University Scheduling using Genetic Algorithm

Ossam Chohan

Master Thesis

2009 **Nr: E3719D**

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

DEGREE PROJECT
Computer Enginering

HÖGSKOLAN
Dalarna

Program **Master in Science in Computer Engineering**	Reg. Number **Nr. 434343**	**Extent** **15 ECTS**
Name **Ossam Chohan**	Year Month Day **2009-03-05**	
Supervisor **Dr. Pascal Rebreyend**	Examiner **Prof. Mark Dougherty**	
Company/Department **Department of Computer Engineering, Dalarna University**		
Title **University timetable scheduling using Genetic Algorithm**		

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Note:

This thesis was started by two students, and later they changed their directions, therefore content similarities can be considered as a combine work.

Ossam Chohan

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Abstract.

The automated timetabling and scheduling is one of the hardest problem areas. This is because of constraints and satisfying those constraints to get the feasible and optimized schedule, and it is already proved as an *NP Complete* [(1)] **[1].** The basic idea behind this study is to investigate the performance of Genetic Algorithm on general scheduling problem under predefined constraints and check the validity of results, and then having comparative analysis with other available approaches like Tabu search, simulated annealing, direct and indirect heuristics **[2]** and expert system. It is observed that Genetic Algorithm is good solution technique for solving such problems and later analysis will prove this argument. The program is written in C++ and analysis is done by using variation in various parameters.

---

(1) In computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC, NP standing for Nondeterministic Polynomial time) is a class of problems having two properties:

- Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP.
- If the problem can be solved quickly (in polynomial time), then so can every problem in NP.

Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Acknowledgement

I would like to acknowledge the continuous and endless support of Dr Pascal Rebreyed in formulating the basic idea and shaping it into practical form which ultimately helped me at each stage to understand the topic better. Another inspiring thing is his best supervision on the entire work.

Also I cannot forget the motivation provided by Dr. Siril Yella for his valuable guidance not only in thesis work but in all academic activities. My words cannot acknowledge him, still my heartiest thanks for him.

A special acknowledgement to Dr. Kennath of Statistics Department for putting me on the right track, and gave me detailed presentation regarding this topic and his work regarding the topic which later proved an important milestone to complete this work.

At last but not least I cannot forget technical suggestions and guidance given by my class fellows and friends in different parts of the world. They were really a source of inspiration for me and their participation in technical discussion during lectures and later in my thesis work helped me to think beyond some certain boundaries. It would be fair to say that without their help, it would not be possible for me to finish this work.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Table of Contents

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Chapter 1: Introduction

Manual timetable management has been a nightmare for many years and may still be. Today we are progressing in science and technology and everything is turning into automation so this issue has been in active research for the past many years. Real life situations can be classified into two categories: First is the category in which all problems have some deterministic or non deterministic solutions and there are many problems falling in the second category which has **no solution for all possibilities**. This is because of large *search spaces* [(2)], and handling such search spaces with respect to some certain circumstances. We think it is because of the variety of expectations we have from the solution. Therefore it is fair to say that no perfect generic solution has been possible till now. Scheduling of university time table is one of this kind, because of many constraints (varies from university to university, class to class, and session to session).

This work shows the usage of Genetic Algorithm in timetable scheduling to get the possible optimal solution. Although it is proven that such problems are **NP-Complete** problems **[1]**, still fair solution can be obtained and I tried to prove that.

This thesis work consists of three phases. First phase will explain the method of Genetic Algorithm, its operator usage with respect to their effect. Phase 2 is bringing Genetic Algorithm in application but after having a slight discussion about benchmarks and generating different schedules along with their respective effectiveness. Last phase will highlight some of approaches to deal with the same problem; this will open new doors for other students to continue this work by those approaches and guidelines provided. Related research can be viewed in literature. **[16]**

---

*(2) When solving some problem, we are usually looking for some solution, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution is) is called **search space** (also state space). Each point in the search space represents one feasible solution. Each feasible solution can be "marked" by its value or fitness for the problem. We are looking for our solution, which is one point (or more) among feasible solutions - that is one point in the search space.*

Chapter 2: Scheduling

"***Scheduling*** *is the process of deciding how to commit resources between varieties of possible tasks. Time can be specified (scheduling a flight to leave at 8:00) or floating as part of a sequence of events*" **[3]**.

Complexity [(3)] of scheduling task is totally dependent on the number of instances and constraints. But in case of limited resources and strict time and cost restrictions, it is very hard to formulate schedules.

In university timetable scheduling, the proven NP-completeness property lead solutions to be satisfactory but sub-optimal [(3)]. In fact there are two type of constraints involved, critical and soft constraints. The primary concern of research is to satisfy critical constraints and then soft constraints, but it is extremely hard to handle soft constraints **[4]**, despite the fact that we limit the scope of soft constraints but their violation is always alarming toward the efficiency of the solution.

Dalarna University (DU) is my case study (although this work is not addressing DU particularly) for testing purpose, has two campuses, and right now at both places time tables are managed manually and provided for faculty and students through a web application named "TimeEdit".

(3) In computational complexity theory, the amounts of resources required for the execution of algorithms is studied. The most popular types of computational complexity are the time complexity of a problem equal to the number of steps that it takes to solve an instance of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm, and the space complexity of a problem equal to the volume of the memory used by the algorithm (e.g., cells of the tape) that it takes to solve an instance of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm. This allows to classify computational problems by complexity class (such as P, NP ...). An axiomatic approach to computational complexity was developed by Manuel Blum. It allows one to deduce many properties of concrete computational complexity measures, such as time complexity or space complexity, from properties of axiomatically defined measures.

Automated timetabling is a special class of scheduling because no general solution can be provided. Different approaches are available to sort out this problem like Genetic Algorithm **[16]**, Tabu Search, Evolutionary Algorithm and Artificial Intelligence **[21]**, and Particle Swarm Optimization (PSO) and simulated annealing.

There are different Scheduling software **[5]** available in market but the problem is the lack of generality to make these tools flexible according to the demands of different universities and institutions. In other words the specific code requirement to make the whole algorithm as per requirements of respective institution is the biggest problem.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Chapter 3: Genetic Algorithm

The concept of evolutionary algorithms evolved during the last 40 years and the father of these approaches is J.H. Holland [HOL75], USA who described in his book "Adaptation in natural and artificial systems" (MIT Press, Cambridge, MA, 1992) and De Jong's "Adaptation of the behavior of a class of genetic adaptive systems," (published in 1975).

What Holland's method described was a method for classifying objects, selecting breeding with these objects to produce new ones. This methodology is the same as modeling Darwinian natural system following the simple natural pattern of growth and reproduction. A description of this genetic (evolutionary growth) can be described as follows:

```
Create a population of objects (creatures)        //Initialization
Evaluate the fitness of each object               //analyzing
While the population is not fit enough            //Fitness check
                    {
Delete (kill) all unfit objects                   //Removing unfit objects
While population size <max:                       //size check
                    {
Select two best populations
Create new objects
Random mutations
Evaluate and place in population                  //breeding
                    }
                    }
```

Figure-1: Evolutionary growth

The above figure gives some clear overview of genetic algorithm operations, little variations (like data type) are possible but top level overview is the same as given above.

The genetic algorithm acts upon genes chromosome. Every object or creature has its own chromosome.

The following picture is showing the significant processes of GA such as Initializing, breeding, mutating, choosing and removing unfit creatures. Although the improvements in GA has changed its way as compared to recommended by Holland but the basic idea is still the same.

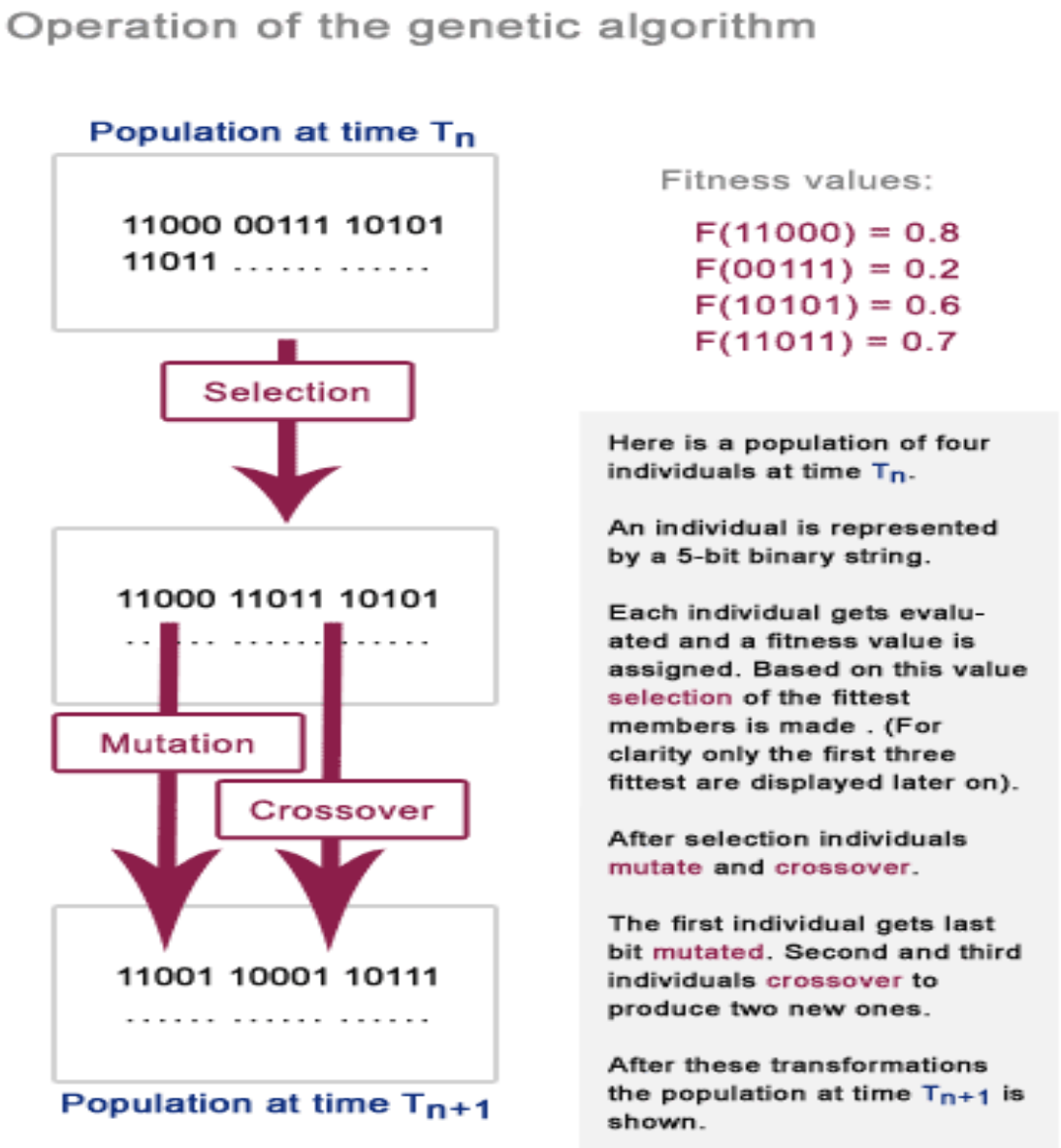

Figure-2: The basic Genetic Algorithm **[6]**

Population

Generally population is initiated as random allocation or assignment it but should be a legal one; it means that it should satisfy all the pre-requisite constraints and limitations of the problem under discussion. Unfortunately there is no particular methodology to decide about the size of population but one thing agreed upon is that it is all dependent upon the problem. Search spaces are also important phenomena to decide about the size of the population but according to some researchers for a regular search space population of 40 to 100 and for larger, more complex problems and irregular search space ,larger populations of 400 or more are recommended. The other factor in deciding the size is the convergence time, that for small size problems convergence might occur sooner and vice versa, but it all depends on the payoff decided between convergence time and exploration of search space.

Initializing can be done using two techniques. One is already established benchmarks with stored data and loading from secondary storage (in my case I made new and flexible benchmarks). Afterwards generating a random solution; This method is used by most GA. **[8]**

Other approach is gene by gene comparison by single offspring called overcrowding strategy, with continuous replacement of parents **[9]**

Tournament selection is another approach in which two creatures are chosen after having competition between each other, the winner produce news creatures **[10]**.

All of above approaches are almost same as recommended by Holland concept that healthiest should be allowed to breed so that more required creatures can be produced **[9]**. The reason of having same traits in all of the approaches is the main objective which is to obtain highest possible fitness value chromosomes **[9]**

Chromosomes Encoding

To provide simple, elegant and effective flow of GAs, Holland used a string of binary digits to encode chromosomes. But it does not mean that no other ways are available to

encode chromosomes, rather many other schemes are recommended along with their respective advantages **[8]**.

Phenotype and Genotype:

This is very well known a chicken and egg problem. In fact in GA the encoded objects are called genotype and actual object is called phenotype **[9]**, anything can be encoded because different approaches are available in literature. The only difference is storage of these genes, that in GA genes are not stored in pairs while in natural process are; pairs representing their parents' participation. **[8]**

Chromosome Evaluation:

According to Davis, randomly generated population are most of times extremely unfit **[8]**. Therefore evaluating the fitness is the most critical part of Genetic Algorithm or other approaches following same as Particle Swarm Optimization **[7]**. This evaluation is done on the basis of some predefined criteria (some sort of minimum or maximum value) and for that purpose we need to have information about environment. Multiple fitness evaluation is also possible, although we are not using this approach in our work, but some multiple fitness criteria can be introduced as introduced by Rich **[10]**.

The fitness evaluation will provide us with the fit (according to our predefined criteria) population. The most discussed criterion is cost. Cost is not money here but according to Gen and Cheng cost is a unit of efficiency **[8,9]**. Both types of cost either minimum or maximum according to the requirement of the problem can be set.

Crossover Operator:

According to the basic idea of Darwinian Theory the fittest individual tends to survive, and in above mentioned process, once the best parents are selected then breeding process will take place to improve the average fitness of the whole population. In fact these best individuals will mate and form the next generation having better traits than their parents.

According to genetics theory if the parents are stronger in terms of fitness (particularly in our case study) then the offspring will be fitter. So experimentation showed that this logic can lead us to a better solution.

There is no specific method associated with crossover. To maintain randomness we generated random numbers and then followed the process according to that random number. Cross over emulates the exchange of chromosomes (better offspring of parents) to generate a much fitter new generation. After selecting parents, we generate a random number between 0 and 1 and compare it with the cross over probability (usually it is taken 0.7), If the randomly generated value is less than crossover probability then the crossover takes place otherwise parents are simply placed into the next generation. This process can be seen in the figure below.

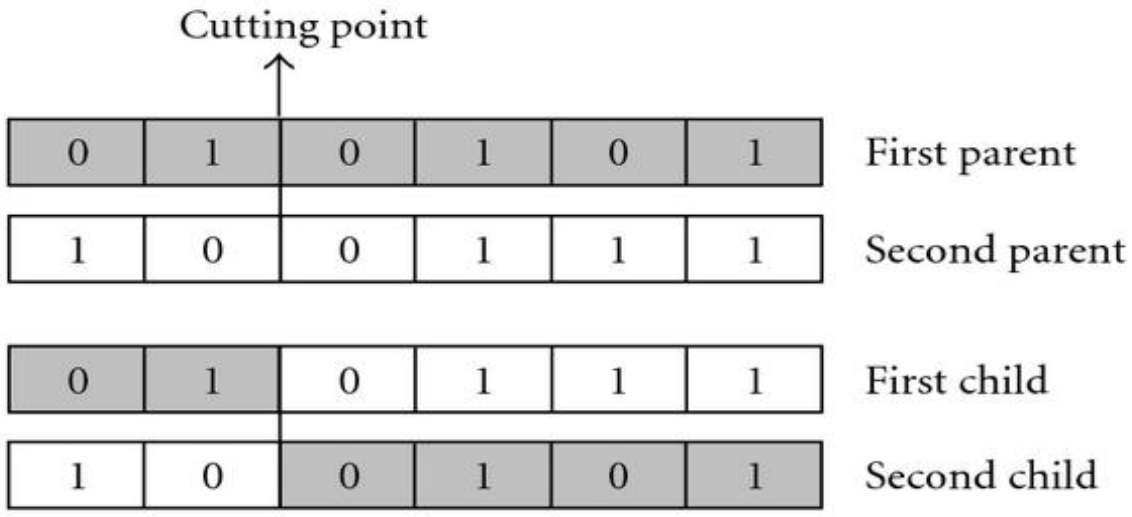

Figure-3: Crossover [23]

But there are many different methods available in GA to do crossover, a deep testing is still required to find out the efficiency of these methods. One point, two point or multi-point cross over is possible **[9].** Heitkoetter recommended using a unity order based crossover in which each gene has equal probability of coming up from either parent, so everything is equally likely. **[11]**

Mutation:

The important question is why to do this process. The major purpose is to introduce or inject noise and new alleles in the population. According to Gen and Cheng it is useful in escaping local minima as it helps explore new regions of the multi dimensional solution space. **[9]**. Mutation is not an integral part of Genetic Algorithm, but there are some systems in which mutation is not used, rather they prefer to use some noisy random populations at initial stage, which make effective search **[11]**.

Type of mutation is another important matter as there are different forms of mutation **[8]**. These methods vary with binary and non binary representation as in binary representation conventional mutation of 0 to 1 or vice versa can be helpful but for non binary representation and for more complex situation adding zero mean Gaussian number to original values is a recommended choice but it makes mutation more complicated.

The result of mutation operator cannot guarantee that it will provide a stronger solution, but it is believed that mutation is doing something new by changing some part of chromosome. But if it does not happen, then it cannot have any negative impact on the problem because only the fitter wins to stay; otherwise it will die out.

The following figure can represent the process of mutation:

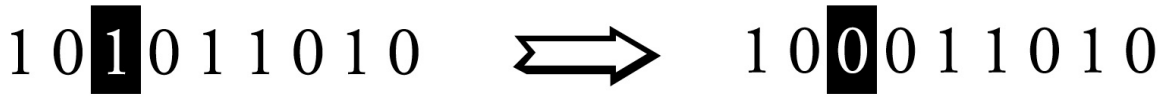

Figure-4: Mutation [6]

Inversion:

Other than conventional operators used to solve Genetic Algorithm another operator introduced by Holland's work is Inversion. **[8]**. Inversion is a simple process detaching the portion of chromosomes and after changing the direction, attaching it again with the chromosome. Generally due to its difficulty level (at implementation level), this method is not used while using GA **[8]**. But according to many known researchers in the field of GA this can play an important role in future **[8,9].**

Chromosome	
Before Inversion:	00100101101010010101101001010010101000101 0
During Inversion:	00100101101010 01010110100101 00101010001010
	01010110100101
	One portion inverts: becomes
(Order is reversed)	**10100101101010**
Recombination:	00100101101010 10100101101010 00101010001010
After Inversion:	00100101101010101001011010100010101000101 0

Figure-5: Inversion

Other GA operators:

Gen and Cheng also suggested some other operators like

- **Lamarckian operator**
  - Lamarckian operator is good effort to solve the problem of convergence. In the following two figures it can be seed easily that as compare to the normal GA convergence, a hybrid GA with Lamarckian operator converge well. It can be good future work if somebody wants to work on same task using Hybrid GA.

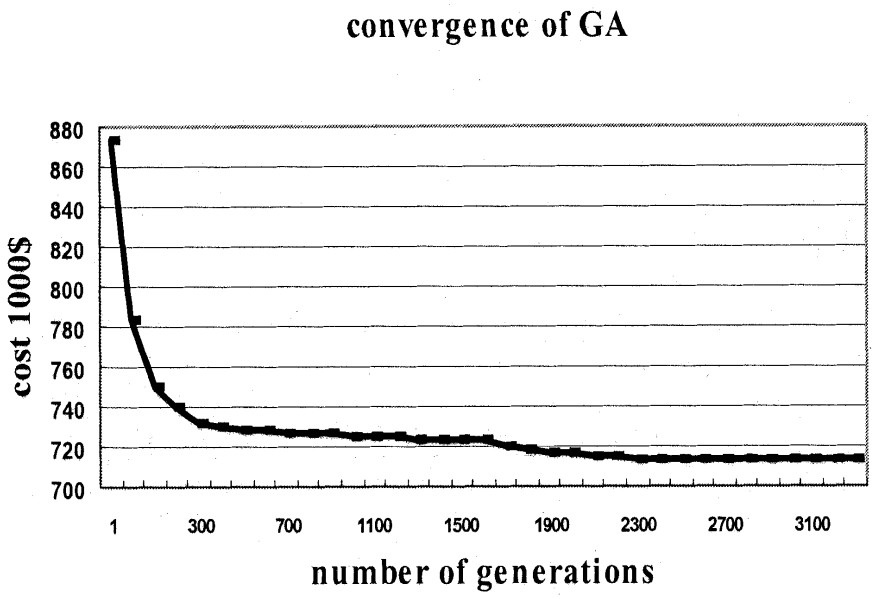

Figure-6: Convergence of Genetic Algorithm **[31]**

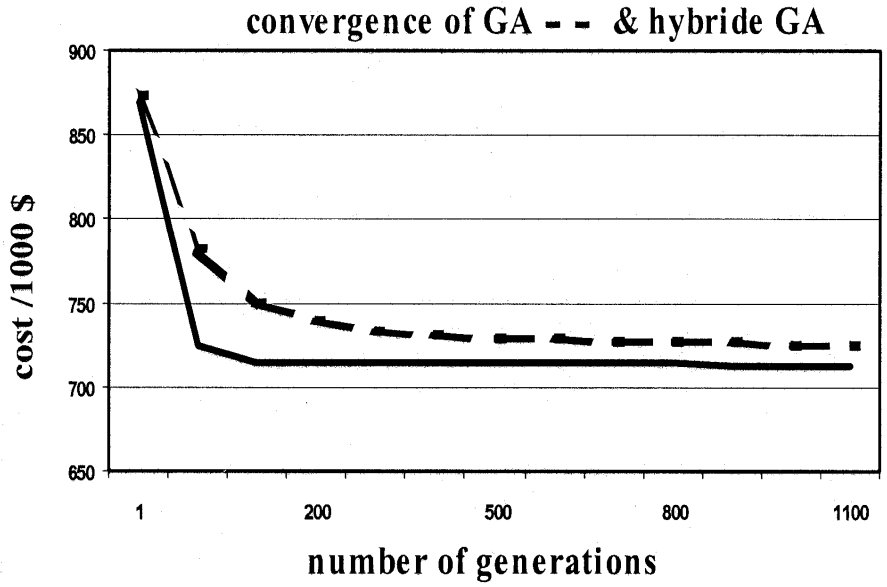

Figure-7: Convergence comparison of GA vs. HGA **[31]**

- **Memetic Algorithm**

  - *Memetic Algorithms* is a population-based approach for heuristic search in optimization problems. They have shown that they are orders of magnitude faster than traditional *Genetic Algorithms* for some problem domains. Basically, they combine local search heuristics with crossover operators. For this reason, some researchers have viewed them as *Hybrid Genetic Algorithms*. However, combinations with constructive heuristics or exact methods may also belong to this class of metaheuristics. Since they are most suitable for MIMD parallel computers and distributed computing systems (including heterogeneous systems) as those composed by networks of workstations, they have also received the dubious denomination of *Parallel Genetic Algorithms*. Other researchers known it as *Genetic Local Search*.**[32]**

- **Repair strategies**

The GeneRepair enhanced genetic algorithm operates in the manner of traditional genetic algorithms, and can be summarized as follows:

1. Generate the initial population P(0) at random and set i = 0;
2. Evaluate the fitness of each individual in P(i);
3. Select parents from P(i) based on their fitness.
4. Apply standard crossover
5. Apply standard mutation.
6. Apply GeneRepair.
7. Repeat until convergence. **[33]**

Figure-8: Gen Repair

Final Step in Genetic Algorithm:

After applying all of above (or few of them) steps, again fitness is checked as done in the first step of initialization.
But still no deterministic solution is there saying that how many times GA should run for. Simple problems may converge to good solutions after only 20 or 30 generations. More complex problems may need more. It is not unusual to run a GA for 400 generations for more complex problems such as job shops. The most reliable method of deciding on this is trial and error, although a number of authors have suggested methods for determining how long a solution should live. **[3]**

Application of Genetic Algorithm:

Although GA is entitled for many listed applications but it can be used for any function involving minimizing or maximizing function **[8]**, but GA proved good for pipe network optimization problems **[13]**, Transportation problems **[9]**, conformational analysis of DNA **[8]**, image processing and machine learning **[14]** and at last but not least the problem we are working on is scheduling **[15]**.

Chapter 4: Benchmarks and programming logic

Guide lines for Benchmarks

Important points:

1. There are three benchmarks. One each for rooms, time slots and for groups.
2. After giving inputs for these benchmarks, these benchmarks produce benchmarks in the form of text files.
3. Required data for schedules are stored in respective benchmark.
4. The algorithm for Genetic Algorithm loads the data while making the schedule, after comparing it with constraints to avoid any type of violation.

Benchmark-1 (name SBM-1)

Input: Number of rooms to schedule

Output: A benchmark text file in following format (sample format):

Room ID	Capacity	Type
A 301	40	Lecture (LC)
T 506	60	LC
A 125	70	C. Laboratory (LBC)
B 362	30	LBC
S 236	40	LC
T 258	50	Other Labs (LBO)
……	70	LBO
…….	……	LC
	……	LBC

Figure-9: Sample benchmark for Rooms

Set the capacity for each room using following standard:

Percentage of rooms	Actual size	Capacity
50%	N	30
20%	N	40

10%	N	50
10%	N	60
10%	N	60+
Where N is input of total number of rooms to be scheduled.		

Figure-10: Room capacity criteria

Explanation:

1. The above criteria is set only for having schedule, for future purpose slight changes in parameters setting, according to the requirement
2. Assign the room capacities randomly to each room using following standard:
   - 60% of n should be lecture rooms including all type of capacities.
   - 30% of n should be computer labs.
   - 10% of n should be other labs. (Like Electrical or mechanical labs etc)
3. For Room ID: Room ID reflects very important information, as suggested in this work; it should be 4 places ID. First place should be alphabet and rest of the places should be integers.

   For example if we have T 355, it means it is lab on third floor and number is 55, hope it is giving fair idea why we want to keep id like this.

Benchmark-2 (name SBM-2)

Input: Number of Groups required to schedule.
How many slots required by a group. *

Output: A well designed benchmark with filled data in following format:

Group	Group Description
1	M.Sc CS (AI)
2	M.Sc CS(OR)
3	BS Phy
4	…….
5	…….

6	…….
7	.....
8	…….
9	……..

Figure-11: Group Benchmark

Explanation:

The only reason to introduce this benchmark separately is to reduce the dimension of second benchmark. This benchmark handles the possible groups (in act classes for understanding). Then it would be easy to schedule these classes according to the teacher and time slots, while keeping consistent with the constraints.

Benchmark-3 (name SBM-3)

Input: How many time slots a group need in a week
How many courses can a professor can teach.

Output: A well designed benchmark with filled data in following format:

Time slot	Teacher	Room Requested
1	T1	LC
2	T2	LBC
3	T3	LBC
4	T4	LC
5	T5	LBO
…	…	LC
…	…	LC
…	…	….
….	…	….

Figure-12: Sample benchmark for time slots

Explanations:

1. Dimensions of above matrix will be set after taking input which is how many groups we have to schedule, and how many time slots a group needs.

*. Here in my case, it is assumed that each group needs equal number of slots per week.

2. Students group means, BBA-1, MSC-2, B.S Physic and etc.
3. Time slots means, there are 4 time slots are there in a day, as (8-10, 10-12) AM and (1-3, 3-5) PM. and we are designing schedule for weekly basis, so there are (5*4)=20 time slots are available in a week for each room.

Example: Suppose there are 5 groups of students available and each group needs 6 time slots per week, it means we have to schedule 30 time slots in a way that no constraint is violated (Constraints are explained in next section)

Pseudo Code:

Get the input of the number of groups and number of slots each group needs
Assign each time slot to groups in a way that their maximum required time slots do not increase.
Assign a teacher to group, but in a way that each professor should not exceed their maximum limit specified as input.
Out of all of slots reserved for a group, two third should be in lecture room (LC), and rest in laboratories. For example if a group needs 6 time slots, then 4 should be lectures and 2 should be labs.
End

Role of constraints in Automated Timetabling:

The schedule we are looking for should essentially be consistent with the constraints set required for the schedule. There are some universal constraints set by the people (students, professors, administration) dealing with that schedule **[15]**. According to Burke and Ross constraints can be divided into two categories; soft and hard (called critical in this work).

Critical Constraints:

Critical constraints (CC) are the backbone of the system devised. Any of the violation in CC can fail the schedule. For example one professor cannot appear in two time slots **[10]**.

Critical Constraints:	
CC1:	No professor appears more than once in one time slot.
CC2:	No room appears more than once in one time slot.
CC3:	Room allocation should be done after evaluating the size* and kind** of class.
CC4:	No professor can appear in two simultaneous time slots.
CC5:	One course cannot appear in two time slots in the same day.
CC6:	Multiple class subjects should be allocated to the same time slot.
*Size:	Size mean seats capacity, it should be more than registered students to avoid the problem in case of any guest student(s).
**Kind:	Kind of class can be lecture or laboratory.

Figure-13: Critical Constraints

Soft Constraints:

Most of the times soft constraints are set according to the preferences of the professor and sometimes with respect to the class preference: different researchers presented different types of soft constraints and it is according to their respective research domains.

Soft Constraints:	
SC1:	A professor may not prefer to teach morning or evening session.
SC2:	A professor may not prefer to teach more than certain number of credit hours.
SC3:	A professor can refuse to teach certain class.
SC4:	Lectures should evenly be distributed during week.

Figure-14: Soft Constraints

Timings can be divided into two categories:

First two time slots (8-10, 10-12) AM----------Morning session

Last two time slots (1-3, 3-5) PM----------Afternoon session.

Chapter 5: Other Approaches

Tabu search [16]:

A metaheuristics approach, developed by two researchers Glover (1986) and Hansen (1986) independently **[12]**, considered a strong approach to solve large and difficult combinatorial optimization problems **[22]**. As timetabling is difficult combinatorial optimization problem, this approach applied extensively to solve and make schedules. Although having its own merits and demerits (described below), according to the Brucker (1995), "Tabu search is an intelligent search technique that uses a memory function in order to avoid being trapped at a local minimum and hierarchically canalizes one or more local search procedures in order to search quickly the local optimality"**[14]**.

To improve the efficiency of the exploration process, some historical information related to the evolution of the search is kept. Such information will be used to guide the movement from one solution to the next one avoiding cycling.

Merits:

The Application of Tabu Search requires the definition of the

1. **Feasible solution**

   It is not defined based on the hard constraints because it does not guarantee that the corresponding search space is connected with respect to neighbor relation.

2. **The neighbor relation**

   The neighborhood of a solution consists of all timetables that can be obtained by assigning a lecture to a different period.

3. **The objective function to minimize**

   It is the weighted sum of the number of teacher and students' conflicts for each period.

Demerits:

The work in Tabu search contains the optimization problem with the grouping option, whereas it extends the approach to a more complex case which takes into account also lectures of different length **[19]**.

Constraint Logic programming approach:

CLP is widely and successfully used for finite domains and solving large combinatorial problem. A timetabling is the same kind of problem having a number of resources (Rooms and Teachers) with certain conditions (constraints) to schedule in a way to get the optimum or near to optimum solution by making maximum utilization of resources possible. Declarative handling of constraints (both hard and soft) is the real beauty of CLP. Since Timetabling is proved to be NP complete and though a number of soft wares are available in market, still due to the variety of constraints in certain situations, it is inflexible, and this ultimately makes it a challenging field of research.

Merits:

As mentioned above the declarative representation of all the constraints is the real beauty of this approach

Demerits:

The full back tracking capability of the prolog machine is overridden by a heuristics that allows only for a limited attempt to reschedule assignments that create conflicts.

Simulated annealing:

This approach is used for examination timetabling. **[18]**.The presentation can be repeated more than once if it is necessary. Here, the number of rooms is fixed and all the rooms are used for all the periods.

Merits:

It has an ability to provide good solution for many combinatorial problems and relative ease of implementation. It is relatively easy to code even for complex problems. It can also deal with arbitrary and cost function.

Demerits:

The main drawback of this method is that it cannot tell whether it has found an optimal solution. It requires other methods like 'branch and bound' to do this.

Particle Swarm Optimization:

This idea is based on the pattern of birds and fish, the way they coordinate collective behaviors.
The behavioral model follows the rules below: **[20]**

- **Separation**. Each agent tries to move away from its neighbors if they are too close.
- **Alignment**. Each agent steers towards the average heading of its neighbors.
- **Cohesion**. Each agent tries to go towards the average position of its neighbors

The basic algorithm:

1. Create a 'population' of agents (called particles) uniformly distributed over X.
2. Evaluate each particle's position according to the objective function.
3. If a particle's current position is better than its previous best position, update it.
4. Determine the best particle (according to the particle's previous best positions).
5. Update particles' velocities according to
   $\mathbf{v}_i^{t+1} = \mathbf{v}_i^{t} + \varphi\, \mathbf{U}_1^{t} (\mathbf{pb}_i^{t} - \mathbf{x}_i^{t}) + \varphi\, \mathbf{U}_2^{t} (\mathbf{gb}^{t} - \mathbf{x}_i^{t})$
6. Move particles to their new positions according to

   $\mathbf{x}_i^{t+1} = \mathbf{x}_i^{t} + \mathbf{v}_i^{t+1}$
7. Go to step 2 until stopping criteria are satisfied

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

A number of research directions are currently pursued:

- Matching algorithms (or algorithmic components) to problems.
- Application to different kind of problems (dynamic, stochastic, combinatorial)
- Parameter selection. (How many particles, which topology?)
- Identification of ”state-of-the-art” PSO algorithms (comparisons)
- New variants (modifications, hybridizations)
- Theoretical aspects (particles behavior, stagnation)

Time Tabling approach can be used in various application like:

Bruke and Ross used the same approach for examination timetables, explained in this work and it is believed having almost the same nature as timetabling has **[2]; [15]**. Minor changes are required to make this approach available for exam timetabling, like avoiding two exams in one day, two exams on two consecutive day **[17]**. Exam schedule facilitates in some areas as number of students, timing constraints and bit complicated in a sense that we have to consider each and every student on an individual basis **[17]** because students study courses according to their own preferences, so this is the trickiest part of exams scheduling.

Another application is scheduling for transportation system in a large corporation, where optimization concerns are minimizing shipping time and cost, while maximizing clients' satisfaction and improving services.

Multi-level job-scheduling of Operating Systems can be tackled with the same approach, and intricate digital circuits **[9].**

Chapter 6: Experimental Methodology:

The program written to check the efficiency of the suggested approach while considering constraints and satisfying is named timetable.cpp. Numbers of tests are carried upon the program to evaluate the performance of Genetic Algorithm by having different constraints at different times:

Test-1:	Population Size.
Test-2:	Room Capacity test
Test-3	Teacher unavailability
Test-4:	Repetition of Teacher
Test-5:	Class clash test

For the purpose of required test, I created benchmarks and loaded the data dynamically and according to any criteria (dynamic), dummy data is loaded. I had data for rooms, time slots and groups. The beauty is that, almost all of the parameters can be set dynamically.

Test-1 **Population Size**

Idea: This test was a bit complex, because it was related to other factors as well such as mutation rate. So this analysis was done by fixing one population size and changing different mutation rates, and for carrying out the test on a mutation rate, same method was repeated for different mutation rates.

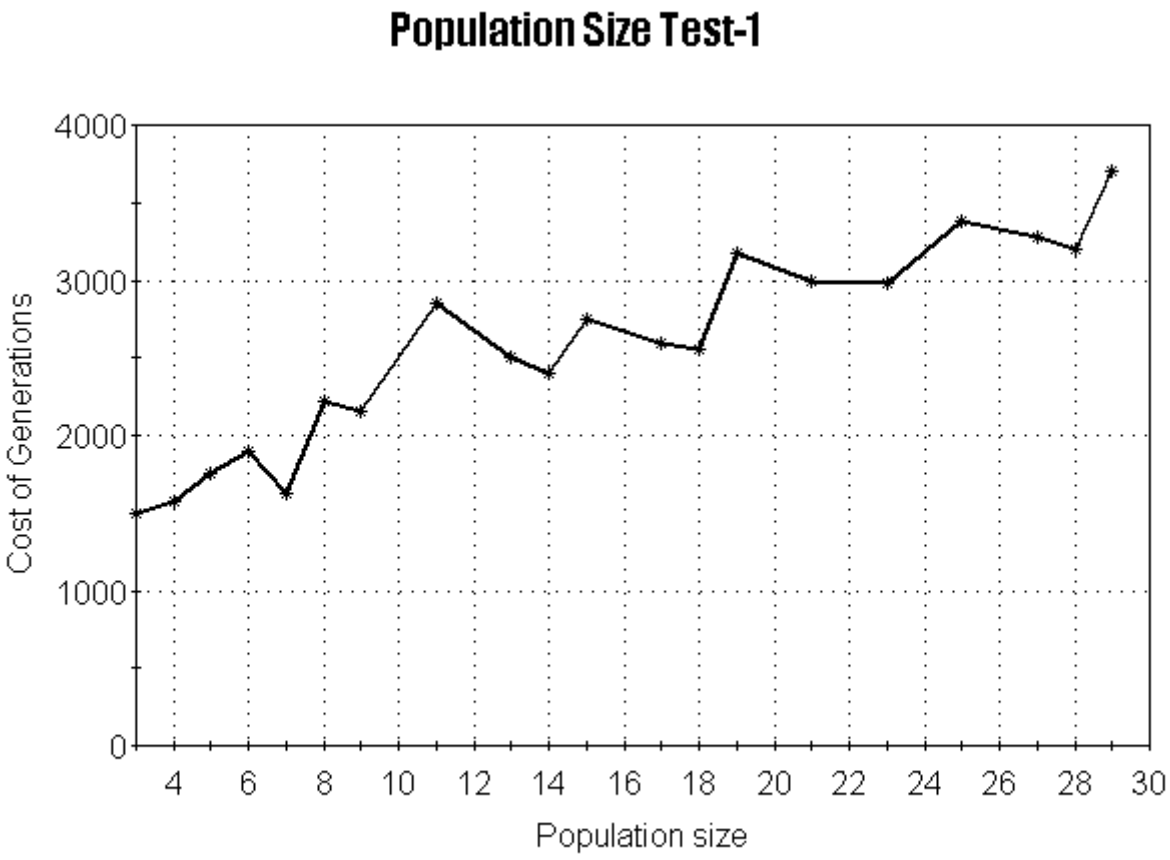

Figure-15: Population size impact with respect to generation

Description:

In fact this test was repeated for all population sizes (from 3-27), and it is felt that the system was taking more time. I am not certain about this but I think by increasing size, it will move to exponential class. Then cost is evaluated according to the fixed mutation rate presented in above figure.

Performance:

A very important finding for me is that smaller sizes work well and evolve more quickly but

as the size increased, time complexity also increased. what I think and somebody can test it in future is that if parallel machines are used with multiple processes, it can give better performance and for much larger population sizes it can be evaluated, and at the same time it will give more chances for chromosome having a low probability of selection (particularly in case of using Roulette wheel selection method). This will involve more competition between chromosomes and provide fittest chromosomes.

Test-2 Room Capacity Test:

Idea:

The objective of this test is to find out the number of classes which satisfied all of the constraints but the room capacity was not according to the requirement for that particular class. Another important dimension of this test is to check the possible number of iterations (in terms of Genetic Algorithm called generations), needed to avoid allocation in the rooms which does not satisfy the room capacity requirement. For smaller number of generations, error was very high (can be seen on graph attach with). The plotted results are average results.

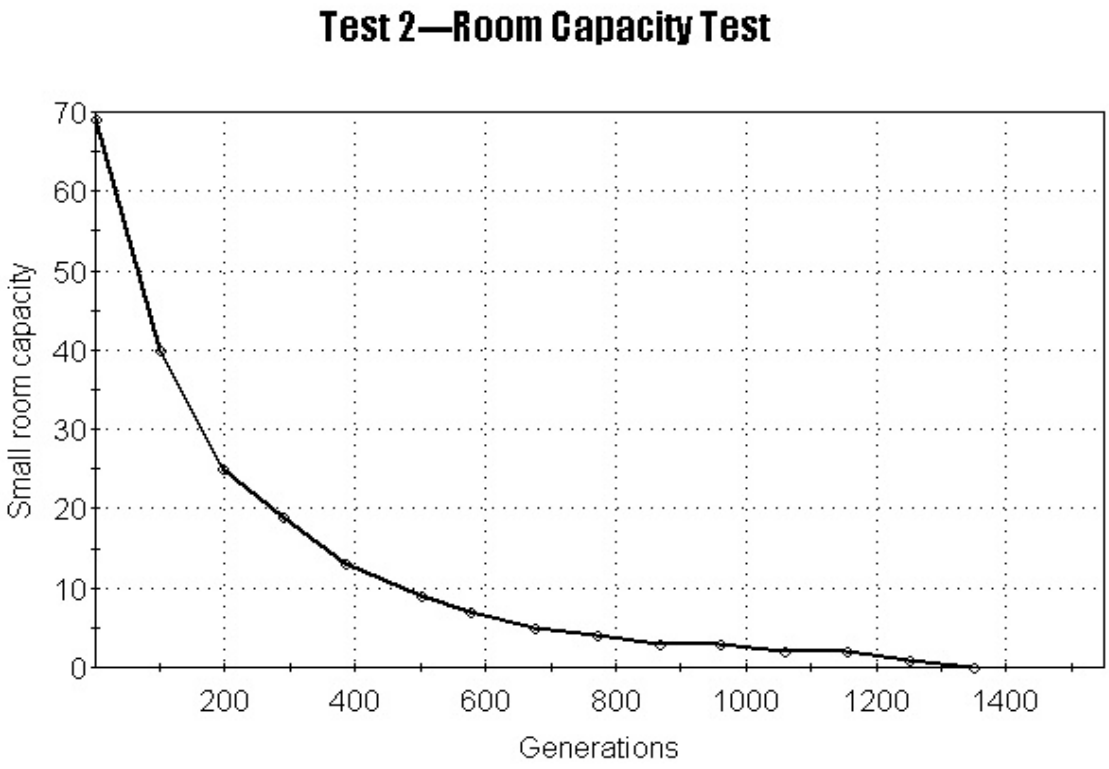

Figure 16: Room Capacity Test

Test 3 **Teacher unavailability test:**

The major problem in randomly generated schedules (as we created the initial schedule on random allocation basis, although satisfying critical constraints), is the increase in time complexity while removing the errors such as teacher availability and teacher should not book more than one time, and at the same time a room should not be booked more than one time. So the following pattern shows the error pattern of teacher unavailability. This is important to understand, since most of the time constraints are satisfied with other criteria but only teacher availability makes things complex. The average cost of population was recorded below:

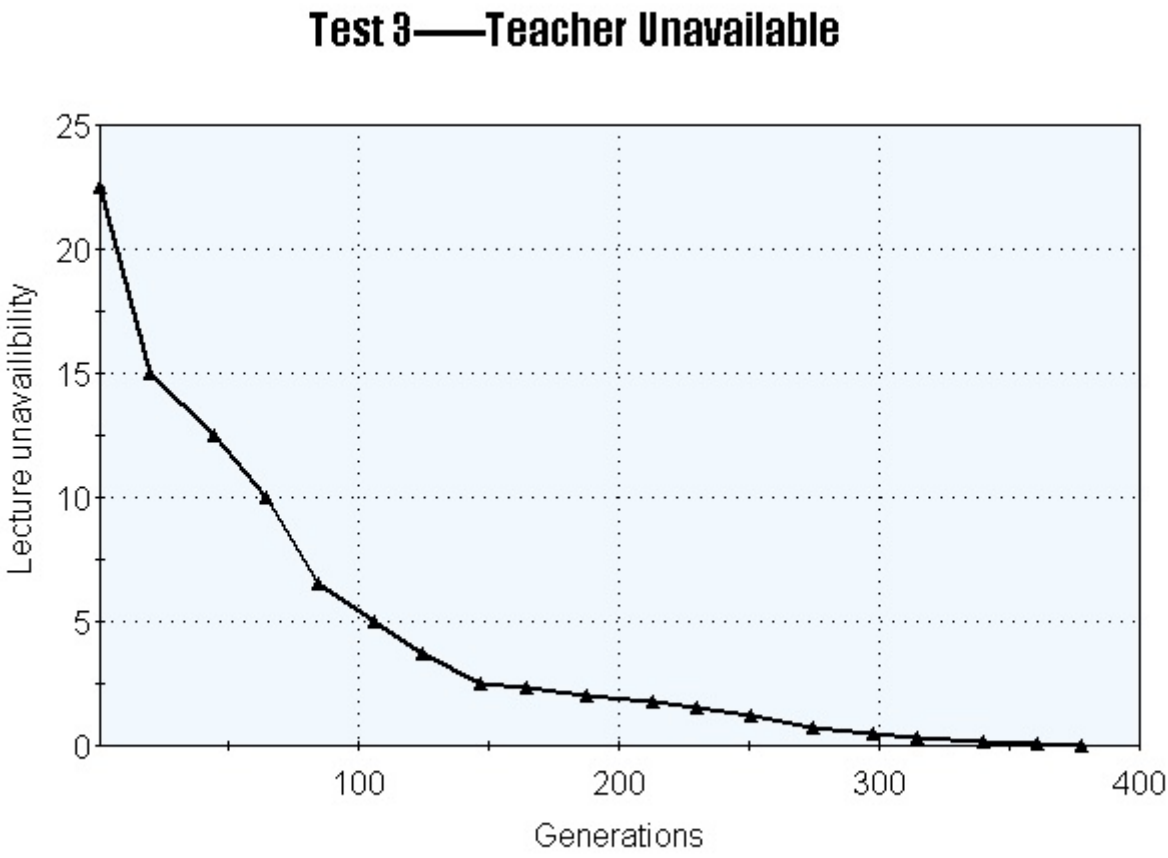

Figure 17 Teacher Availability Test

Test 4 **Repetition of Teacher**

Idea:

There were constraints regarding the maximum number of courses a teacher can teach during any learning period, or the other way around is that a teacher can take only two courses. And according to the time slots needed for each group (class in general), it can be decided that how many time slots a teacher needed. So repetitions of teachers were eradicated with the course of generations. Following figure gives the pattern of this error declining process.

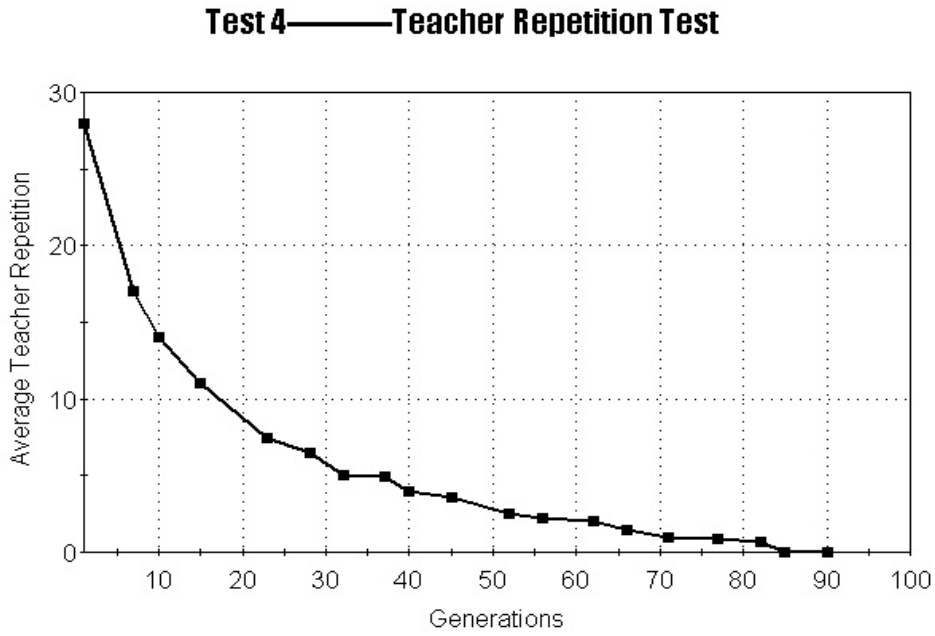

Figure 18 Teachers Repetition Test

Performance of test-2, 3 and test 4:

In both of the tests, the program worked well, that is improved error reduction. In case of room capacity problem, it took longer time to eradicate this error but I think this is because of initial random generation of schedules. Another important feature in this perspective is the inclusion of different critical constraints and all results are specific to constraint data. Group size (which is actually a class size) taken into account for this thesis is assigned randomly, and this can be a fair reason that error reduction and eradication took a long time.

Chapter 7: Statistical Perspective

Note: This idea was taken by the research work done by Andrei Petrovski, Alex Wilson and John Mccall "*Statistical identification and optimization of significant GA factor*"

Statistical method of inferences are considered one of the strongest mathematical tools for almost all of the disciplines, therefore GA have no exception. With the help of Statistical tests of significance, we can

- Identify GA factors significantly affecting the efficiency of GA
- Optimal values of such factors.

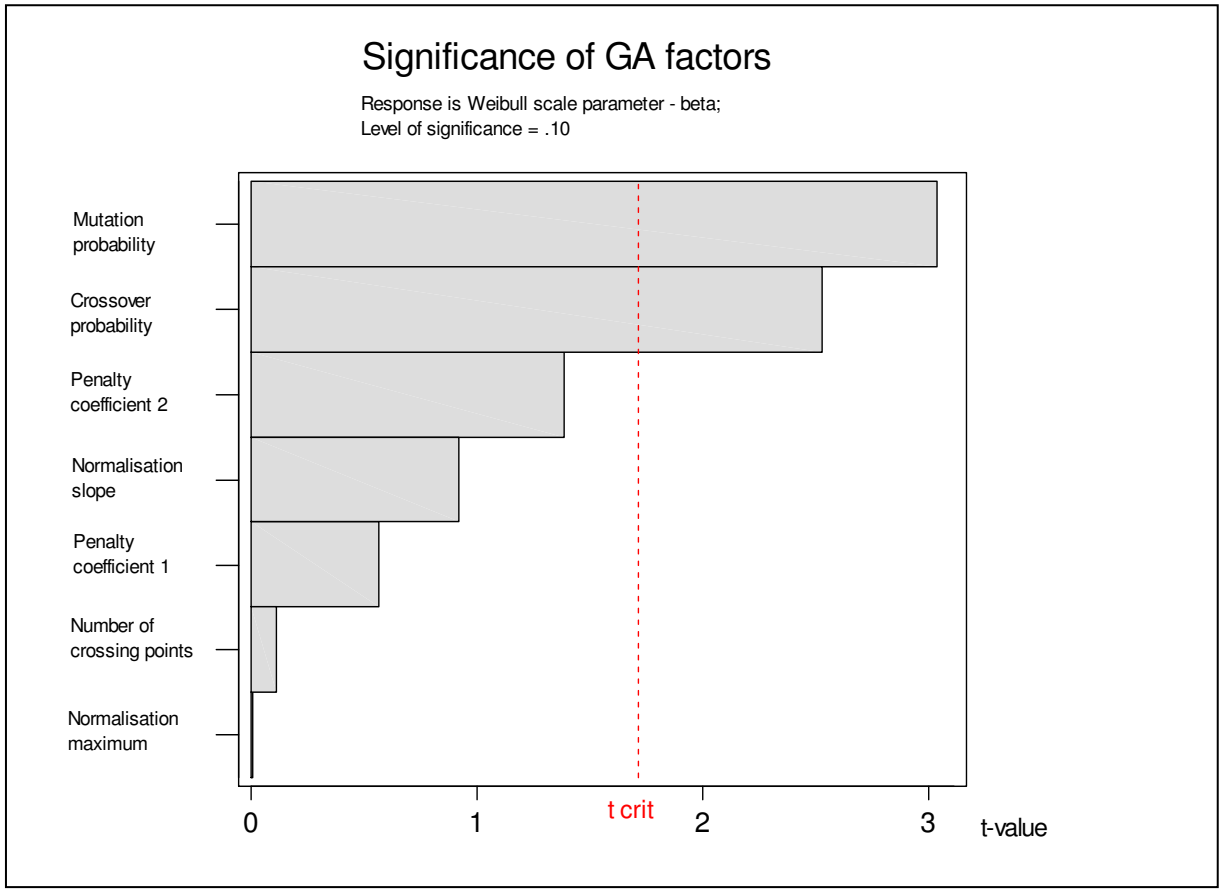

Figure -19: GA Factor analysis using t-test

The only and major advantage of this approach is to speed up search speed which is core issue in using Genetic Algorithm. Above figure gives a slight view of analyzing GA factors and their significance using t-test with significance of 0.10 (α=0.10).As time consumpotion is one of the disadvantage of evolutionary optimisation in general and Genetic Algorithms in particular, but after having the analysis done by the authors in their work, GA factors can be tune and ultimaltely improve the effeciency of GA.

There exist situations when due to either a large random variation in experimental responses or due to an inappropriate choice of factor domains the screening experiment cannot reliably identify the significant factors; then the consequent regression analysis becomes meaningless. What one may attempt to do in that case is to modify the factor ranges using an intuitive approach and to repeat the experiment. If this attempt fails, the meaning is that GA performance cannot be controlled by varying factors. Therefore, the process of factor tuning should be abandoned. Still, the merit of the statistical approach is that it necessitates the evaluation of much fewer factor settings (in comparison, for example, with meta-GA optimisation) to establish the fact that GA performance is not affected by the factors' values.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Chapter 8: Implementation Strategy:

The new system designing and development is research area and responsibility of computer scientist. An important question is how to replace the current manual system with the automated system. There are different approaches and schemes devised for this purpose (UPID MIS Study V1.0). But I would like to be very specific with respect to the proposed automated timetabling solution.

The main activities for implementation any newly devised automated system can be listed as follows:

- Planning the activities
  - Task assignments, time required parallel or independent implementation.
- Planning the activities and organizing the personnel.
- Acquiring the facilities required for implementation
- Developing step by step procedures, proper implementation regarding installation and testing.
- Developing the training program for operating personnel
  - Because the administrative staff cannot handle the technical stuff, so they need some training to understand the system and its functionality better.
- Designing GUI(for non-IT people) , Database (record keeping) and reports (history and future planning)

Proposed timetabling solution:

The solution this thesis proposed is not complete to implement as it needs two more things.

- Examination scheduling system
- Graphical User Interface for administrative purposes.

So this thesis is going to open some projects for students. One is to design examination scheduling system and secondly making an expert system solving complete scheduling

problem, after that this work will be having some impact factor and will be a useful work for university.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Chapter 9: Critics and conclusion

Critics on Genetic Algorithm

Although Genetic Algorithm is widely used for many types of application discussed above, and at the same time large number of applications are possible with other approaches such as Neural Network and Fuzzy Logic, there are very serious drawbacks attached with this approach.

Number of parameters (even large number) involved in the solution of Genetic Algorithm is a serious concern for different application developers. For simple constraint restricted problem (as my case study), Genetic Algorithm can work well, but in case of some complex problems it takes much more time to get some solution.[27]

Another problem is long trial and errors processes to reach the required (or optimized) solution with no certainty of reaching the target. Similarly another core barrier on Genetic Algorithm is the unique solution required for new problem of same domain [28]. Therefore evolutionary computational scientist is looking for some adaptive solution. One proposed solution to solve such problems is parameter free genetic algorithm (PfGA) [29]. PfGA also solve the same issues involved in steady state genetic algorithm (SSGA) [30]

Conclusions

This work presented genetic algorithm for general timetabling problem by keeping the general scheduling in mind. This approach was examined with different population size and mutation rates. Despite the critics discussed above on GA, in this problem GA provided good results.

Results obtained by many generations are quite satisfactory and can be used for further extension. The same approach can be used for industry applications, where tight constraints limit the functionality of operations and need optimization. Job shop is good example of such problem.

It seems that complexity will increase in case of adding more constraints, even with soft constraint. But one recommendation is that the same experiment should be checked on parallel computers with different processors for different task, and by using some grid approach recombines and try to make schedule

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

Chapter-10: Future Recommendation:

There are two types of extensions possible in this work; one dimension is that this work should be continued in the same direction by introducing more critical constraints and then leading it to the solution with soft constraints. As this problem is a hard one, therefore ultimate solution goal should be to have a complete expert system consisting of timetabling and examination system.

Another direction is consolidating examination system and making a well designed expert system. Testing of the proposed solution with other approaches discussed above will also lead to a better understanding of the efficiency of Genetic Algorithm.

Following are the possible future directions.

1. As per the instruction of my supervisor I tried to design very generic solution to the time tabling problem. Therefore an important area of work would be the real data of some university and then use same logic to design time table. It was found that there is a lot of variation in results, which might be due to random assignment behavior. So one can do the same work for specific data.
2. Repair strategy is one of the good approaches to fix errors occurring in solving such problems using GA.
3. Varieties of Genetic Algorithm techniques are available, so the same work can be done using other approaches.
4. A max-min ant system is also a successful story in solving time tabling problem **[25]**; therefore it would be a good option that if we can obtain some good result using GA, then on those results max-min approach can be applied. It will help in handling soft constraint, and dynamic nature of timetabling approach.
5. Variable and value selection heuristics also tested in University of Munich, achieved good milestones **[26]**.

6. What I recommend was that there should be some good blend of two or more approaches, because each approach addresses different direction of such problem, and to get a proper solution, multiple approaches can be engaged.
7. Benchmarking was the major hurdle while handling and it increases complexity with respect to memory, so some other data structures or different form of benchmarking can be used to improve the data loading at the initial run of the algorithm.
8. The same approach can be implemented in job shop scheduling, having multiple constraints (both critical and soft). This work can be extended in any of the market case study to know how effective these results are using Genetic Algorithm.
9. Another and one of the most important future extensions can be statistical analysis of Genetic Algorithm as discussed in above discussion named Statistical perspective.
10. Last recommendation would be an interesting piece of work. Although I am not sure that how it will work, but the results obtained are all from serial computers, and I am sure better results can be obtained by parallel computers with some grid approach, then NP Hardness can tackled.

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

List of figures:

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

References:

[1] S. Even, A. Itai and A. Shamir, "*On the complexity of timetable and multicommodity flow problems*", *SIAM J. Comput.* 5 (1976) (4), pp. 691–703.

[2] Burke EK, Newall JP and Weare RF(1995) : "*A Memetic Algorithm for University Exam Timetabling*". In Burke E and Ross P (Eds): *Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers.* New York: Springer-Verlag Berlin Heidelberg. pp 241-250.

[3] http://en.wikipedia.org/wiki/Scheduling (January 15, 2009)

[4] Shu-Chuan Chu, Yi-Tin Chen, Jiun-Huei Ho, *"Timetable Scheduling Using Particle Swarm Optimization"*, Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)

[5] http://www.mimosasoftware.com/

[6] http://www.catonmat.net/blog/genetic-algorithms-101/

[7] Marco A. Montes de Oca. "*Particle Swarm Optimization*"

[8] Davis L (Ed) (1991): "*Handbook of Genetic Algorithms*". New York: Van Nostrand Reinhold

[9] Gen M and Cheng R (1997: "*Genetic Algorithm and Engineering Design*". New York: John Wiley & Sons, Inc.

[10] Rich DC (1995): *"A Smart Genetic Algorithm for University Timetabling"*. In Burke E and Ross P (Eds): *Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers.* New York: Springer-Verlag Berlin Heidelberg. pp 181-197.

[11] Heitkoetter J (1993): *FAQ for" comp.ai.genetic Usenet newsgroup"*. Available from The Sante Fe Institute on the WWW at http://alife.santafe.edu/~joke/encore/www/

[12] Glover, F. (1986). *"Future Paths for Integer Programming and links to artificial intelligence"*. Computers and Operations Research 13: 533-549.

[13] Anderson A and Simpson AR (1996): "*Genetic Algorithm Optimisation Software in FORTRAN Research Report No. R136"*. Department of Civil and Environmental Engineering, The University of Adelaide

[14] Brucker, P. (1995) "Scheduling algorithms". Springer-Verlag, Berlin.

[15] Burke E and Ross P (Eds) (1996): "*Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference", Edinburgh, U.K., August/September 1995, Selected Papers* . New York: Springer-Verlag Berlin Heidelberg.

[16] Mieke Adriaen, Patrick De Causmaecker†, Peter Demeester and Greet Vanden "Tackling the university course timetabling problem with an aggregation approach"
Berghe KaHo Sint-Lieven, Information Technology
Gebroeders Desmetstraat 1, 9000 Gent, Belgium

[17] Ergul A (1995): "GA-based Examination Scheduling Experience at Middle East Technical University".
In Burke E and Ross P (Eds): "*Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference", Edinburgh, U.K., August/September 1995, Selected Papers.* New York: Springer-Verlag Berlin Heidelberg. pp 212-226.

[18] Eglese, R.W., and Rand, G.K. (1987), "Conference seminar timetabling", *Journal of the Operational Research Society* 38, 591-598.

[19] Gendreau, M., Hertz, A. and Laporte, G. (1994) "*A tabu search heuristic for the vehicule routing problem"*. Management science 40(10): 1276-1290.

[20] Craig W. Reynolds.
Flocks, herds, and schools: "*A distributed behavioral model"*.
ACM Computer Graphics, 21(4):25–34, 1987

[21] Corne D and Ross P (1995): *"Peckish Initialisation Strategies for Evolutionary Timetabling"*.
In Burke E and Ross P (Eds): *Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers.* New York: Springer-Verlag Berlin Heidelberg. pp 227-240.

[22] Ferland, J.A., Berrada, I., Nabli, I., Ahiod, B. Michelon, P., Gascon, V. and Gagné, E. (2001)
"Generalized Assignment Type Goal Programming Problem: Application to Nurse Scheduling". Journal of Heuristics 7(4) 391–413.

[23] http://www.hindawi.com/floats/861512/figures/2008.861512.fig2.xht (February, 12, 2009)

[24] http://en.wikipedia.org/wiki/NP-complete (February 12, 2009)

[25] Krzysztof Socha, Joshua Knowles, and Michael Sampels
"A MAX-MIN Ant System for the University Course Timetabling Problem"
RIDIA, Universit´e Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium

[26] Slim Abdennadher and Michael Marte *"University Timetabling using Constraint Handling Rules"*
Computer Science Department, University of Munich
Oettingenstr. 67, 80538 Munich, Germany

[27] Hagiwara M. Neuro, fuzzy and genetic algorithm.
Sangyo Tosho Publishing; 1994

[28] Kizu S, Sawai H, Adachi S. Parameter-free genetic algorithm (PfGA) using adaptive search with variable-size local population and its extension to parallel distributed processing. Trans IEICE 1999;J82-DII: 512–521.

[29] Masaaki Kanakubo, and Masafumi Hagiwara "Parameter-Free Genetic Algorithm using Pseudo-simplex Method.
Faculty of ICC, Keio University, yokohama, 223-8522 Japan

[30] Syswerda G. A study of reproduction in generational and steady-state genetic algorithm. In: Foundation of genetic algorithms. Morgan Kaufmann; 1991. p 94–101.

[31] Habib Rajabi Mashhadi, Hasan Modir Shanechi*," A New Genetic Algorithm With Lamarckian Individual Learning for Generation Scheduling"*
Senior Member, IEEE, and Caro Lucas, Senior Member, IEEE, IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 18, NO. 3, AUGUST 2003

[32] http://www.densis.fee.unicamp.br/~moscato/memetic_home.html (Feb 19,2009)

[33] George G. Mitchell, Diarmuid O'Donoghue, Diarmuid O'Donoghue, Mark McCarville, "*GeneRepair - A Repair Operator for Genetic Algorithms"*, Department of Computer Science National University of Ireland Maynooth, Co. Kildare, Ireland.

[34] Andrei Petrovski, Alex Wilson and John Mccall "Statistical identification and optimization of significant GA factors"
The Robert Gordon University, School of computer and Mathematical Sciences St. Andre Street, Aberdeen Ab25, 1HG, Scottland.

Sample Code for benchmarks:

Note: This code is not complete, this is just to give an idea that how to start working with benchmarks

```
//Made by Ossam Chohan (770404-T115) Dalarna University Sweden.
//---------------------------------------------------------------------------

#include <vcl.h>
#include <cstdlib>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
#pragma hdrstop

//---------------------------------------------------------------------------


class class_room
{
    public:
        int room_no;
        char *room_initial;
        int room_cap;
        int floor;
        char *des;
        int time_slot;
        int is_room_empty;

        int rooms_floor(int );
        void rooms_no(int ,int , class_room *);
        void type_rooms(int , int ,int ,class_room *);
};

int class_room::rooms_floor(int rooms)
{
            int floors;
            floors = rooms/100;
            if(rooms-(floors*100) > 0)
            {
                    floors++;
            }
```

```
		cout <<"total rooms : "<<rooms<<endl;
		cout <<"total floors :" <<floors<<endl;
		int temp=floors ;
		for ( int i =1; i<=floors; i++)
		{

				if(rooms - ((i-1)*100) >100)
				{
					cout<<"rooms on " << i <<" floor are 100" <<endl;

				}
				else
				{
					cout<<"rooms on " << i <<" floor are "<<rooms-(i-
1)*100 <<endl;

				}
		}
		return floors;
}

void class_room::rooms_no(int floors,int rooms_top, class_room *room)
{
		int temp=0;
		int capcity=0;
		int typ;

		int cap [] ={30,40,50,60,70};
		unsigned int seed=time(0);


   cout<<"total floors "<<floors<<endl;
		cout<<"rooms on top "<<rooms_top<<endl;

   for(int i=1; i <=floors; i++)
		{
				if(i<floors)
				{
					seed=time(0);
	srand(seed);
	for(int j=0;j<10;j++)
					{
						for(int k=0;k<10;k++)
```

```
                                                {

                room[temp].room_no=temp+1;
                room[temp].floor=i;
                room[temp].room_cap =cap[rand()%5];
                temp++;
        }
                                        }
                                }
                                else
                                {
                                        seed=time(0);
        srand(seed);
        for(int j=1;j<=rooms_top;j++)
                                        {
                                                room[temp].room_no=temp+1;
                                                room[temp].floor=i;
                                                room[temp].room_cap =cap[rand()%5];
                                                        temp++;
                                        }
                                }
                }

                cout<<temp<<endl;

}

void class_room::type_rooms(int floors, int rooms_top,int rooms,class_room * room)
{
                int temp =0;

                ////////////////// assing room type no, as lec, lab or others///////////
                for( int i =0; i < floors; i++)
                {
                        if ( i < (floors -1))
                        {
                                for(int j=0; j < 100 ; j++)
                                {
                                        if(j< 100*60/100)
                                        {

room[temp].room_initial="A";
```

```
room[temp].des="Lecturer Room";
                                                }
                                                else
                                                {
                                                        if(j< 100*90/100)
                                                        {
room[temp].room_initial="T";

room[temp].des="Labatory Room";
                                                        }
                                                        else
                                                        {

room[temp].room_initial="O";

room[temp].des="Other Room";
                                                        }
                                                }
                                                temp++;
                                        }
                                }
                                else
                                {
                                        for(int j=0; j < rooms_top ; j++)
                                        {
                                                if(j< rooms_top*60/100)
                                                {

room[temp].room_initial="A";

room[temp].des="Lecturer Room";
                                                }
                                                else
                                                {
                                                        if(j< rooms_top*90/100)
                                                        {

room[temp].room_initial="T";

room[temp].des="Labatory Room";
                                                        }
```

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

```
                                                                    else
                                                                    {

room[temp].room_initial="O";

room[temp].des="Other Room";
                                                                    }
                                                            }
                                                            temp++;
                                                    }
                                            }
                                    }



                                    ////////////////////////////////show contancts///////////////
                                    temp=0;
                                    for( int i =0; i < floors; i++)
                                    {
                                            if ( i < (floors -1))
                                            {
                                                    for(int j=0; j < 100 ; j++)
                                                    {

cout<<room[temp].room_initial<<room[temp].room_no<<"\t"<<room[temp].floor<<"\t"<<ro
om[temp].room_cap<<"\t"<<room[temp].des<<endl;
                                                            temp++;
                                                    }
                                            }
                                            else
                                            {
                                                    for(int j=0; j < rooms_top ; j++)
                                                    {

cout<<room[temp].room_initial<<room[temp].room_no<<"\t"<<room[temp].floor<<"\t"<<ro
om[temp].room_cap<<"\t"<<room[temp].des<<endl;
                                                            temp++;
                                                    }
                                            }
                                    }
                                            cout<<endl;

}
```

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

```
int ** banchmark1 (int floors, int top_floor)
{
		int **room_no;
		int i=0;
		room_no =  new int*[floors];
		for(i=0;i<floors-1;i++)
		{
				room_no[i]= new int [100*2];

		}
				room_no[i]=new int [top_floor*2];

				return room_no;
}
class class_group
{
		  public:
	int group_id;
					int group_size;
					char group_name[20];
					int time_slot_week;
					void get_group_data(class_group *,int );
};
void class_group::get_group_data(class_group * group,int total_groups)
{
							for(int i=0;i<total_groups;i++)
							{
									cout<<"Please enter size
of group :"<<i+1<<endl;
									cin>>group[i].group_size;

									cout<<"Please enter
group Name:"<<endl;

		cin>>group[i].group_name;

									cout<<"Please enter Time
slots per Week:"<<endl;

		cin>>group[i].time_slot_week;
									group[i].group_id = i+1;
							}
```

Dalarna University
Röda vägen 3
S-781 88 Borlänge
Sweden

Tel: +46 (0)23 778000
Fax: +46 (0)23 778080
http://www.du.se

```
}
class class_teacher
{
        public:
                int teacher_id;
                char  teacher_name[25];
                char  teacher_field[20];
                void get_teacher_data(class_teacher *, int);
};
void class_teacher::get_teacher_data(class_teacher * teacher, int total_teachers)
{
               for(int i =0 ;i<total_teachers;i++)
               {
                        cout<<"Please enter teacher
name:"<<endl;
                        cin>>teacher[i].teacher_name;
                        cout<<"Please enter teacher
field:"<<endl;
                        cin>>teacher[i].teacher_field;

                        teacher[i].teacher_id= i+1;
               }
}

class time_slots
{
        public:
                int slot;
                int group_id;
                int teacher_id;
                int is_empty;
};
class class_banchmark2
{
        public:
                time_slots time[20];
                 class_banchmark2();
                class_banchmark2* set_time_slot(int ,int ,int );
                void show(class_banchmark2 *,int );
};
 class_banchmark2::class_banchmark2()
{
```

```
		for(int i=0;i<20;i++)
		{
			time[i].is_empty = 0;
		}
}
class_banchmark2* class_banchmark2::set_time_slot(int total_rooms,int total_teachers, int
total_groups)
{
		class_banchmark2 * ban_mark2 = new class_banchmark2 [total_rooms];
		int rand_room =0;
		int rand_slot =0;
		int rand_teacher=0;
		int rand_group=0;

		for (int i =0;i<total_rooms*total_groups*20;i++)
		{
												rand_slot =
rand()%20;
												rand_room
= rand() % total_rooms;

rand_teacher=rand()% total_teachers;

rand_group= rand()% total_groups;

if(ban_mark2[rand_room].time[rand_slot].is_empty=0)
												{

			if((ban_mark2[rand_room].time[rand_slot-1].teacher_id !=
rand_teacher) &&

			(ban_mark2[rand_room].time[rand_slot+1].teacher_id !=
rand_teacher))

			{

				ban_mark2[rand_room].time[rand_slot].teacher_id =
rand_teacher;

				ban_mark2[rand_room].time[rand_slot].group_id =
rand_group;

			}
```

```
                                                                                }
                }
                return ban_mark2;
}
void class_banchmark2::show( class_banchmark2 * ban_mark2,int total_rooms)
{
                for(int i = 0;i< total_rooms ; i++)
                {
                                for(int j=0; j<20 ; j++)
                                {
                                                cout<<ban_mark2[i].time[j].slot<<"
"<<ban_mark2[i].time[j].group_id
                                                <<" "<<ban_mark2[i].time[j].teacher_id<<endl;
                                }
                }
}
void main( )
{
                class_room *room;
                class_group *group;
                class_teacher *teacher;
                class_banchmark2 * ban_mark2;

                int total_rooms,total_groups,total_teachers,floors;

                cout<<"Plz enter no. of room :"<<endl;
                cin>>total_rooms;

                cout<<"Plz enter no. of groups :"<<endl;
                cin>>total_groups;

                cout<<"Plz enter no. of teachers :"<<endl;
                cin>>total_teachers;

                room = new class_room [total_rooms];
                floors=room->rooms_floor(total_rooms);
                room->rooms_no(floors,total_rooms-((floors-1)*100),room);
                room->type_rooms(floors,total_rooms-((floors-1)*100),total_rooms,room);

                group = new class_group [ total_groups];
                group->get_group_data(group,total_groups);

                teacher = new class_teacher [total_teachers];
```

```
		teacher->get_teacher_data(teacher,total_teachers);

		ban_mark2 = ban_mark2-
>set_time_slot(total_rooms,total_teachers,total_groups);
		ban_mark2->show(ban_mark2,total_rooms);
		system("PAUSE");
}

//---------------------------------------------------------------------------
```