NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF APPLIED SCIENCE DEPARTMENT OF COMPUTER SCIENCE

April 30, 2022

FACULTY OF APPLIED SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

PROJECT TITLE: Timetabling using Genetic Algorithm

NAME OF STUDENT: Ashley Jordan Chihiya

STUDENT NUMBER: N01519508X

SUPERVISOR: MR. K. Mzelikahle

This project is submitted in partial fulfillment of the requirements of the Bachelor Honours Degree in Computer Science at National University of Science and Technology.

ABSTRACT

Scheduling remains one among the foremost difficult areas of research and study in computing. The University Course Timetabling downside (UCTP) formalizes the task of making a weekly course schedule for

students and lecturers at a university. Manual programming of university timetables stays inefficient because the previous year timetable is employed whereas some courses remain on the timetable that are removed from the syllabus. as the number of courses and students increase the amount of rooms and lecturers don't increase at an equivalent rate, and this creates a programming problem. Timetabling being an extremely affected combinatorial problem, this work can attempt to resolve this problem by applying Charles Darwin's biological process technique referred to as Genetic rule. Genetic Algorithms are nature impressed metaheuristics used to solve search and optimization issues.

ACKNOWLEDGEMENTS

This research project would not have been successful without the cooperation and support of a number of people. First, I would like to appreciate the Almighty God for the charitable time, strength and aptitude that enabled me to complete my research project. Secondly, my supervisor, Mr. Mzelikahle, thank you for your guidance, support and patience; I know it cannot have been easy at times. I wish to acknowledge the entire NUST fraternity, in particular the department of Computer Science for making the research possible.

CHAPTER 1

1 INTRODUCTION

Scheduling is a real-world problem that affects our day to day lives because of different constraints that exist.Today, scheduling is applied in educational foundations and industries as varied as procurement, production, transportation, distribution, information processing, communication, organizing, and education, Boddy, M. and Carpenter, T. (2007). Simple jobs can be "scheduled" manually. However, the moment the number of jobs, the activities in each job and the number of machines increase, the problem immediately becomes notorious for its complexity. Thus the need of these constraints to be managed effectively requires a certain degree of scheduling. Course timetabling falls into this category. It is a problem in which events (classes, courses, etc) have to be assigned into a number of time-slots such that conflicts in using a given set of resources are avoided. Timetabling is the scheduling of courses in such a way that there is a lecturer, room, time and students for each course offered. The problem with doing it manually is that as the number of variables increases the complexity and difficulty of task grows exponentially. It becomes tedious and time consuming as the person has to remember and manage all the constraints. To overcome this problem, universities need an automated system to organize their timetables. Therefore, it becomes essential to timetable courses with the help of modern computerised optimisation techniques as they have become vital in our modern society. If scheduling is done right, it allows organizations or educational institutions to achieve more while operating with a minimum of resources.

1.1 BACKGROUND

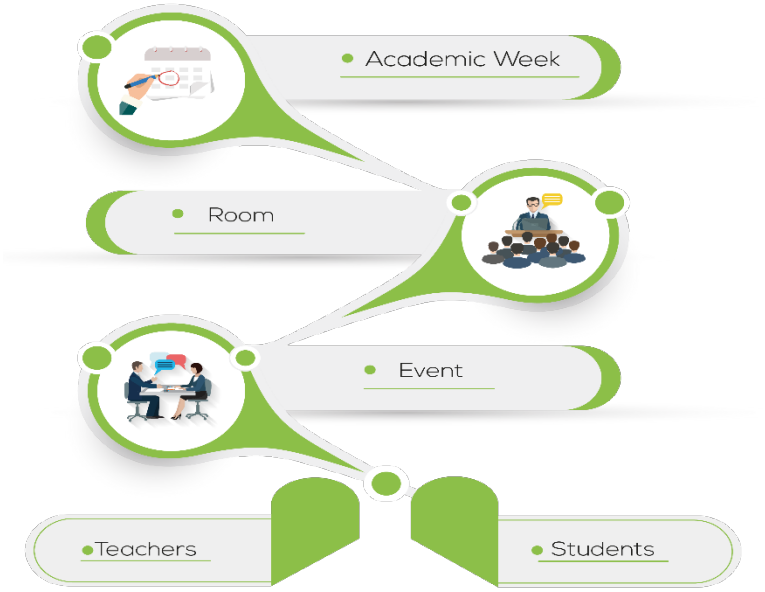


Figure 1: Timetabling problem

The timetable planning downside is common to any or all academic establishments. The main requirement is to eliminate or at least minimize the number of conflicts in the timetable. The constraints in timetabling can be divided into two categories: hard constraints and soft constraints. These constraints are needed to produce an optimal solution. Hard constraints are constraints that are inviolable (rigidly enforced) and must be satisfied in order to create a feasible solution. Example of a hard constraint is, two or more courses taught by the same lecturer cannot be scheduled for the same time slot.
On the other hand, soft constraints are those that are desirable but not absolutely essential.It is sometimes not possible to satisfy all soft constraints.

Thus, the acceptance of a feasible solution depends on the degree to which soft constraints are needed to be satisfied. Example of a soft constraint is, trying to avoid scheduling lectures from 1600 to 1800.
The other problem with a manual timetable is human bias. A human may make a biased timetable which favors some and is against others. It is prone to human errors which may come up because of human lack of concentration. The course timetable is usually done with the limited help of simple excel sheets and usually involves taking the previous year's timetable and modifying it hoping it works for the new year. In artificial intelligence, evolutionary algorithms and other different algorithms have attempted other different algorithms have attempted to solve timetabling.
A number of evolutionary algorithms exist and have been applied to solving the timetabling problem and the two most widely used are genetic programming and genetic algorithms. Evolutionary algorithms are stochastic search and optimisation heuristics derived from the classic evolution theory. In evolutionary algorithms there is a population of individuals, each individual is a potential solution to a problem and those individuals that are each closer enough to the solution are the ones that are bred to produce a new

generation. These evolutionary algorithms follow Darwin's evolution theory, which can be described in short as the "survival of the fittest" (Darwin 1859).

Genetic Algorithms (GA) exhibit the mapping from the natural process of evolution onto a computer system, because they stress the coding of attributes into a set of genes. One common coding of attributes is a binary coding into a bit-string representing the genes (Holland 1992). They are based on an analogy with the behavior and genetic structure of chromosomes within a population of individuals using the following basis: Individuals in an exceedingly large population contend for resources and mates.

Successful individuals in each 'competition' form the new population.
Traits of successful individuals propagate throughout the population so that two successful parents will sometimes produce offspring that are better than either parent. Each successive generation becomes more suited to their environment.

Genetic Programming (GP) is related to Genetic Algorithms regarding the general processing scheme. But instead of representing attributes in a general binary coding, genetic programming uses syntax trees (Khoza 1973).

1.2 STATEMENT OF PROBLEM

Finding the most effective allocation of lectures between lecturers and students over a finite number of timeslots and rooms, while satisfying a large range of various constraints.

1.3.1 AIM

The aim of this project is to develop a simple, easily understandable and efficient program using an effective and optimal Genetic algorithm, which automatically generates good quality time tables within seconds, while considering all constraints and violating none of the hard constraints.

1.3.2 OBJECTIVES

1. To identify hard constraints
2. To identify soft constraints
3. To optimize the slot and arrangement course timetabling process.
4. To schedule courses using genetic algorithm

1.3.3 JUSTIFICATION

Empirical evidence has supported the notion of using Genetic Algorithms for dynamically optimizing scheduling problems. The main reason being they accommodate a big search space, look for optimum and robust solutions. It is seen that the genetic algorithm could be improved by further incorporation of repair strategies, and is readily scalable to complete the scheduling problem.

1.3.4 PROJECT SCOPE

University timetable problem is divided into two categories, that is the examination timetable problem and the course timetable problem. Only the course timetabling problem will be covered in this research. A case

of National University of Science and Technology (NUST) limited to the Faculty of Applied Sciences and Department of Computer Science will be used.

1.3.4 EXPECTED RESULTS

We expect to evolve conflict-free course timetables (solutions). A lecture/student not be assigned to a course in the same time slot. No course in the same slot should be assigned to the same venue.

1.3.5 PROJECT OVERVIEW

1.3.5.1 CHAPTER 1 INTRODUCTION

Introduction sets the tone of the research and gives an overview of the project. It provides a summary of the background to the problem and its relevance to other work. The importance of the problem is highlighted and why it warrants investigation. Aims, objectives and scope are identified.

1.2.1 CHAPTER 2 LITERATURE REVIEW

A review of both recent and old literature relating to timetabling, detailing the approaches that have been applied by different researchers. In evaluating all the work that has been done previously in timetabling, it clearly highlights the areas where further research and improvements is required.

1.2.2 CHAPTER 3 METHODOLOGY

It outlines how the research is to be carried out starting from the problem to the solution. The research methodology adopted is explained and is incorporated into the problem so that it spells out steps to the solution. It also gives a brief description of a few other methodologies available to researchers.

1.2.3 CHAPTER 4 SYSTEM ANALYSIS AND DESIGN

Outlines what the system should do, its parameters, properties and resources used. It also outlines how the system would do showing relationships between parameters and or properties of the system

1.2.4 CHAPTER 5 TESTING AND IMPLEMENTATION

The experiments are set-up in this chapter and the parameters that are being varied to achieve the required performance are highlighted. The number of experiments and the number of runs is stated.

1.3.5.6 CHAPTER 6 CONCLUSION AND FUTURE WORK

Based on the results obtained from the experiments, the chapter concludes the research highlighting knowledge gaps available for future work.

1.3 CONCLUSION

Different scheduling software are available on the market, commercial and open source, but the problem is lack of generality to make these tools flexible according to the requirements of different institutions. Customizing the software will only get the institution closer to its requirements, hence continued research in the timetabling. Uni-time university timetabling, its research interests include constraint-based timetabling and scheduling and its applications to real-life large-scale problems. They are promoting research in this area by providing benchmark data sets for course timetabling, student scheduling and examination timetabling.

2 CHAPTER 2

3 LITERATURE REVIEW

2.1 INTRODUCTION

In this chapter, we give a general overview of the timetabling problem, specifically the university course timetabling problem and the constraints that should be considered. Different approaches that are used for solving the university course timetable in the literature are also reviewed.

3.1 TIMETABLING

Timetabling problem is a scheduling problem that is concerned with the allocation of events to time slots, factoring in constraints that exist and therefore the resultant solution constitutes a timetable [1]. Universities use timetables to schedule classes and lectures, allocating time and venues to events in the best method making use of available resources. Scheduling is one of the issues researchers have been researching over the years. The course timetabling problem that is an NP-hard problem is a type of scheduling problem. With the complexities related to this, it's necessary that each university, school or department comes up with a possible timetable. A timetable which will be effectively executed and meets the proposed requirements is a feasible timetable. This implies that each course must be scheduled within a time slot, located in a suitable classroom and assigned to a lecturer in order that each student registered for a course will attend the lectures. Without a possible timetable, not one university would reach its goals. Poorly designed course timetables will be expensive in terms of time and money, and an inconvenience to both lecturers and students. An optimal timetable will have an effect on a student's or lecturer's university life. Departments and faculties even have an influence on the development of course timetables. Depending on policy a university will have a centralized method of timetabling or a decentralized method of timetabling. Timetabling, for many years, has attracted the attention of many researchers as they explored different methods of achieving better feasible timetables. This great interest led to the creation of a series of conferences, Practice, and Theory of automated Timetabling (PATAT) in 1995 (Arogundale et al.2010).

3.2 UNIVERSITY COURSE TIMETABLING PROBLEM

Carter and Laporte (1998), defined a course timetable as:
"a multi-dimensional assignment problem, in which students and lecturers (or faculty members) are appointed to courses sections or classes, and events (individual meetings between students and teachers) are assigned to lecture rooms and timeslots".
It is a problem of scheduling a set of events to specific time slots such that no resource or students/lecturer is expected to be in more than one location at the same time. It should also be noted that classes of a course should be evenly distributed throughout the week to provide a conducive learning environment. Through the course timetabling exits in several variations, every single problem boils down to this: the main objective is to combine the following elements:

A set of courses $\mathbf{C} = [c_1, c_2 .... c_n]$ with n the number of courses

A set of lecturers $\mathbf{P} = [p_1, p_2 ..... p_m]$ with m the number of lecturers

A set of classrooms $\mathbf{R} = [r_1, r_2 ..... r_k]$ with k the number of classrooms

A set of timeslots $\mathbf{T} = [t_1, t_2 ..... t_{dh}]$ with d the number of days and h the number of timeslots per day

in such a way that every course **C** has assigned to it a lecturer **P**, a classroom **R** and a time slot **T: C (P, R, T)**. For example, course 8, taught by lecturer 3 in room 15 at timeslot 27 would be represented as '$c_8(p_3, r_{15}, t_{27})$'. This should result in a feasible timetable satisfying all the hard constraints and minimizing the violation of soft constraints. The terms used in timetabling are stated below:

(a) **Event**: It is an activity to be scheduled. Examples are resources.

(b) **Timeslot(period)**: It is an interval of time in which events can be scheduled.

(c) **Resource**: It is what is required by events. Examples include classrooms and equipment (i.e. projectors).

(d) **Constraint**: It is a restriction to schedule the events. Examples include classroom capacity and specific period.

(e) **Individual**: Is a person who has to attend the events.

(f) **Conflict**: Occurs when two events at least a common individual and are scheduled in the same time slot.

In course timetabling a set of courses is scheduled into a limited number of classrooms and periods within a week. Lecturers and students are also assigned to courses so that lectures can take place. Course timetabling problem, define a class of hard-to-solve combinatorial optimization problems and has hard and soft constraints. Hard constraints that should never be violated at any cost and they have to be satisfied for a feasible solution. Below are examples of hard constraints for the course timetabling problem.

1. A student and a lecturer cannot be in two venues at the same time.
2. Courses with students in common cannot take place at the same time.
3. No two courses can take place at the same time in the same room.
4. No lecturer may be assigned to different events at the same time.

The classroom assigned to the course should satisfy the features required by the course.

1. There must be a maximum number of time periods per day, which may not be exceeded.

The soft constraint should not necessarily be satisfied, but it is important to satisfy them if possible as this increases the quality of the timetable. Below are examples of soft constraints:

1. Students should not attend more than two consecutive courses on a day.
2. Students should not be scheduled to attend a course that is assigned to the last period of the day.
3. Each lecturer has his/her own availability schedule.
4. Students should not have one lecture in a given day.
5. The travel time of lecturers and students between classrooms within the campus should be minimized.

Pereira and Valdecy (2016) presented a university course timetabling with hard and soft constraints when they proposed a model that solves the university timetabling problem in a Faculty of Rio de Janeiro. In course timetabling a solution can be described as optimal and infeasible. An optimal solution satisfies all the hard and soft constraints and there is no violation of both soft and hard constraints. An infeasible solution fails to satisfy all or some of the hard constraints. A feasible solution satisfies all hard constraints but not necessarily all soft constraints.

The availability of literature for the timetabling problem has led to new insights, like improved methodologies and more comprehensive models. Due to a variety of constraints, diversity of the problem and specific requirements has made finding an effective and general solution in course timetabling more difficult. Despite the amount of research that has been done on this problem, a gap still exists between theory and practice. [2], identified that it is extremely difficult to find a generally applicable model because different institutions recognize different constraints. Studies that attempt to propose a model use specific datasets for proving their solution, which then excludes generalisability.

3.3 DEVELOPMENTS IN COURSE TIMETABLING

There has been notable progress regarding course timetabling, considering the research work found in surveys and literature. [3]) developed an algorithm, combining it with simple heuristics on a mainframe computer. It generated a timetable that was not optimal which required manual modifications. In the '80s, computational course timetabling was still not widely accepted. A few institutions had microcomputers by then and the thought of scheduling using a computer caused resistance. [4] proposed network, graph and mathematical ways for solving the timetabling problem. A survey conducted by [5]) illustrated how modern heuristics, like evolutionary algorithms, gave the impression to outperform the traditional operational research methods. Researchers were now able to generate feasible timetables in an acceptable time-frame. However, there were still many cases in which the problem was computationally too hard.

3.4 APPROACHES

Researchers have developed completely different approaches to deal with the university timetabling problem. Many papers are published that discuss or classify major solution methods. [6], discussed the work of other researchers with their different approaches to solving the course timetabling problem. From available both published and unpublished, the approaches have been used over the years include sequential methods,

constraint-based methods, cluster methods, meta-heuristic or hyper-heuristics methods, population-based methods, and recently case-based reasoning and knowledge-based ways. We will briefly review some of the approaches that have been accustomed to solve the course timetabling drawback, then narrow the scope to population-based methods in particular distributed genetic algorithms.

GRAPH-BASED METHODS The course timetabling problem is represented as a group of events to be processed separately from one another. The graph coloring problem is restricted to assigning a color to each of the vertices of the graph. Events (courses) are represented as vertices whereas conflicts are represented by edges. An example, if a lecturer is to teach two events there is an edge between the nodes that represent this conflict. If the number of vertices and edges becomes large the exact methods fail and, in such cases, the heuristic methods are used (Velin and Radoslava 2016). Graph-based methods are also referred to as sequential methods as they assign events (courses) into valid time slots in order that no events in the slot are in conflict with one another.

To construct conflict-free course timetables, a number of graph-based methods are used and they assign events based on how difficult it is to schedule them [7]. These are:

1. Largest degree first: All events with a large degree of conflict are scheduled early.
2. Largest weighted degree: it is a modification of the above in that it weights every conflict according to the number of students involved in the conflict.
3. Saturation degree: in the construction of the timetable an event with the smallest number of valid slots available for scheduling in the timetable constructed so far is selected.

These are the earliest set of algorithms for solving timetabling problems, they show great efficiency on small instances but not in large instances.

CONSTRAINT-BASED METHODS Constraint-based methods originate from research on artificial intelligence. These methods encompass constraint logic programming and constraint logic programming and constraint satisfaction methods. However, such methods are generally computer extensive by means of the increasingly exponential number of variables. In more recent literature, constraint-based methods are integrated with different heuristics and methods in order to keep better-performing methods (Vrielink et al.2016). Uni-time university timetabling currently uses this approach in both course timetabling and exam timetabling. Their software is commercial and open source. They work with all researchers interested in improving this approach.

CASE-BASED REASONING METHODS Case-based reasoning (CBR) has been applied to scheduling problems since the year 1997. Since then, more research has been done focusing on using CBR with different methods for timetabling optimization. It is a method of solving new existing problems by referring back and adapting previous case solutions to fit the new situations. CBR has been applied to various scheduling problems including the planning and scheduling of large-scale airlift operations, dynamic job-shop scheduling, etc. In CBR a number of previously solved timetabling problems. A number of researches have been done, which applied CBR in educational timetabling problems (Hong et al.1992).

CLUSTER BASED METHOD Babaei et al. (2015) describe cluster methods which can be thought of as representing a three-phase approach. In the first phase, the examinations are grouped into periods to construct a feasible timetable. The second phase attempts to reduce second order conflicts by considering permutations of timeslots. Then the third stage is employed with the aim of improving the solution quality further. This is done by moving a particular examination between time slots such as by employing a hill climbing local search.

META-HEURISTIC METHODS These are classified into two categories:

1. single-based approach
2. population-based approach

3.4.1 SINGLE-BASED APPROACH

A single-based approach works with a single solution and then tries to improve for a better result. The constraints are satisfied in an iterative manner.

HILL CLIMBING (HS) It is a classical local search technique also called simple descent. Hill climbing is simple and easy to implement. However, the disadvantage is that it is easily trapped in local optima. Therefore, researchers tend to hybridize hill climbing with other search methods such as meta-heuristic methodologies (e.g. evolutionary algorithms, simulated annealing etc.).

TABU SEARCH (TS) Tabu Search is a search algorithm that was originally created by Fred W. Glover in 1989. Tabu Search may be a local search algorithm. local searches takes a potential solution to a problem and assesses its neighbors, by distribution an evaluation score to each of them, so as to visualize if they are a higher solution. Neighbors during this case suggest solutions with some minor distinction from the answer in question. If a neighbor contains a higher evaluation score, and thus could be a better solution, it becomes the new current solution, and within the next iteration neighbors to the present solution are evaluated. This continues till some stopping criteria is fulfilled, just like genetic algorithms. Picking that neighbors should be evaluated in every iteration could be a key part of local search algorithms, and this is where Tabu Search comes in.

In Tabu Search, we tend to keep track of previously visited solutions in a tabu list. Any solution within the tabu list won't be visited again till an explicit condition has been fulfilled, as an example a certain number of solutions having been visited. This keeps the local search from getting stuck in a local optimum or a plateau, and helps it expand the search to different regions.

Algorithm 1 Tabu search Algorithm

1. Generate initial solution x.

1. Initialize the Tabu List.

1. **while** a set of candidate solutions X' is not complete.

1. Generate candidate solution x' from current solution x

1. Add x' to X' only if x' is not tabu or if at least one Aspiration Criterion is satisfied.

1. **end while**

1. Select the best candidate solution x$*$ in X'.

1. If fitness(x$*$) fitness(x) then x=x$*$.

1. Update Tabu List and Aspiration Criteria

1. If the termination condition met finish, otherwise go to Step 3

Table 2.1 Tabu Search Algorithm Pseudocode

SIMULATED ANNEALING (SA) Simulated annealing (SA) is another local search method, proposed by Kirkpatrick in 1983. It was motivated from the physical annealing process of heating up a solid to a high

temperature and slowly cooling it down until it crystallizes and no further changes occur. For each material, the cooling schedule was very important. Simulated annealing starts from an initial solution (generated using a constructive heuristic) and it will always accept an improved solution, while the worse solution is only accepted with a certain philosophy.

According to Dowsland & Thompson (1998), the cooling schedule has a large influence on the quality of the final solution. Faster cooling schedules tend to lead the search to converge to local optima, while a slower cooling schedule generally produces a better-quality solution but increases the search time. Table 2.2 below shows the simulated annealing pseudo code.

Algorithm 2 Simulated Annealing Algorithm

1. Generate initial solution x.

1. Assign Temperature to Initial Temperature.

1. Generate candidate solution x' from current solution x

1. If fitness(x')> fitness(x) then x=x'.

1. If fitness(x') <= fitness(x)then calculate Acceptance Probability.

1. If Acceptance Probability? random [0,1] then x=x'.

1. Update Temperature according to the cooling schedule.

1. If the termination condition is met, finish, otherwise go to step 3.

Table 1: Table below shows the simulated annealing pseudo code.

This approach aims to search for a wider area of search space in which worse steps are accepted and allows for a more extensive search for the optimal solution. Simulated annealing encompasses a certain number of variants but is often combined with hill climbing techniques and constraint programming. In line with a local search-based method, a recent trend recognizes the definition of more different neighborhoods. Structures like variable neighborhood search and large-scale neighborhood search are associated with such techniques.

Single based methods begin with one initial solution and employ search strategies that try to avoid local optima (Vrielink et al.2016). All of these search algorithms can produce high-quality solutions but often have a considerable computational cost. They are utilized due to their capacity for solution space exploitation, and they work with a single solution and then try to improve for a better result.

GREAT DELUGE ALGORITHMS (GDA) Dueck (1993), introduced the great deluge algorithm (CDA) that operates in a similar way to simulated annealing(SA). However, GDA uses an upper limit (often referred to as the water level) as the boundary equal to the initial solution quality. It accepts worse solutions if the cost (objective value) is less than the boundary that is lowered in each iteration consistent with a predetermined rate (known as the decay rate). Silva and Obit in 2008 used a non-linear decay rate to manage the boundary and allow the boundary to rise when its value is about to cover with the current solution. Experiments on the course timetabling problem revealed that the non-linear CDA gives good quality solutions. They compared the performance of the great deluge, an algorithm with simulated annealing and hill climbing, and concluded that GDA was superior to the other two algorithms.

GDA has additionally been hybridized with alternative methods. (Abdullah et al.2009) hybridized GDA with TS. Their algorithm applied four neighborhood moves and selected the best solution that was generated. If there is no improvement within a specified time, the boundary is increased randomly within a value zero and three. The approach gave good results when applied to the course timetabling problem. Based on the above, it is shown that GDA is ready to produce a good quality solution.

POPULATION-BASED METHOD A population-based method aims to find the best solution in the search space and it starts with many candidate solutions. They are utilized due to their capacity to search space exploration and can be easily combined with local search methods based on the capability of recombining solutions to obtain new ones. Ant Colony Optimisation is an example of a population-based method that keeps track of information gathered during a search, subsequently, this information is used for generating new solutions in the next stages. Another population-based method is evolutionary algorithms which encompass genetic and memetic algorithms. [8], conducted research on the use of genetic algorithms in educational timetabling and provided a survey on this. Such algorithms are based on the best individual solution in the population space and each best solution provides the basis for a new evolutionary cycle.
Ant colony optimization is a population-based method proposed by (Dorigo et al.1996). ACO is inspired by the behavior of ants, and the way they forage for food (that is, through cooperation by depositing trails of pheromone). Table 2.3 shows the ant colony algorithm pseudo code.

Algorithm 3 Ant Colony Algorithm

1. Initialize pheromone values

1. Release each ant in the colony to construct an independent solution through components

1. Update pheromone values

1. If the termination condition is met finish, otherwise go to Step 2

Table 2: shows the ant colony algorithm pseudo code.

When ants are walking to and from the food source they deposit pheromones that gradually evaporate over time. Ants have a probability of following the pheromone trail proportional to the strength of it. Initially, each randomly chooses a route to a food source. Due to random fluctuations, one route will develop a stronger pheromone trail than the others and therefore attracts more ants.

The ants that randomly choose the shortest route will be the quickest to return to the nest and therefore this route receives pheromone earlier than other routes and it is more likely that ants will choose this route over others. As more ants follow the shorter path the pheromone trail strengthens until no ants follow the longer route. [9], proposed an ant colony optimization algorithm called ATNCOL for coloring graphs. The ANTCOL was applied to university scheduling problems by Downsland and Thompson (2005).

3.4.2 GENETIC ALGORITHMS

The most applied algorithms in terms of timetabling research are genetic algorithms which are also known as canonical genetic algorithms. Memetic algorithms have led to some success in the field by combining genetic algorithms with local search methods i.e hybridization. Genetic algorithms mimic the evolutionary process in nature by manipulating populations (also referred to as chromosomes) of solutions in the search space using crossover and mutation operations and evolving a new population.

Algorithm 4 Genetic Algorithm

1. Initialize a population of solutions
1. Evaluate the individuals in the population
1. **while** termination condition is not met do
1. Select parents through a selection scheme
1. Crossover parents to create offspring
1. Apply mutation to offspring
1. Replace individuals for the next generation
1. **end while**

Table 2.4: Genetic Algorithm Pseudocode

Genetic algorithms (GAs), start from an initial population of (often) random solutions. Table 2.4 outlines the genetic algorithm pseudo code. Each of these solutions is known as an individual and they each have a cost} value (fitness) evaluated based on the objective function.

A fitness function is a measure that evaluates the quality of the chromosome as a solution to a particular problem. The fitness function is problem specific. Its aim is to minimize a function. In most of the timetabling problems, quality of a chromosome is measured by a number of constraints satisfied (Sonawane and Ragha 2014). Next, is a selection part where the individuals are chosen by the selection operator to undergo the recombination method. In the recombination phase, crossover and mutation operators are used to explore the solution space, so creating new individuals replace old individuals (usually the worst individual based on their fitness). This process is repeated till a stopping criterion is reached, which may be the maximum number of generations or a time limit. Usually, the quality of solutions produced by population-based algorithms outperformed by trajectory methods. The explanation being caused by premature convergence, where population-based algorithms are more concerned with exploration than exploitation (Al-Betar and Khader 2008). Therefore, much recent research has involved the hybridization of genetic algorithms with

trajectory methods to optimize individual results.

Massoodian and Esteki (2008), implemented genetic algorithm-based approaches to solve the ITC2007 course timetabling problem. The approach consists of two stages with local search being applied (on the best body) at each stage to further refine the chromosome. The first stage concentrates on finding a feasible solution, while the second stage minimizes violations of the soft constraints. The approach was able to produce good solutions in less computational time compared to using GA alone. JAT in 2008, proposed a hybridization of a genetic algorithm with local search to solve the course timetabling problem from ITC2007. The problem is solved in two phases, with the primary phase the genetic algorithm uses information from previous good individuals to guide the generation of offspring with local search techniques to improve the quality of individuals. In the second phase, the tabu search is used on the best answer obtained to try and improve the solution. The experimental results show that the proposed hybrid approach outperformed the other tested methods.

The aim of using genetic algorithms is to get better solutions through a series of generations. Most of the research done in solving the course timetabling problem using canonical genetic algorithms have used direct representation(direct genetic algorithms). In recent years indirect representation (indirect genetic algorithms) has been accustomed to solve course timetabling problems and have generated better solutions compared to direct genetic algorithms. Raghvjee and Pillay (2008). The canonical genetic algorithm formulated by Holland et al. in the sixties is a sequential one. This made it easier to reason and understand the mathematical properties of the algorithms as there was a lack of hardware. With the supply of hardware with high computing power, genetic algorithms provide opportunities for parallel implementation.

3.4.3 HYPER-HEURISTICS

The need for an increased level of generality for automatically solving a range of problems motivated the development of hyper-heuristics. From literature most, meta-heuristics operate directly on a search space of solutions; however all a hyper-heuristic operates on a search space of heuristics (Burke et al.2003). The hyper-heuristic framework is given a set of pre-existing heuristics and therefore the task is to discover a good sequence to effectively solve the problem indirectly.

[10], used tabu search as a high-level heuristic to search through a space of moving strategies for course timetabling and nurse rostering problems. The proposed approach shows good results on each of the problems considering the generality of the approach. After two years, [11] improved on the above work aiming at investigating the learning of low-level heuristics that are suitable and effective for individual objectives in multiple-objective space allocation and course timetabling problems. The approach shows promising results compared with the state of the art approaches.

Burke et al. (2007), utilized tabu search hyper-heuristic using graph heuristics for the course timetabling problem. a set of low-level heuristics represents the search space. Tabu search is employed to randomly seek for the list of low-level heuristics while not considering the details of the actual solutions. The heuristic sequence is used to order the events(courses) that are not yet at that iteration. Qu and Burke (2009) add on the work from (Burke et al.2007) by proposing an adaptive approach rather than tabu search, where heuristics are dynamically hybridized during solution construction.

2.6 CONCLUSION

A review has been done on most of the approaches that have been applied to solve the timetabling problem. Each of the approaches has its own drawbacks and some approaches are suitable for certain timetabling problems. Single based algorithms focus on exploitation while population-based algorithms focus on exploration. New ideas have been developed in recent years to improve the quality of solutions. Distributed genetic algorithms (DGA) focus on both exploration and exploitation implying they are capable of combining speed and

efficacy. Genetic algorithms suffer from premature convergence, but DGA escapes premature convergence by maintaining diversity in the population through migration. Hyper-heuristics are being investigated further as they have produced promising results in solving scheduling problems.

CHAPTER 3
METHODOLOGY

3.1 INTRODUCTION

Methodology is the systematic, theoretical analysis of the methods applied in carrying out the research. Moreover, a methodology is the general strategy that outlines the method within which the project is to be undertaken and, among alternative things, identifies the methods to be utilized in it (Setzer, 2017). It outlines some research views and approaches that are used inside the field of computing. Research is a careful investigation through search for new facts in any branch of knowledge. It comprises defining and redefining problems, formulating hypotheses or recommended solutions; collecting, organizing and evaluating data; making deductions and reaching conclusions; and finally testing the conclusions to see whether or not they match the developed hypothesis.

3.5 RESEARCH METHODOLOGIES

FORMAL METHODOLOGY Formal methods are mostly used to prove facts about algorithms and systems. The research will focus on the correctness or the quality of the solutions generated by the genetic algorithm in solving the University Timetabling Problem to produce optimum solution without clashes.
Formal methods help provide comprehensive testing hence robust solutions are achieved through model checking — a technique that relies on building a finite model of a system and checking that a desired property holds in that model. Model checking is an exhaustive automated testing. It checks all possible states of all executions against the list of properties that executions should have.

Model checking is quick, automatic, and supports partial specifications.
The technical challenge in model checking is in making algorithms and data structures that can handle giant search areas.Above all, it produces counterexamples that typically represent delicate errors in design and might aid in debugging.

Formal methods are not only concerned with what is doable today but also with what will be possible in the future with new architectures, faster machines, and future problems. For instance, in the University Timetabling Problem, variables such as capacity of a room have to be considered, constant revision of courses as departments might change their curriculum over time.

BUILD METHODOLOGY A “build" research methodology is a powerful & consists of building an artifact either a physical artifact or a software system to demonstrate that it is possible. To be considered research, the construction of the artefact must be new or it must include new features that have been demonstrated before in other artefacts [12]. Build methodology is powerful because it enables the visualisation of

the structure of the system to be built. It produces design diagrams, data models and workflow diagrams. The researcher needs to put into practice the following practices:

1. Designing of the software system. Whether the system is complex or simple, it should be evolved with a plan. A modular approach is to be followed such that testing is made simpler.
2. Reuse of components. If some software components that are needed for the development of the system are already available, then they should be used in order to save time. When deciding which components to reuse, the researcher must consider the terms of use attached with them.
3. Choosing an adequate programming language. When choosing a programming language to use, the researcher must consider the following factors: required run-time speed, expressiveness, reliability which includes run-time checks and garbage collection, and the available libraries of that programming language.

Consider testing all the time. The system should not all be tested when it is entirely built; however modules should be tested first. This allows for future changes to be tested as they are introduced.
Once the software is useful, researchers ought to compare its functionality and/or performance with that of existing systems to verify that the claim(s) that they require to form regarding the
the system still holds.

4 CHOSEN RESEARCH METHODOLOGY

The build research methodology was implemented because it leads to the building of a software artefact. It focuses on designing the software before moving to development. The methodology was chosen because it emphasizes continuous testing which enforces early discovery of bugs and improves the overall quality of the developed system.

5 Tools Used in the project include:

1. FXML: is an XML-based language that provides the structure for building user interfaces separate from the application logic of your code.
2. Java programming language
3. Eclipse IDE: which is a Java IDE.

5.1 SOFTWARE METHODOLOGY

It comprises procedures, standards, techniques, an analytical framework, a philosophy and models in order to make system development more manageable [13].

6 RATIONAL UNIFIED PROCESS (RUP) METHODOLOGY

This methodology segregates the expansion process into four different stages that each includes business modeling, scrutiny and design, enactment, testing, and disposition.
RUP represents an repetitive approach that is superior for variety of reasons:

- It enables you to take into consideration dynamic requirements that despite the most effective efforts of all project managers are still a reality on almost each project.
- Integration is not one “big bang" at the end; instead, components are integrated more and more.
- Risks are usually discovered or addressed during integration.
- With the repetitive approach, you can mitigate risks earlier.
- Iterative development provides management with a method of creating plan of action changes to the product.
- It permits you to release a product early with reduced practicality to counter a move by a rival, or to adopt another seller for a given technology.
- Each iteration is risk-driven.
- Iteration facilitates reuse; it is easier to spot common elements as they are partly designed or enforced than to acknowledge them throughout designing.
- When you correct errors over many iterations, the result is a lot of strong design.
- Performance bottlenecks are discovered at a time once they will still be self-addressed, rather than making panic on the eve of delivery.
- Developers will learn on the means, and their varied skills and specialties are a lot of absolutely utilized throughout the complete lifecycle.
- Testers begin testing early, technical writers begin writing early, and so on.
- The development method itself is often improved and refined on the means.
- The assessment at the top of iteration not solely appears at the standing of the project from a product or schedule perspective, but also analyzes what should be changed in the organization and within the method to create it perform higher within the next iteration.

CHAPTER 4
SYSTEM ANALYSIS & DESIGN

6.1 SYSTEMS ANALYSIS

It is a problem-solving technique that improves the system and ensures that each one the parts of the system work with efficiency to accomplish their purpose. A Use Case, Sequence diagram and an activity diagram are used to specify what the system should do.

6.2 MODELING THE SYSTEM

Modeling a system is the process of abstracting and organizing significant features of how the system would look like. Modeling is the designing of the software applications before coding. Unified Modeling Language (UML) tools were used in modeling this system.

6.2.1 UML (UNIFIED MODELING LANGUAGE) MODELING

The Unified modeling language is an object-oriented system notation that provides a set of modeling conventions that is used to specify or describe a software system in terms of objects. The Unified Modeling Language (UML) has become an object modeling standard and adds a variety of techniques to the field of systems analysis and development hence its choice for this project.

In this project, the Use case diagram, Class diagram, Sequence diagram, Activity diagram, Collaboration diagram, Component diagram and State diagram will be used for system modeling.

4.3.1 USE CASE DIAGRAM FOR TIMETABLING USING GENETIC ALGORITHMS

Use case diagrams describe what the program does from the standpoint of an external observer. The emphasis of use case diagrams is on what the program does rather than how. They are used to show the interactions between users of the program and the program. A use case represents the several users called actors and the different ways in which they interact with the system.

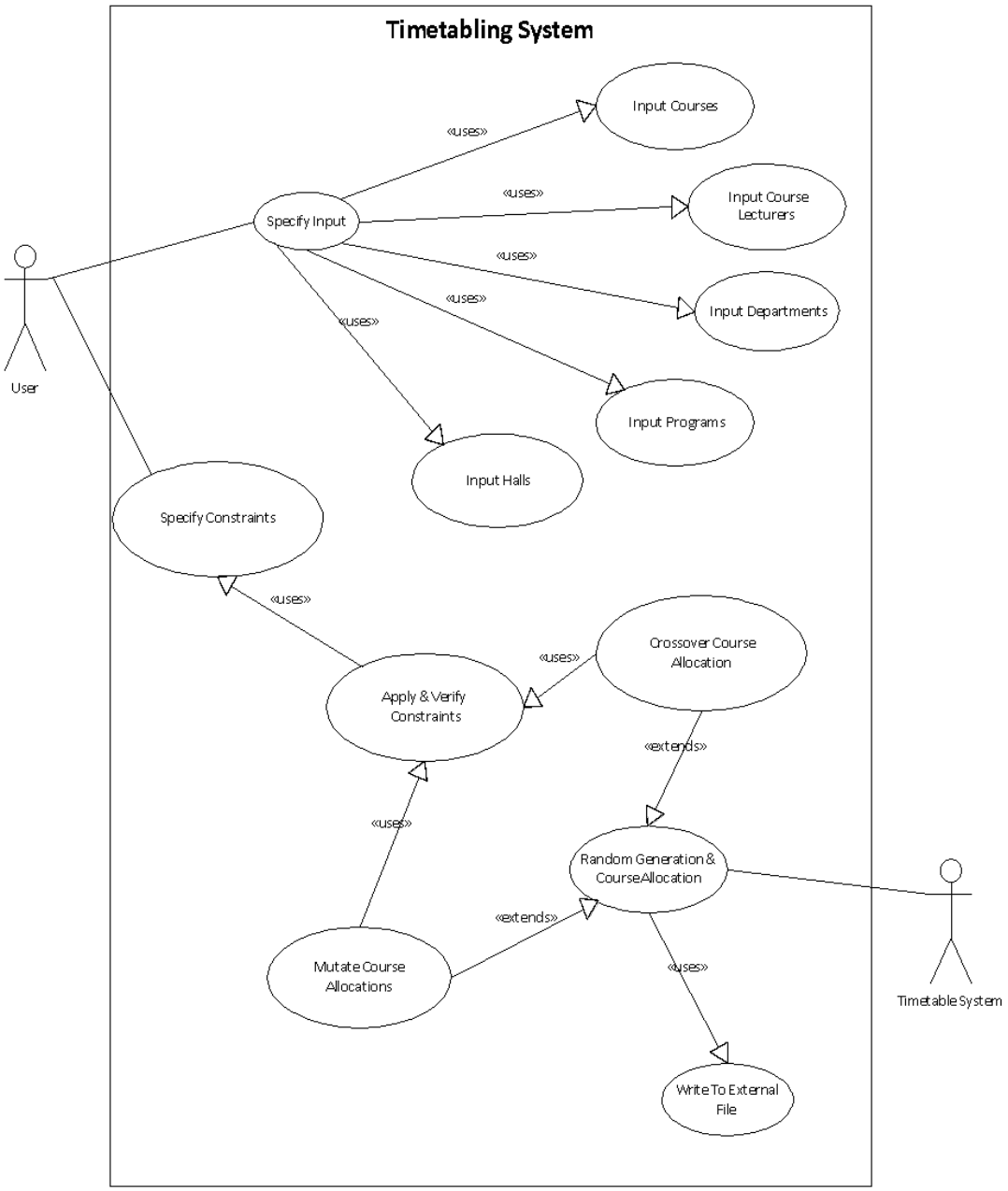
ACTORS

- User
- Timetable System

USE CASES

- Specify Input
- Specify Constraints
- Input Courses
- Input Course Lecturers
- Input Departments
- Input Programs
- Input Halls
- Apply and Verify Constraints
- Crossover Course Allocation
- Mutate Course Allocations
- Random Generation and Course Allocations
- Write to External File

Figure 4.1: Use Case Diagram to show the interaction between the timetabling system and user



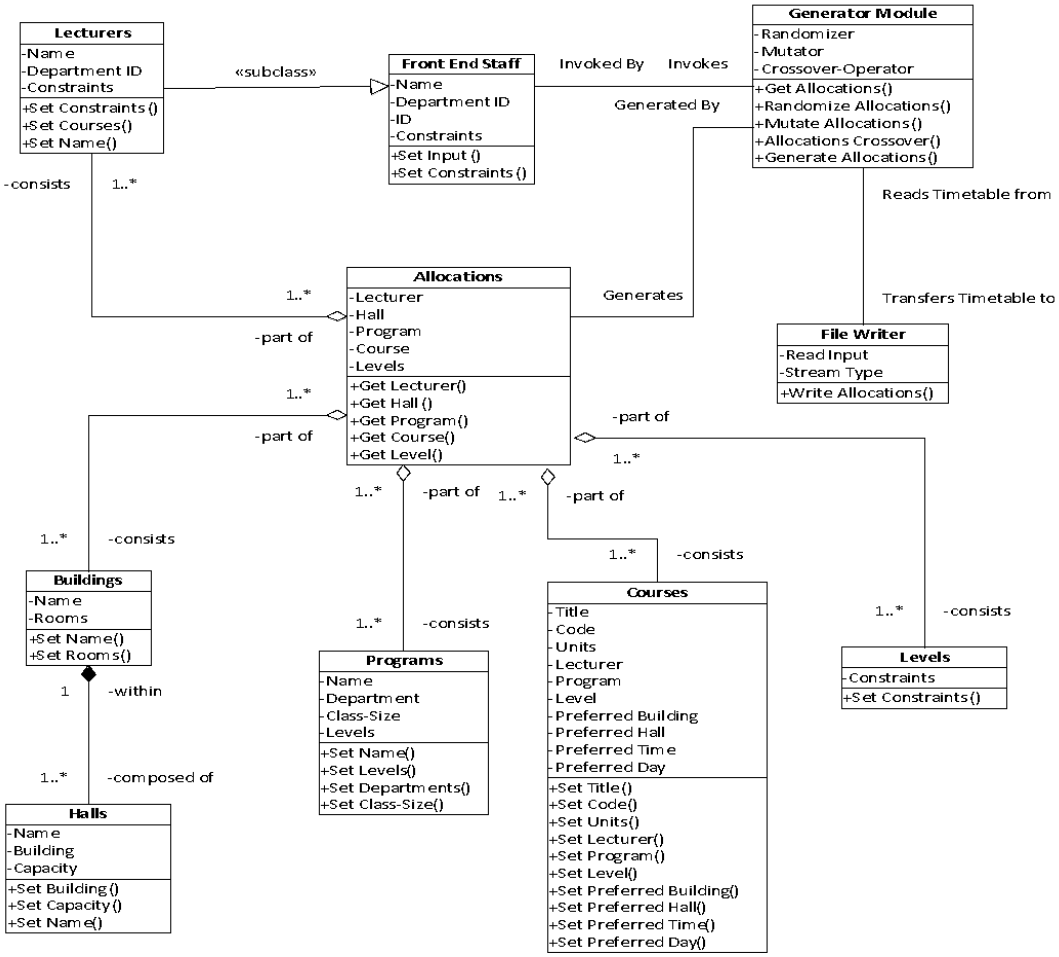
4.3.2 Class Diagram

A class diagram is an organization of related objects.
It provides an outline of a system by showing its categories and the relationships among them. Class diagrams solely show what interacts however not what happens throughout the interaction then they are static diagrams.

CLASSES

- Lecturers
- Buildings
- Halls
- Program
- Courses
- Levels
- Allocations
- Front End Staff
- Generator Module
- File Writer

Figure 4.2: Class Diagram to show the relationships between the different classes associated with the system



4.3.3 Sequence Diagram

A sequence graphically depicts how objects interact with each other via messages in the execution of a use case or operation. They illustrate how messages are sent and received between objects and the sequence of message transfer. It also details how operations are carried out according to the time of operation.

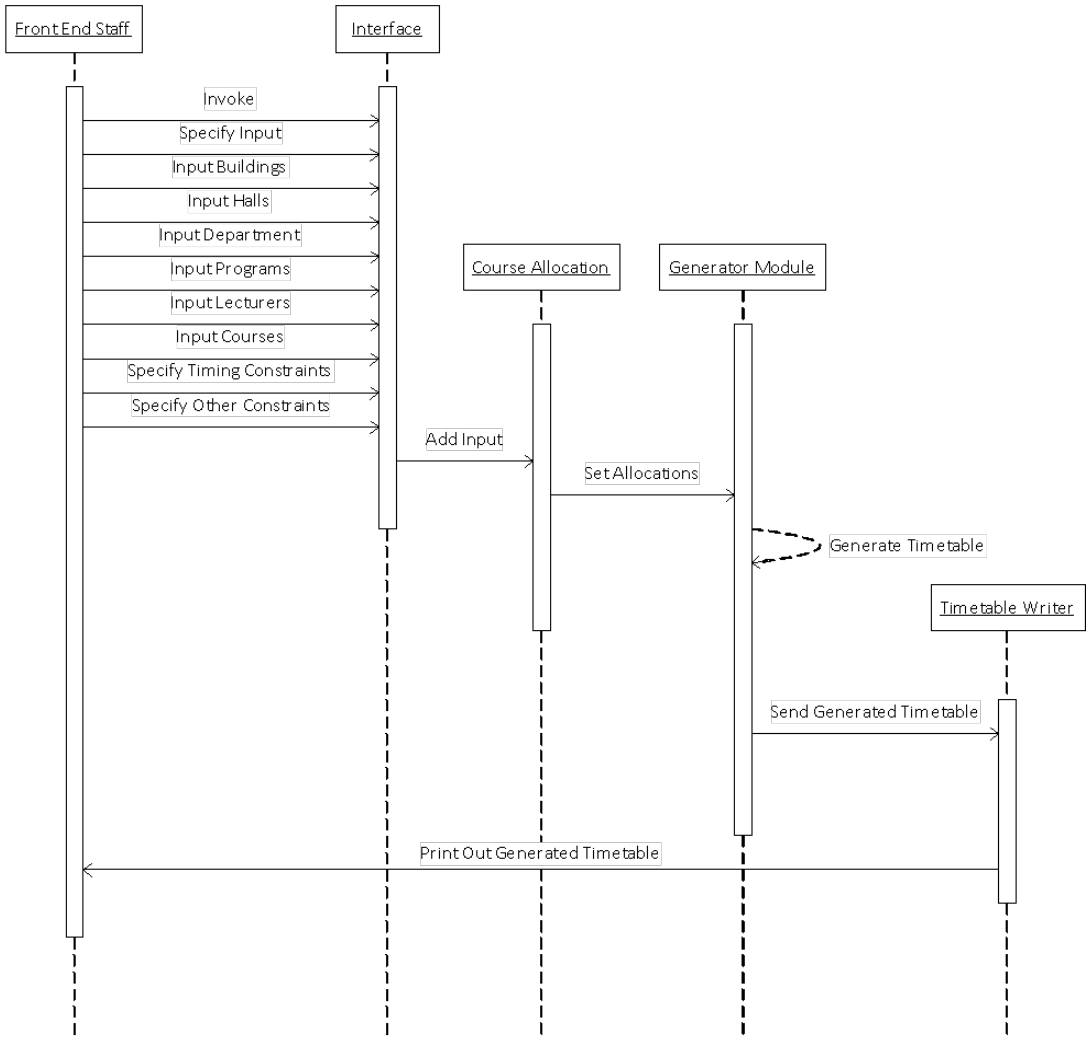
CLASSES

- Front End Staff
- Interface
- Course Allocation
- Generator Module
- Timetable Writer

MESSAGES

- Invoke
- Specify Input
- Input Buildings
- Input Halls
- Input Departments
- Input Programs
- Input Lecturers
- Input Courses
- Specify Timing Constraints
- Specify Other Constraints
- Add Input
- Set. allocations
- Generate Timetable
- Send Generated Timetable
- Print Out Generated Timetable

Figure 4.3.: Sequence Diagram to show how the different objects interact during the execution of the system



4.3.4 Activity Diagram

Activity diagrams graphically depict the sequential flow of activities of either a business process or a use case. They can also be used to model actions that will be performed when an operation is executed as well as the results of those actions. They focus on the flow of activities involved in a single process. The activity diagram shows however those activities depend upon each other.

SWIMLANES

- Front End User
- Interface
- Timetable Generator Module
- Writer

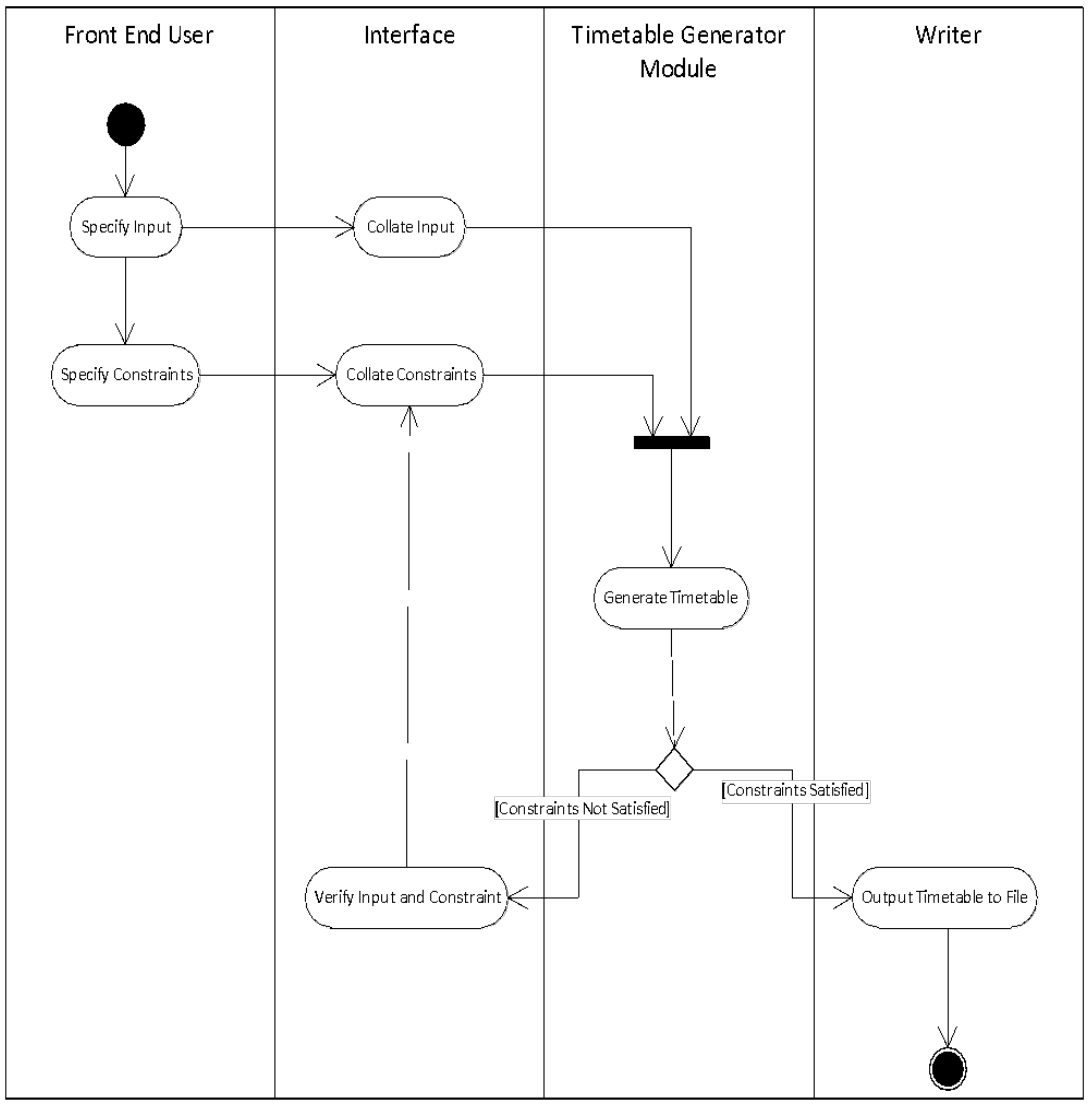
STATES

- Specify Input
- Specify Constraints
- Collate Input
- Collate Constraints
- Generate Timetable
- Verify Input and Constraints
- Output Timetable to File

GUARD EXPRESSIONS

- Constraints not Satisfied
- Constraints Satisfied

Figure 4.4: Activity Diagram to model the actions and the output of those actions when an operation is carried out in the system.



4.3.5 State Diagram

State diagrams are used to model the dynamic behavior of a selected object. They illustrate an object's life cycle i.e. the various states that an object can assume and the events that cause the object to move from one state to another.

STATES

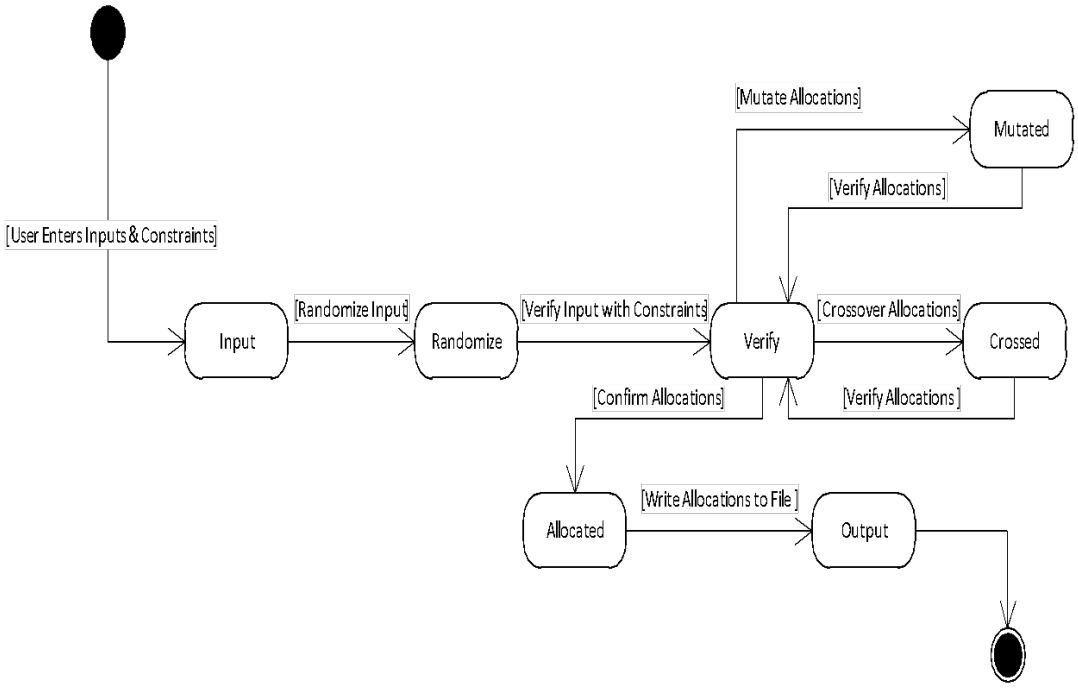
- Input
- Randomize
- Crossed
- Mutated
- Verify
- Group

- Output

TRANSITIONS

- User Enters Inputs & Constraints
- Randomize Input
- Verify Input with Constraints
- Crossover Allocations
- Verify Allocations
- Mutate Allocations
- Confirm Allocations
- Write Allocations to File

Figure 4.5: State Diagram to depict the different states of the system during its execution



4.3.6 Collaboration Diagram

Collaboration diagrams are similar to sequence diagrams but do not focus on the timing or sequence of messages.Instead they give the interaction between objects during a network format.

Figure 4.6: Collaboration Diagram showing the interaction between the objects in the system

4.3.7 Component Diagram

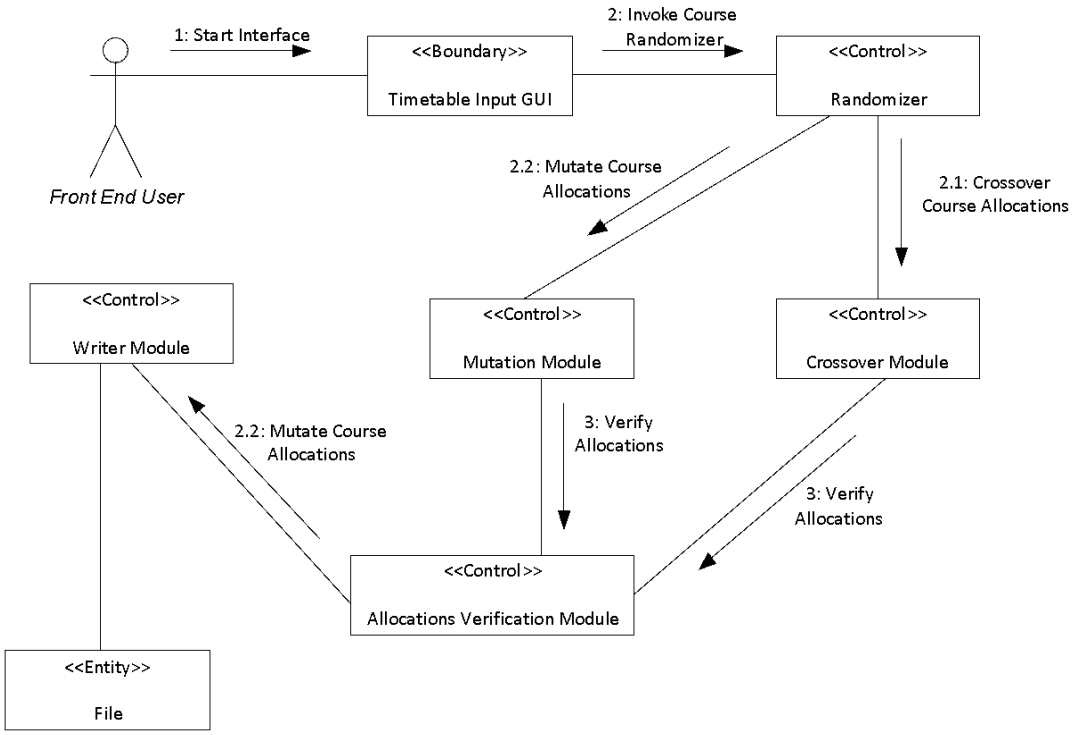
Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executable.

COMPONENTS

- Input Module

- Randomizer
- Crossover Operator
- Mutation Operator
- Verifier
- File Handler

Figure 4.7: Component Diagram to show the connection between the various software modules in the system



REFERENCES

References

[1] Wren A. "Scheduling, timetabling and rostering-a special relationship?', Practice and Theory of Automated Timetabling". In: (1996), pp. 46–75.

[2] Mccollum B and Ireland N. "University timetabling: Bridging the gap between research and practice". In: *Practice and Theory of Automated Timetabling Conference* (2006), p. 35.

[3] Gunzenhauser R and Junginger W. "About a method for creating, schedule of school hours with the help of a central pension scheme". In: (1964), p. 100.

[4] De D. "An introduction to timetabling". In: *European Journal of the Operational Research society* 19 (1985), pp. 151–162.

[5] Schofer E and Meyer J. “The worldwide expansion of higher education in the twentieth century”. In: *American sociological review* 70 (6 2005), pp. 898–920.

[6] Babaei H, Karimpour Hadidi J, and A. “A survey of approaches for the university course timetabling problem”. In: (2015), pp. 43–59.

[7] Burke E and Petrovic S. “Recent research directions in automated timetabling”. In: *European Journal of Operational Research* 140 (2 2002), pp. 266–280.

[8] Corne D, Ross P, and Fang H. “Evolutionary timetabling: Practice, prospects and work in progress”. In: *UK planning and Scheduling SIG Workshop* (1994).

[9] Costa D and Hertz A. “Ants can color graphs”. In: *Journal of the operational research society* 48 (3 1997), pp. 295–305.

[10] Burke E, Kendall G, and Soubeiga E. “A tabu-search hyperheuristic for timetabling and rostering”. In: *Journal of Heuristics* 9 (6 2003), pp. 451–470.

[11] Burke E, Kendall G, and Soubeiga E. “Multi-objective hyperheuristic approaches for space allocation and timetabling”. In: *Metaheuristics: Progress as Real Problem Solvers* (2005), pp. 129–158.

[12] Amaral J et al. “About Computing Science Research Methodology”. In: *Webdocs.cs.ualberta.ca. Available* (2000).

[13] Hibbs C, Jewett S, and Sullivan M. “The art of lean software development”. In: (2009).