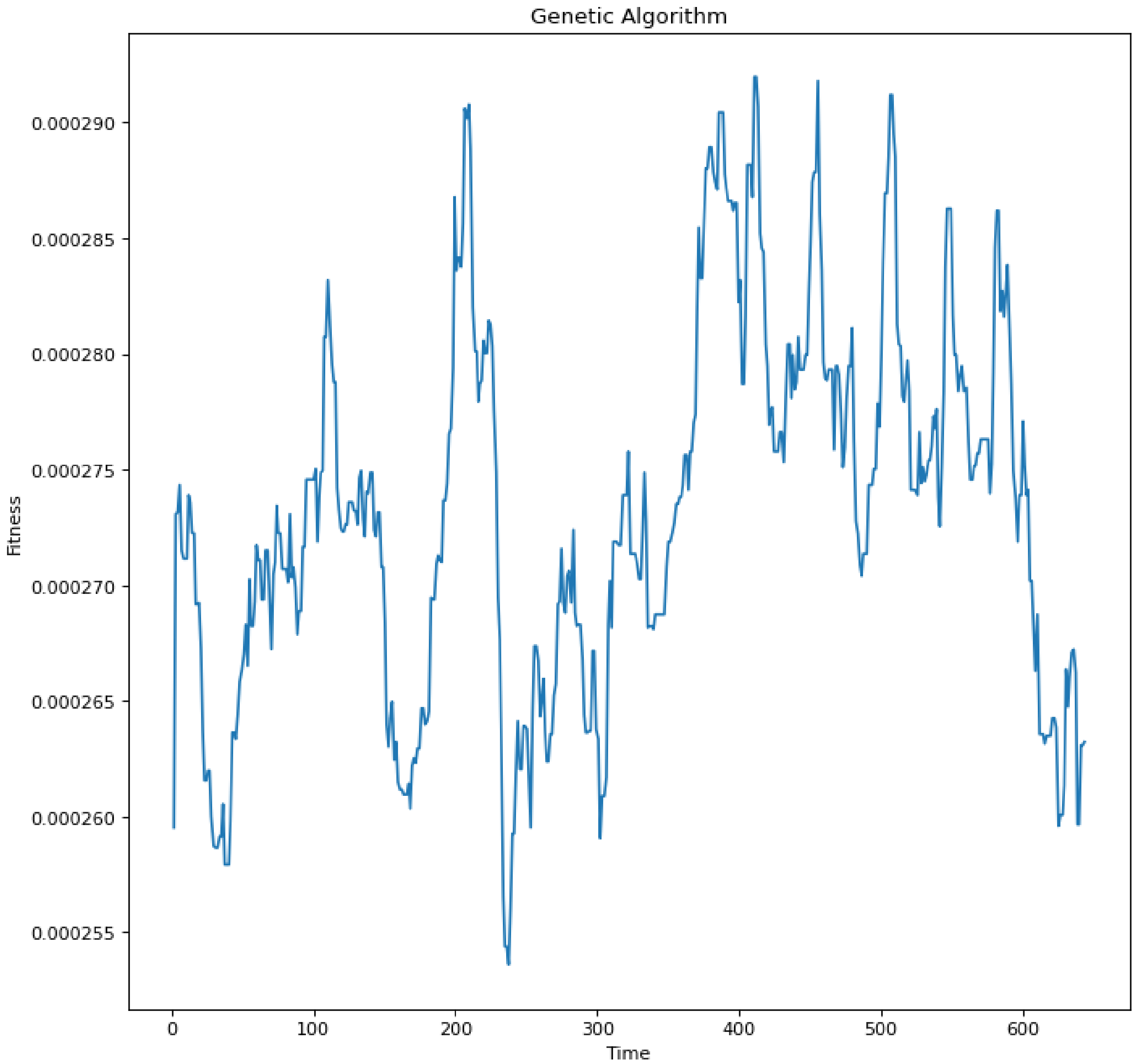
```
{'conflicts': 28, 'name': 'one_exam_in_one_slot', 'weight': 10},
{'conflicts': 2, 'name': 'one_room_have_one_exam', 'weight': 10},
{'conflicts': 2651, 'name': 'student_one_exam_at_a_time', 'weight': 1}
{'conflicts': 46, 'name': 'one_exam_per_course', 'weight': 10},
{'conflicts': 13, 'name': 'room_cap_enough_for_students', 'weight': 1}
```

Best Output after 500 generations:

RID/SID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1			67																	
2																22				
3								9												14
4										18	34	38								
5			43																	
6																				
7				31				38									17			
8								5								2				
9														36						
10																		55		41
11														14, 28						
12												61								
13	24						3	47								48				
14									11											
15											22								68	
16	45					37					58					46	52	39		
17																				
18																				
19		50																		
20				66								15								29
21		15				2										22				
22																				
23					19		3													19
24										22						43				
25	61			50			17							60						

26																		14		
27													25							
28						17			58, 38										57	
29			65																	
30		15	4	38																
31		56								31										
32																				
33						61														
34						38							21							60
35																				
36																				
37														18						
38			26		19															
39																		20		
40																				
41	43														40					
42		63															62			
43								70		38								38		
44														39		41	1			
45								28												
46		7						15												

Output with respect to time:



Hybrid Approach

No of Iteration: 500
Neighbors: 10
Best Output conflicts:

```
{'conflicts': 28, 'name': 'one_exam_in_one_slot', 'weight': 10},
{'conflicts': 2, 'name': 'one_room_have_one_exam', 'weight': 10},
{'conflicts': 2651, 'name': 'student_one_exam_at_a_time', 'weight': 1}
{'conflicts': 46, 'name': 'one_exam_per_course', 'weight': 10},
{'conflicts': 13, 'name': 'room_cap_enough_for_students', 'weight': 1}
```

Output with respect to time