E-learning Timetable Generator Using Genetic Algorithms

Eng. AHMED HAMDI ABU ABSA
University of Palestine
E-mail: a.absa@up.edu.ps
P.O.Box:322 Gaza-Palestine
Tel. 082880001 Fax:082880006

Dr. SANA'A WAFA Al-SAYEGH
University of Palestine
E-mail: s.sayegh@up.edu.ps
P.O.Box:322 Gaza-Palestine
Tel. 082880001 Fax:082880006

ABSTRACT

In this paper we explain the details of the implementation of a computer program which employs Genetic Algorithms (GAs) in the quest for an optimal lecture timetable generator. GA theory is covered with emphasis on less fully encoded systems employing non-genetic operators.

The field of Automated Timetabling is also explored. A timetable is explained as, essentially, a schedule with constraints placed upon it. The program, written in java, that has a good object oriented to do it, and it has the special libraries to deal with genetic algorithm which will be used for the implementation. In a simplified university timetable problem it consistently evolves constraint violation free timetables. The effects of altered mutation rate and population size are tested. It is seen that the GA could be improved by the further incorporation of repair strategies, and is readily scalable to the complete timetabling problem.

KEY WORDS: AI, Genetic Algorithm, Timetable Generator, E-learning Application.

1. INTRODUCTION

Paradigms of design theory are defined for the GA by natural selection as a quantitative measurement of life and ability to contribute to the reproduction process. The second property is the random genetic difference – mutation of the genetic information (this is important for small populations. The third typical property is the process of reproduction, which is the opportunity for genetic information exchange – crossing, which is how the algorithm to be effective. [1]

The basic properties of design algorithms are defined as follows:

Algorithm design can be an effective tool for optimization of processes. (They are used in cases, in which the aim is a searching of global extreme with a lot of several local extremes of the same type exists).

Design algorithms give the tools for monitoring of properties of k -time generation. It can be monitored some changes, which are made by the changes, and by this way of comparison of results of experiments without the changes, and with the change of information in “chromosome”.

Design algorithms give the opportunity to “social relations” monitoring between the generations on the basis of information changes of “chromosome”. [1]

2. PREVIOUS WORK

The initiator of research in GAs is John Holland. Holland published the book "Adaptation in Natural and Artificial Systems", and then on many dissertations and papers began to be published by different researchers. In 1985 the general approach began to receive wide attention. Two formal conferences on GAs: [2,4]

"Proceedings of the International Conference on Genetic Algorithms and their Applications" It is held in odd years from 1985 in United States and "Parallel Problem Solving from Nature" It is held in even years from 1990 in Europe. In addition, the theory-oriented workshop "Foundations of Genetic Algorithms and Classifier Systems" It is held every two years from 1990; the term classifier system denotes a simple kind of GA based machine learning system. A well-organized GA-List digest news list and another EP-List digest are available via e-mail, while major journals publishing GA-related papers are Evolutionary Computation, Machine Learning, Neural Networks, Artificial Life, Adaptive Behavior, and other Artificial Intelligence based journals. [3]

3. THE STRUCTURE OF E-LEARNING TIMETABLE GENERATOR

The structure of e-learning timetable generator consist of four main modules, a top level of this system structure is shown in figure 1. The major system modules are: Input Data module, connection and relation between the input data module, timeinterval and time slots module and applying GA module then extract the reports.

3.1. INPUT DATA MODULE

The input data module is described by the CVS; CVS is a type of data from the database. It can be opened by Microsoft excel software. The data contains:

Person: data describe the name of lecturers in the university.

Subject: data describe the name of courses in the class in the university.

Room: data describe the name of classes in the university and the capacity of each it.

Timeinterval: is a time slot with a starting time and duration. It has a subject and room where it takes place.

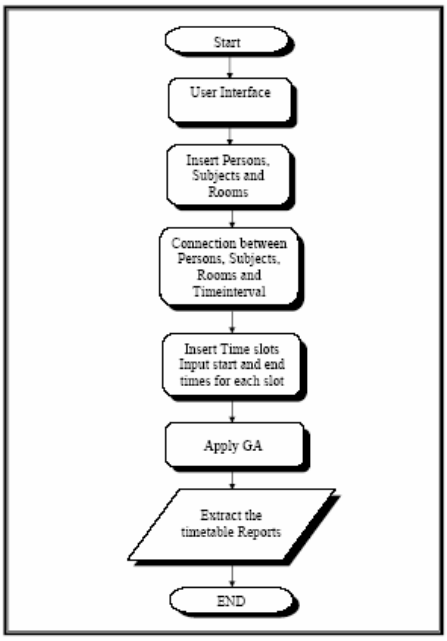

Figure (1) The Structure of the e-learning timetable generator

3.2 CONNECTION BETWEEN INPUT DATA MODULE

The relation between the input is the main problem, because if the relation is true, it will extract a good timetable and you can control the entity of it. The relations between the input objects are shown in figure2.

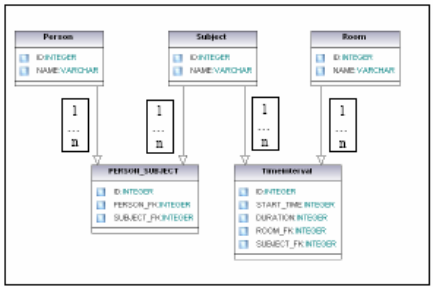

Figure (2): The relation between the input objects in the timetable problem

We show Person Object that has two attributes ID number and the name of the lecturers in the university and Subject Object that has two attributes ID number and the name of the courses will be studied in this class and Room Object has two attributes ID number and the name of the classes in the university and Timinterval Object have five attributes ID number, Start time, Duration,Room_fk and Subject_fk. The Timeinterval Object is a time slot with a starting time and duration and. It has a subject and a room where it takes place in the time slot.

The PERSON_SUBJECT Object is done to make a combination between the Person Object and the Subject Object, because the Person Object has more than one timeinterval for each person so to solve this problem. The object was done in the generator, table (1) describe this note.

Table (1): The relation between Person object and Timeinterval object

Person1	Person2	...	PersonP	Subjects	Room1	Room2	...	RoomR
1	0		1	TimeInterval1 of Subject1	1	0		0
1	0		1	TimeInterval2 of Subject1	0	1		0
...			...	...				
0	0		0	TimeIntervalJ of Subject2	0	1		0
				....				
0	0		1	TimeIntervalT of SubjectS (SubjectS could have more TimeInterval's)	0	0		1

3.3 INSERT TIME SLOTS MODULE

The time slot in the program is dependent on the university satisfaction, for example, in The University of Palestine there are 5 study days in the week. The student studies these days which has the start time from 8:00 am and the duration time is 90 minutes. The number of hours in the day is 12 hours, so if we want to program it, the day duration is 144 (number of hours*60/TimeAtom), and the explanation of the timeslot is shown in figure 3.

TimeAtom is a unit of the timeslot in the timeinterval objects that to determine the number of hours in the day.

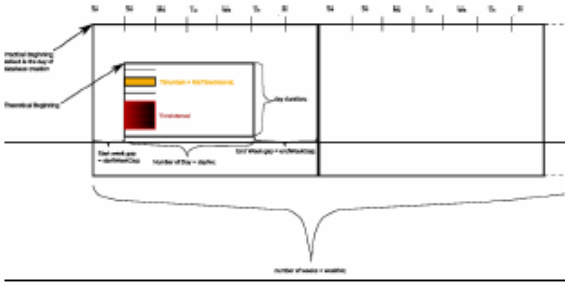

Figure (3): The explanation of the timeslot in the timeinterval

3.4 APPLED GA MODULE

The GA in the timetable problems process and optimize the gaps where the conflicts between the time slots are depending on these constraints:

- Teacher cannot give more the one lecture at the same time.
- Teaching group cannot learn more subjects at the same time.
- Room can be devoted to only one subject at the same time.
- Room must be sufficient for the maximum number of students.

The structure of the genetic algorithm is consist of five sub modules, a top level of this system structure is shown in figure 4.

3.4.1 GENERATION SUB-MODULE

Every row of input data of the teacher, subject or room, it is generated object of corresponding class. The description of data of subjects is processed by different ways.

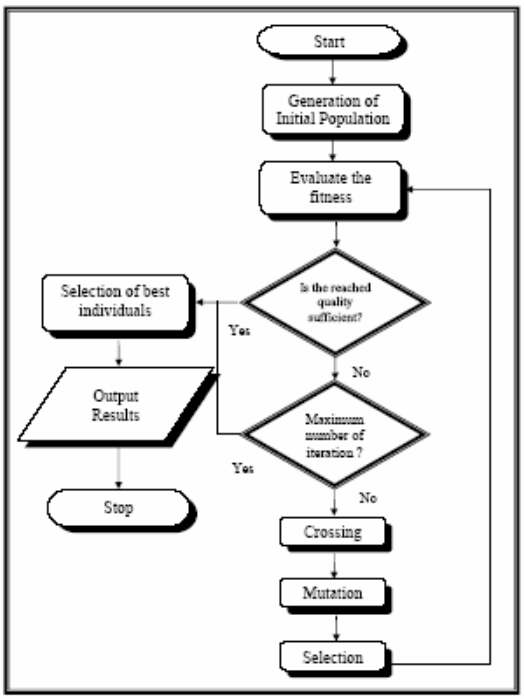

Figure (4) The Structure of the genetic algorithm

Every lecture is represented by the classes of the Subject, compared to the number of the time periods occupies.

If it is, for example, the lecture with the time period 3, it is generated 3 objects Subject with the same data, which differs by the internal counter only. In other words: one object Subject represents a part of longer teaching unit of one time unit (90 minutes).

Every generated chromosome contains two arrays of equivalent size as data.db4o.Counter. Into the first array data.db4o.RoomDb4oImp there are saved rooms, and into the second data.db4o.TimeIntervalDb4oImpl time periods are saved. By this way, triplets are generated (lecture, room, and time period).

Both of the values and structure of these objects are not changed through the algorithm running, and there are common for all chromosomes. Every chromosome contains another two rows (rooms and time periods) which concatenates, by the array index, to every Subject in data.db4o.Counter the room and time period. The values in data.db4o.TimeIntervalDb4oImpl and data.db4o.RoomDb4oImp are different in every chromosome, and they generate the variants of timetable by this way.

Note to the time representation: the time period of the whole week is the natural number in the interval from 1 to ITimeInterval. If it is, for example, the maximum time 75, the Saturday first time period is 1 and the last is 15, the Sunday contain 16, … 30, and so on.

3.4.2 FITNESS SUB-MODULE

The fitness function Chromosome.fitness is devoted to solve the suitability of timetable represented by the chromosome. The calculation is based on the evaluation of "penalty" if it is disturbed any limitation. If the fitness is equal to zero, it was found good timetable.

Due to the sequence of elements with the length n, data.db4o.Counter belongs to one lecture, n elements must be concatenated with the same room (the both time periods are done in one room) and the time slots must be in the time sequence.

The method Chromosome.fitness is penalized where:

- Various teaching time periods belonging to one lecture are not given at the same room.
- Various teaching time periods belonging to one lecture are not given on the same day (the time 15 is the last on Saturday, the time 16 is the first on Sunday).
- Teaching time periods are not in the sequence (for triple periods are optimal 5+6+7 or 7+6+5, but it is wrong 4+5+7).
- Room has not sufficient large (the room for 15 persons is not sufficient for the group of 20 persons).
- One room is planned for two lectures at the same time.
- One teacher is planned for two lectures at the same time.

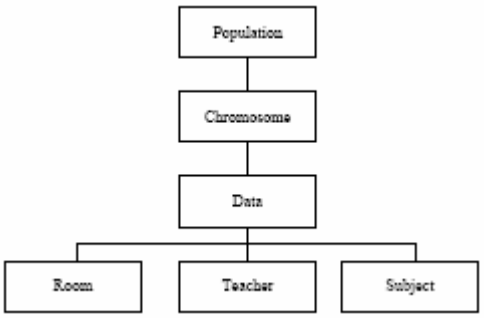

Figure (5) The Structure of the input data in the generator

3.4.3 CROSSING SUB-MODULE

The new descendant is started (by the construct Chromosome (Chromosome parent1, Chromosome parent2)) by crossing of two parents for every case. The descendant receives random part of field data.db4o.RoomDb4oImp,and data.db4o.TimeIntervalDb4oImpl for the first parent, the rest it receives from the second parent. The cross-point is various for the two fields.

3.4.4 MUTATION SUB-MODULE

The mutation of the chromosome is realized by the call of the method setMutationRate. It can be realized one of four possibilities. The first two cases are random changes of the chromosome:

Exchange of randomly chosen elements of data.db4o.RoomDb4oImp, array, and two elements of data.db4o.TimeIntervalDb4oImpl – the two randomly chosen lectures are exchanged in time periods, and the another two are exchanged in their rooms.

The randomly chosen lecture is substituted by the randomly chosen time period, and to another lecture is randomly chosen room.

These two kinds of mutation were tested stand alone. The algorithm did not rich asked results, and due to this reason, there were completed other two mutations for their "correction". These new mutations stream to solve the higher complexity of searching, which has its origin in dividing one lecture into more Subjects. Both mutations ask recomputing fitness value by call of the Chromosome.fitness with the special parameter. After this call, the position of lectures will be (in the first time period) in the chromosome with the error. The process will choose one position randomly and it streams to correct these errors:

If the error exists, in which teaching periods are not concatenated or teaching periods are not in one day, the teaching unit is equipped by the "random time slots" (random is only the first, the others are defined in concatenation with it) by this way, the errors to be eliminated.

If the error exists, in which teaching periods have not the same room, the random room equips the teaching unit with the suited type, and size.

Both of these corrections have a local influence only. The corrections can cause collisions in the chromosome. The convergence of the algorithm was dramatically higher, after completing these "correction mutations". Without these corrections it was needed to done more then one half of million generations for the good timetable, after corrections there were necessary only some thousands for the timetable finishing with the same data.

3.4.5 SELECTION AND REPRODUCTION SUB-MODULE

A lot of strategies were used for parents' selection for crossing and for selection of individuals, which are active in the next generation. The best selection is as follows:

Tournament of parents

The new generation is created by crossing of individuals selected from the original generation by the tournament method. In other words, the two parents are selected for the new individual by the tournament method.

Tournament of parents + children

The temporary set of individuals is created. Parents of each individual are randomly selected from original generation. To the next generation, individuals are selected by tournament form set of both parents and their children.

3.4.5 INFLEUENCE PARAMETERS FOR GA RUN

The main aim of measurement was to know what parameter influences the tendency to searching the time of processing. The speed of solving was evaluated by the number of generations needed to create a "good" timetable. Cases were considered when the algorithm did not finish the timetable in the declared number of generations.

In the tests, mutation probability was changed from 0.0 through 0.5 to 1.0. The size of tournament was 3, the size of population 80, and selection tournament methods were used, such as parents + children, and tournament parents as shown in table 2.

Table (2): The influence of parameters

Parameter	Possible Values	Occurred Values	Mean	RmsError	Comment
#children	80, 100, 120, 140, 160, 180	100, 120, 160, 180	154	26	140 comes not until 64 80 comes not until 188
mutationRate	0.1 0.8 1.5	0.8 1.5	1.0	0.3	0.1 comes not until 41
dayRate	10, 50, 90	50, 90	84	14	10 comes not until 150

startTime, (0.1 and 1) #parents (80 and 120), #TI's (80 and 120) have no tendency for roomRate (0.1, 1, and 10) it seems to be that smaller values are better. The implementation of mutation is very variable compared to the mutation of classic GA. In any case, it has the function of normal mutation (with random change of chromosome part).

The general property of mutation is its ability to correct any error as a part of chromosome by the random value (of suited values). The generalization of mutation is done by the tests, the strongest tool in timetable searching. The probability of mutation is very low value (from 0.01 to 0.2) in classical GA.

The results of testing bring the conclusion that it suits to use the bigger value of probability of mutation. The best results were reached for the value of probability of mutation equal to 1. This is the case when all individuals are mutated in all generations. The highest probability of mutation makes the number of needed generations dramatically low. It makes the number of generations needed for the tendency of fitness = 0, in the same case, it raises the necessary time for computation of individuals in one generation.

The combination of antagonistic influences gives the best results (lower number of generations). From the tests, it is clear that the higher probability of mutation gives the higher speed of tendency of algorithm if we consider both of the number of generation and the total time of computation. The above parameters were chosen in the generator in table 3.

Table (3): The GA Parameters in the timetable generator

GA Parameter	Assigned Value
Size of Population	80
Number of Generations	100
Crossover Probability	0.7
Mutation Probability	0.01
Number of Children	150
Number of Parents	80

Now the algorithm for GA in the e-learning timetable generator is as the follows:

Algorithm: Apply GA in the generator

Input : import de.data.IRoom: The list of rooms in the database.
import de.data.ISubject: The list of subjects in the database.
import de.data.IPerson: The list of persons in the database.
import de.data.TimeFormat: The themes of the timetable.
import de.data.ITimeInterval: The relation between the input data.

Output: The best individual to extract the timetable reports.

Step1: Define some variables in the algorithm:

```
private static final EvalComparator evalComparator = new
    EvalComparator();
private Individuum originalIndividuum;
private Individuum bestIndividuum;
private int generation = 1;
```

Step2: Set the initial parameters to do GA: Size of Population, Number of Generations, Crossover Probability, Mutation Probability, Number of Children and Number of Parents.

Step3: Generate the individuals.

Step4: Store the best individuals in a variable to restore from it.

Step5: Produce new generation.

Step6: The time is enough go step 7 else go step 4.

Step7: Recombination of individuals to produce children.

Step8: Evaluate the individual**.**

Step9: Crossing the individual.

Step10: Mutate the individual.

Step11: If the individual is the best go step 12, else go step 8.

Step12: Select the best individual for the next generation.

Step13: Update the initial timeintervals to the best timeintervals.

Step14: Perfomes some changes on the timeintervals.

Step15: Mutate the genes of all individuals.

Step16: Calculates and determine the gaps and the collision of each individual.

Step17: Choose the best evaluation of the individuals.

Step18: Extract the reports

Step19: End.

For more details pseudo code of Function Generate the individuals is shown below:

Pseudo Code of Function Generate the individuals

Input: Number of parents, Number of children, First evaluation.

Output: Allindividuals.

```
int noOfParents =
      privateProperties.getNoOfParents();
 int noOfAll = noOfParents +
      privateProperties.getNoOfChildren();
 List<Individuum> allIndivid = new
ArrayList<Individuum>(noOfAll);
for(int i = 0; i < noOfAll; i++) {
         allIndivid.add(new
Individuum(originalIndividuum));       }
 generation = 1;
 float firstEval = allIndivid.get(0).getEval() * 0.95f;
  float secEval = firstEval;
 while(generation <
privateProperties.getMaxGeneration() &&
!finishIfRequested() && allIndivid.get(0).getEval()
> targetValue) {
maxDelayInMin =
privateProperties.getDelayInMinutes() * 1000 * 60;
         currentMilliDelta =
System.currentTimeMillis() - firstMillisForStatus;
if(currentMilliDelta > maxDelayInMin) {break;  }
generation++;
//allow changing of parameters
int oldNoOfParents = noOfParents;
  noOfParents =
privateProperties.getNoOfParents();
         noOfAll = noOfParents +
privateProperties.getNoOfChildren();
  allIndivid = ensureSize(allIndivid, noOfAll);
 // recombination of individuals to 'produce' some
children
 for(int i = oldNoOfParents; i < noOfAll; i++) {
       int i1 = random.nextInt(oldNoOfParents / 4);
       int i2 = random.nextInt(oldNoOfParents);
       allIndivid.set(i, new Individuum(
allIndivid.get(i1), allIndivid.get(i2),
random.nextInt(allIndivid.get(i1).size())));    }
assert allIndivid.size() == noOfAll : "" +
allIndivid.size();
```

3.4.6. STATISTICS IN THE GENERATOR

Before knowing the results of the statistics, we must know the main parameters to make them, there are:

- #children should be more than 20 (100 was better)

- roomRate should be less than 50 (50 were the worst results)
- dayRate should be more than 5 (50 were the best results)

The parameters: #TI's, startTimeRate, mutationRate and #parents have no tendency. The Statistics was made [mean, RmsError] at the following table 4.

Table (4): Statistic parameters in the generator

Configuration No.	#parents	#children	#changed TI's	mutationRate	roomChangeRate	startTimeChangeRate	dayChangeRate
3	80	150	80	1	0.5	0.6	80
2	8	50	10	0.5	10	10	100
1	8	50	10	0.1	1	10	5

The mean values of all good values (evaluation < 340) of every column were found in table 5.

Table (5): The best evaluation in the generator

Mean Evaluation	RmsError of Evaluation	Mean Generation	#TIs	#children	#parents	mutationRate	startTime	roomRate	dayRate
290 (+-38)	33 (+-33)	1336 (+-611)	39 (+-45)	84 (+-35)	37 (+-40)	17 (+-22)	16 (+-21)	10 (+-18)	31 (+-22)

where the value in brackets was the rms error.

In this section, we applied timetable generator in the University of Palestine, semester 12007. The fetcher study in the university has two semesters; every semester lasts for 16 weeks. The week starts on Saturday and ends on Wednesday from 8:00 am until 7:00 pm. The time slot is various from 60 to 240 minutes.

There are four types of the timetables in the university:

- Timetable for the lectures in the university.
- Timetable for the discussion hours was used to the students that lives out the country.
- Timetable for virtual rooms that depend on the conference program between the lecturer and the students.
- Timetable for the lab.

First Example

The input data needed to insert in the generator in the table 6.6. has a list of lecturer, classes and rooms.

Table (6): Example parameters in the timetable generator

Input Data	Assign Values
List of lecturer	56
List of Subjects(include the number of sections)	147
List of rooms	19

Now we want to make relation between the inputs data. The first relation is between lecturer and subjects by using the icon add subject and the results will be saved in the database as shown in figure 6, we show how many subjects are chosen for every lecturer in this class.

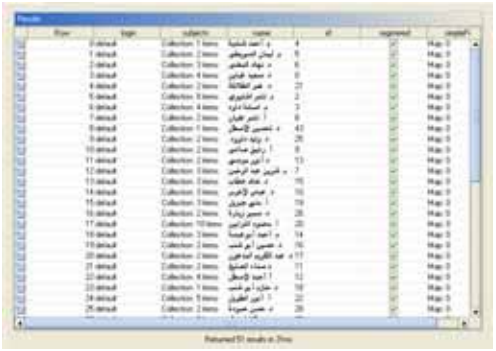

Figure (6): Relation between lecturers and subjects

After that, we will make the relationship between subjects and rooms and insert it in the timeslots depending on the university fetchers, so we will make relation between rooms and timeintervals and the relation between subjects and timeintervals, where the data will be saved in the database as shown in figure 7 and figure 8.

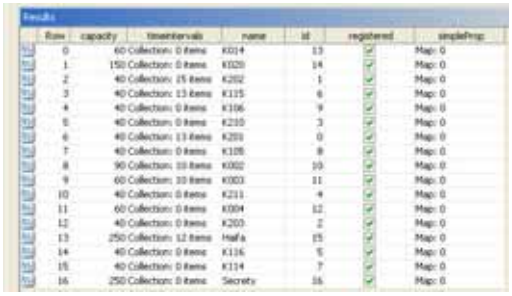

Figure (7): Relation between rooms and timeintervals

After that we want to view the input data and the relation between them, the icon execute make this statement for the list of persons, subjects and rooms as shown in figure 9, and 10.

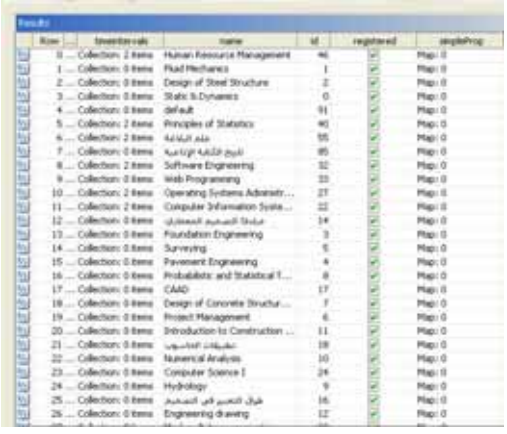

Figure (8): Relation between subjects and timeintervals

Figure (9): View the subjects for the lecturer

Figure (10): View the subjects for the room

However, there are many errors in the relation between the inputs data. Therefore, we want to correct and optimize the errors found in the timetable. The genetic algorithm was build in the E-learning timetable icon that makes the steps was explained in bravoes sections. In practice, we found collision between two subjects in the same time as shown in figure 11, this collision was an observation in the red colure and we will be used the genetic algorithm to solve this problem.

Figure (11): Show the collision in the generator

Figure (12): Use genetic algorithm in the generator

Figure (13): The solution after using GA

The genetic algorithm solves the problem by moving the subject2 to another timeslot that satisfy all the constraints in the University of Palestine. Thus, this is the application for the framework in this dissertation.

Second Example

Timetable for the discussion hours which uses by the students that lives out the Palestine country, from the list of lectures in the first example, some students in the university live in other countries, so the discussion hours is different from the students lives in the Palestine, that depend on the conference program was done in the university.

So this type of the timetable does not depend on the rooms because the teacher want the computer device and other techniques to contact with the students, for that the important constraint in this type that the teacher can not take more than one lecture in the same time, and the student can't take two lectures in the same time.

The virtual rooms was used in the conference program will be using in the discussion hours.

Third Example

Timetable for virtual rooms that depend on the conference program between the lecturer and the students and specific rooms that have special techniques to deal with conference program for example pc device, internet, amplifier, …etc. this type is combination between the first and second example, where the lecture is want a room that students still in it and a virtual room to contact the lecturer by using the conference programming.

This type of the timetable is applied in the Cairo program in the university of gradate studies.

The first relation is between lecturer and subjects by using the icon add subject and the results will be saved in the database. After that, we will make the relationship between subjects and rooms and insert it in the timeslots depending on the university fetchers, so we will make relation between rooms and timeintervals and the relation between subjects and timeintervals.

After that we want to view the input data and the relation between them, the icon execute make this statement for the list of persons, subjects and rooms

Now we will in every room join with a virtual room

Fourth Example

In this type of the timetable is similar in first

example but we want remember that in this type the main constraint is order constraint, that means the theoretical subject is must be before the lab subject.

In the university in this semester, it has two labs and the colleges will study are IT and Software Engineering.

If we want to view the input data and the relation between them, the icon execute make this statement for the list of persons, subjects and rooms.

4 DISCUSION

After reading the behavior examples, we show that genetic algorithm is a good technique to solve the timetable problems, but not excellent because if the program found many errors in the timetable, it needs more than one time to solve all the problems. For example, if the timetable has five problems, the number of iterations to solve the problem is three.

Another point, the optimal parameter values in E-learning timetable generator is very important to extract the reports, so we want to consider the three criterions in the generator:

- Size of population: it is better to start with the low, but important size of population. Thirty individuals are sufficient for deadlock prevent.
- Probability of mutation: the higher is better. Of course, the mutation of all individuals is very far from classical GA, the optimal value of probability of mutation 0.8 is chosen.
- The number of individuals in a tournament: high number of individuals causes very frequent deadlock in the local minimum. Suited value is 2 or 3, which give the higher selection gradient.

5 CONCLUSIONS

We have shown that a Genetic Algorithm approach is very effective and useful on the lecture time tabling problems. Using the methods we have described shows great potential for leading to timetables in future which are fairer to students.

The framework seems directly applicable to a very wide variety of other timetabling problems. For example, experimental results show that a key aspect towards its success is the employment of the mutation operators described. As we have shown, extensions of the basic technique to handle room constraints.

The GA in timetabling framework has been shown to be successful on several real problems of 'University Department size', and so it seems we can justify the expectation for it to work very well on most other problems of similar size and nature. That is, there is no reason to suspect that there is anything particularly easy about the problems it was tested on, in comparison to other real problems. Much work remains to do to see how performance scales to larger and otherwise different kinds of timetabling problems.

In summary, we have shown that GA in timetabling framework, involving a good solution to correct and optimization the errors and give the reports about the lecturers, classes and rooms.

6 FUTURE WORK

Future work is under way to more thoroughly test the performance of our technique on a wider variety of timetabling problems.A GA based timetabling system very similar to the one we describe could arrange for parallel sessions to be organized such that delegates are collectively satisfied as far as possible, in terms of minimizing the degree to which two presentations a delegate wishes to see are scheduled to occur at the same time.

Further work is wanted to perform comparisons in terms of speed and quality between the individuals. Considering details of the GA in timetabling framework itself, the reason why not is that so far we have used the GA mainly to resolve important stated constraints, but not for aesthetic and similar issues which often concern human timetablers. So, there might be some value in a post-processing module that can:

- Move events from one slot into an adjacent slot, subject to not violating any additional constraints, with the goal of packing the rooms more completely so as to reduce room bookings and invigilation requirements.
- If a set of events is at 16:00 and there are no events at 14:00, then shift all events into that time period, subject to not violating any additional constraints. Similarly for shifting from 9:30 to 11:30. The goal is to try to keep events near the middle of the day, which keeps everyone happier.
- Assess the average spread of the events in some way, that is, if we have 2 timetables that satisfy all of the constraints, the one with the larger average event spread is the better. This gives the students more time between events.
- Insert more constraints to extract the excellent results for the timetable, for example the level of the students is very important in the timetable.

REFERENCES

[1] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, A GENETIC ALGORITHM TO SOLVE THE TIMETABLE PROBLEM, Centre for Emergent Computing, Napier University, Edinburgh EH10 5DT, UK,2000.

[2] J. J. Grefenstette, editor. *Proceedings of the First International Conference on Genetic Algorithms and their Applications.Practice and Theory of Automated Timetabling VI Proceedings of The 6th International Conference on the Practice and Theory of Automa*

[3] J. J. Grefenstette, editor. *Proceedings of the Second International Conference on Genetic Algorithms and their Applications.* Practice and Theory of Automated Timetabling VI Proceedings of The 6th International Conference on the Practice and Theory of Auto.

[4] N. R. Jennings.Coordination Techniques for Distributed Artificial Intelligence. University of London Mile End Rd.London E1 4NS UK, 1995.