**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores**

# Automated University Timetable Creation with Genetic Algorithm and Local Search

ANDRÉ GONÇALVES

(Licenciado em Engenharia Informática e de Computadores)

Trabalho de projeto realizado para obtenção do grau
de Mestre em Engenharia Informática e de Computadores

(Relatório Preliminar)

Orientadores:
Mestre Artur Jorge Ferreira
Mestre Nuno Miguel da Costa de Sousa Leite

**26 Março de 2015**

**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores**

# Automated University Timetable Creation with Genetic Algorithm and Local Search

Trabalho de projeto realizado para obtenção do grau de Mestre em Engenharia Informática e de Computadores

(Relatório Preliminar)

AUTOR:

(André Gonçalves)

ORIENTADORES:

(Artur Jorge Ferreira)

(Nuno Miguel da Costa de Sousa Leite)

**26 de Março de 2015**

## Acknowledgments

I wish to express my apreciation to Artur Ferreira and Nuno Leite, for their continous support and orientiation throughout this project. Whithout their input and valuable advice, I would not be able to finish this project.

I would also like to thank my family, for all the support and motivation that they provided throughout my life.

I would also like to thank the Instituto Superior de Engenharia de Lisboa, for allowing me to complete this academic goal of mine.

## Abstract

Every year, universities around the world, at the beginning of each term face the problem of creating a schedule or timetable for the subjects lectured. These timetables must satisfy the needs of a university and, at the same time, attempt to satisfy as many of the wishes and requirements of both the university staff and students.

Historically the timetabling problem was considered a subject of the field of operational research, but in the last decades, it has also become a subject of research in the field of Artificial Intelligence, more specifically the application of metaheuristic-algorithms to solve these type of NP-complete problems (e.g. constraint satisfaction, tabu search and genetic algorithms are a few of the techniques employed).

The goal of this project is to implement a solver program for the problem of Curriculum Based Course Timetabling as it was formulated in the Track 3 of the second ITC competition that took place in 2007.

This solver program will be a hybrid system consisting in the application of a genetic algorithm to obtain feasible solutions in an early stage. The quality of these solutions further improved by the usage of a local strategy, namely a Simulated Annealing algorithm.



### Keywords

Timetabling; Curriculum-Based Course Timetabling; Metaheuristics; Evolutionary Algorithms; Simulated Annealing; Local Search; Java

# Contents

# List of Figures

# 1

# Introduction

Every year, universities around the world, at the beginning of each term face the problem of creating a schedule or timetable for the subjects lectured. These timetables must satisfy the needs of a university and at the same time, attempt to satisfy as many of the wishes and requirements of both the university staff and students.

Elaboration of these timetables done manually by administrative staff of the university is a time-consuming task which is both difficult and prone to error.

Historically the timetabling problem was considered a subject of the field of operational research, but in the last decades, it has also become a subject of research in the field of Artificial Intelligence, more specifically the application of *m*eta-heuristic algorithms to solve these type of NP-hard problems [13], [15] (e.g. *constraint satisfaction*, *tabu search* and *genetic algorithms* are a few of the techniques employed).

Although there is a lot of literature published regarding proposed solutions that are effective at solving these problems, after reading through some of these proposals, we infer that the formulation of the problem to solve is different in each proposal and usually adjusted to the rules and specific requirements of a given institution and the results obtained in the implementations of these solutions reflect this tight coupling.

The International Timetabling Competition (ITC) [21] was established to stimulate interest in the general area of educational timetabling while providing researchers with models of the problems faced which incorporate an increased number of real world constraints.

This competition was divided in three tracks:

- Track 1 - Examination Timetabling
- Track 2 - Post Enrollment Based Course Timetabling (PE-CTT)
- Track 3 - Curriculum Based Course Timetabling (CB-CTT)

For each track, it was provided the common generic formulation of the problem to solve, which in this case is the creation of feasible timetables and the hard and soft constraints that a proposed solution must observe. In order to compare the results obtained with the

proposed solution solvers, common datasets, representing real-word instances that represent the problem to solve, were provided, in order to give researchers a common benchmark.

The goal of this project is to implement a solver program for the problem of Curriculum Based Course Timetabling as it was formulated in the Track 3 [22] of the second ITC competition that took place in 2007.

For this solver program, we propose the implementation of a Hybrid system, in which in the first stage, a genetic algorithm is used to obtain a set of feasible solutions. In a second stage, the results obtained are processed by a local search algorithm – i.e. Simulated Annealing – in an attempt to further improve the solutions, in terms of Soft Constraints.

The resulting implementation will be tested with the ITC 2007 datasets, which in the case of track 3 corresponds to real datasets that were obtained from university of Udine in Italy, and under some of the rules proposed in this competition. The results obtained will be compared both between the implemented s and the results of the solutions proposed by the winner of the ITC 2007 Track 3 [**?**], [**?**], in order to evaluate if the implemented solver can provide both better quality solutions and/or can be faster at obtaining these solutions.

This document is divided in the following Subsections:

1. Curriculum-Based Course Timetabling Problem: In this section, we present the problem of the Curriculum-Based Course Timetabling.
2. State of the Art: In this section we present an overview of several proposed approaches that deal with the timetabling problem.
3. Proposed solution: In this section we present the selected algorithms that will later be implemented. We propose a hybrid solution that employs a Genetic Algorithm to obtain a set of feasible solutions that will be further improved through the usage of local search local search algorithms, more specifically the Simulated Annealing algorithm.
4. Project Development Planning: In this section, we present the overall project plan.
5. Future Work: In this section, we present some project topics that should be considered in the future.

# 2

# Curriculum-Based Course Timetabling problem

In this section we present the formulation of the CB-CTT problem. The specific problem of Curriculum-Based Course Timetabling, as proposed by Di Gaspero et al. [**?**], is an optimization problem, which consists of the weekly scheduling of the lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula published by the University and not on the basis of enrolment data.

The problem consists of the following entities:

- Days, Timeslots, and Periods: Given a number of teaching days in the week (typically 5 or 6) and considering that each day is split in a fixed number of timeslots, which are equal for all days. A period is considered to be a pair composed by a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.
- Courses and Teachers: Each course consists of a fixed number of lectures to be scheduled in distinct periods, is attended by given number of students, and is taught by a teacher. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.
- Rooms: Each room has a capacity, which corresponds to the number of available seats. All rooms are equally suitable for all courses.
- Curricula: A curriculum is a group of courses such that any pair of courses in the group has students in common. Based on curricula, we have the conflicts between courses and other soft constraints.

A feasible solution of the problem will correspond to an assignment of a period (day and timeslot) and a room to all lectures of each course and must conform to the following Hard constraints:

- Lectures: All lectures of a course must be scheduled and they must be assigned to distinct periods.
- Room Occupancy: Two lectures cannot take place in the same room and in the same period.

- Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must be all scheduled in different periods.
- Availabilities: If the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period.

Additionally, the following Soft constraints should be met as well as possible:

- Room Capacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures.
- Minimum Working Days: The lectures of each course must be spread into a minimum number of days.
- Curriculum Compactness: Lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day.
- Room Stability: All lectures of a course should be given in the same room.

A solution that conforms to these Soft Constraints is one that exhibits desired structural properties like coherent daily time slots for lectures of the same curriculum, etc.

# 3

# State of the Art

As mentioned previously, the automated construction of timetables is a research field for which extensive literature exists. Carter [9] proposed a categorization of four types of methods that can be used to solve these type of problems: sequential methods, cluster methods, constraint-based methods and generalized search. Petrovic et al. [25] proposed additional methods, such as hybrid evolutionary algorithms, meta-heuristics, multi-criteria approaches, hyper-heuristics and adaptive approaches.

Socha et al. [28] proposed an approach to tackle the enrollment-based course timetabling problem that applied local search and ant colony optimization strategies. Rossi-Doria et al. [26] proposed the usage of evolutionary algorithms to solve the timetabling problem and presented a comparison between this method and several meta-heuristic methods. Burke et al. [5] applied a tabu-based hyper-heuristic to the university course timetabling problem. Burke et al. [7] proposed the use of tabu search along with a graph based hyper-heuristic. Abdullah et al. [1] developed a variable neighborhood search approach in conjunction with a tabu list.
McMullan [23] proposed the application of an extended great deluge algorithm and Landa-Silva et al. [19] presented another variation of the great deluge algorithm, the non-linear great deluge.
The combination of genetic algorithm and local search has been previously employed by Abdullah et al. [2].

Müller [24] presented a constraint-based solver to the curriculum-based course timetabling problems in the 2nd International Timetabling Competition, ITC2007 (Track 1 and Track 3) and achieved the first place in this competition. Lü et al. [17] applied a hybrid heuristic algorithm called Adaptive Tabu Search (ATS) to the same instances.
Clark et al. [12] applied repair-based heuristic search on Track 3 datasets in the ITC2007 timetabling competition. Geiger [16] applied a stochastic neighborhood method based on threshold acceptance criteria to overcome the local optima to the same instances. Atsuta et al. [3] applied the constraint satisfaction problem (CSP) which implemented a hybridization of tabu search and iterated local search algorithms to handle weighted constraints.
De Cesco et al. [14] applied a dynamic tabu search to curriculum-based course timetabling, a short term tabu exclusion with variable size tabu length, with dynamic weight adjustment for hard and soft constraints.

Lach et al. [18] applied an integer programming method in order to create a solver to handle timetabling problem instances.
Burke et al. [6] proposed a solver based on a hybrid meta-heuristic to tackle scheduling problems.

Surveys that document these and additional approaches that handle these type of problems, can be consulted in works such as Carter et al. [10], Burke et al. [8], Carter [9], and Schaerf [27].

# 4

# Proposed Solution

## 4.1 Overview

After researching through the various proposals for solving different timetabling problems (e.g. [4], [11]) it became clear that although genetic algorithms (GA) are an appropriate method to solve this type problem, it also became clear that it's not efficient in terms of time and computation effort needed to attain feasible solutions, when compared with other strategies such as tabu search or simulated annealing. In fact, the work submitted by Chiarandini et al. [11] proposes that the usage of meta-heuristic algorithms such as GA and Ant Colony Optimization (ACO) alone should be hybridized with local search procedures in order to achieve good results.

With this in mind and since the ITC2007 rules [20] for evaluating the algorithms indicate that the solutions must be produced within a time limit, it became apparent that it would be more appropriate to explore a hybrid approach, in which:

- A GA is used to create the initial feasible solutions
- A local search algorithm – i.e. Simulated Annealing - is applied over the results produced by the GA in order to improve the quality of solutions.

The general overview is depicted in Figure 4.1.

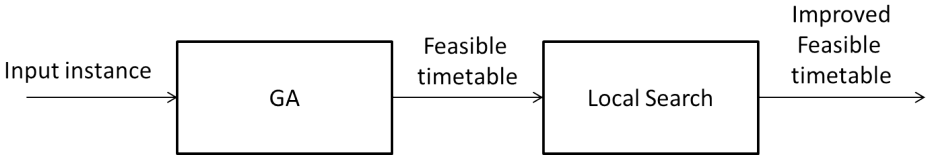


Figure 4.1: Overview of the proposed approach.

## 4.2 First Phase - Genetic (Evolutionary) Algorithm

In the first stage, for the proposed genetic algorithm, only the hard constraints are considered, meaning that the fitness function used in this stage, will take into account only the violations of hard constraints, in an attempt to produce feasible timetables where all of these constraints are satisfied.
It's expected that in the end of this first stage, the algorithm will have produced at least one feasible timetable and it will initiate the second stage if and only if a feasible timetable is found.

Figure 4.2 ilustrates the steps that will be executed by the algorithm.

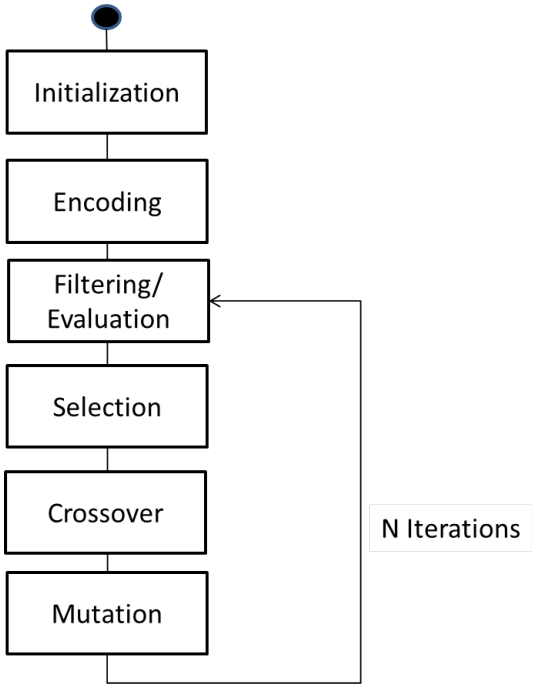


Figure 4.2: Overview of the steps of the proposed genetic algorithm.

The following subtopics present an overview of the implementation considerations that will be taken into account in each one of these steps.

### 4.2.1 Initialization

The algorithm in the initialization step will create a population of chromosomes. In this stage we propose to use a sequential greedy heuristic as proposed by Lü et al. [17] starting from an empty timetable, where assignments are constructed by inserting one appropriate lecture into the timetable at each time. At each step, two distinct operations are carried

out: one is to select a still unassigned lecture of a course and the other is to determine a period-room pair for this lecture.
The algorithm does not place a course lecture in a timeslot which is not available for that course, and will attempt to satisfy this constraint satisfied in the next generations for as long as possible. After the completion of the generation of the population, the resulting timetables will be encoded as chromosomes.

### 4.2.2 Encoding

The proposal for encoding the chromosomes that represent the timetables consists of using a 1 dimensional array of binary values, with a structure such as the one in Figure 4.3.

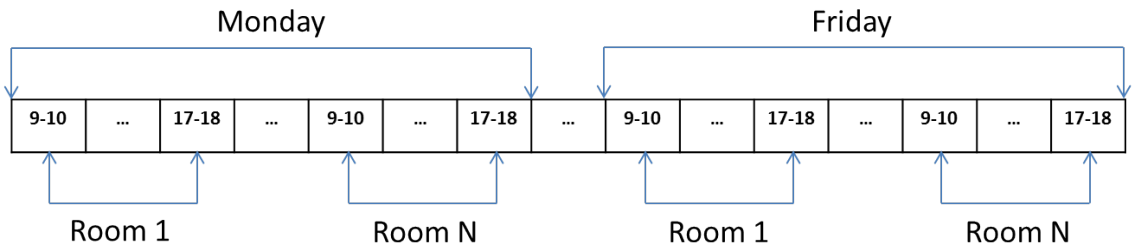


Figure 4.3: Chromosome representation.

Each cell in the array represents a timeslot. A group of X adjacent timeslots represents one day of lectures that can be assigned in a single room. A group of Y adjacent room-timeslots represents the one day of lectures that can be allocated on that given day. Value 0 represents that the timeslot is available for allocation, 1 represents that it is already alocated.
The information regarding the lecturer, curriculum, etc. will be maintained on an associative array that will keep the information regarding the current allocated position in this array.

### 4.2.3 Evaluation

A fitness value is assigned to each timetable based on the number of violations of hard and soft constraints detected. In each generation the chromosomes are evaluated and then the chromosome pool is sorted. The evaluation process employs a fitness function to calculate the fitness of each timetable.

### 4.2.4 Selection

In the selection step, the fittest individual will be kept unchanged. The remaining individuals will subjected to a tournament selection method. In this method, K chromosomes are chosen

from the population, and the ones with a greater fitness value will be selected to perform the crossover, and thus generate new offspring that will replace the K least fit individuals.

### 4.2.5 Crossover

During the implementation phase, experiments with different types of crossover operators – e.g. One-Point Crossover, Two-Point Crossover, N-Points Crossover – will be conducted in order to evaluate which one of these is more adequate to generate the offspring.

### 4.2.6 Mutation

The mutation function randomly exchanges the values in every pair in a random number of location pairs of a timetable. The mutation rate is not fixed. In each generation the maximum and average fitness values will be calculated. If the two numbers are close enough to each other, the mutation rate value will be increased to avoid any chance of an early convergence.

### 4.2.7 Repair Function

It is expected that the application of the crossover operator will produce non feasible solutions, possibly with worse fitness values than the K least fit individuals that were determined in the selection step. Therefore, instead of purging the K least fit individuals immediately after the crossover step, a new fitness evaluation will be performed over the new population along with the K least fit individuals. After this evaluation, the new K least fit individuals group will be purged, and a new iteration of the process will begin.

## 4.3 Second Phase - Simulated Annealing Algorithm

Simulated annealing (SA) has been widely used to solve combinatorial optimization problems. It accepts the new solution when the objective value, which is determined by an objective function, is lower than or equals to the current one. There is a probability to accept worse quality solution using a probability acceptance function which controls the acceptance of the new solution. The current temperature is iteratively reduced according to the cooling schedule with a given cooling rate in each iteration or level until its temperature reaches final minimum temperature which is close to zero.
When the SA temperature becomes very low, the SA will exhibit a behavior similar to a descent heuristic (i.e. accepts only an improving solution) and therefore the search might be trapped in local optimum. To limit this effect, reheating will be used, in order to attempt to restart the search in another point in the search space.
The proposed SA algorithm involves: neighborhood structure, temperature, cooling schedule, termination criteria.

- Case I: The minimum temperature is close to zero
- Case II: Number of iterations reaches the limit
- Case III: Timeout based on the ITC 2007 course timetabling track 3 stopping condition.

## 4.4 General System Architecture Overview

In this section we discuss the general system architecture that will be implemented. Figure 4.4 illustrates the proposed architecture.

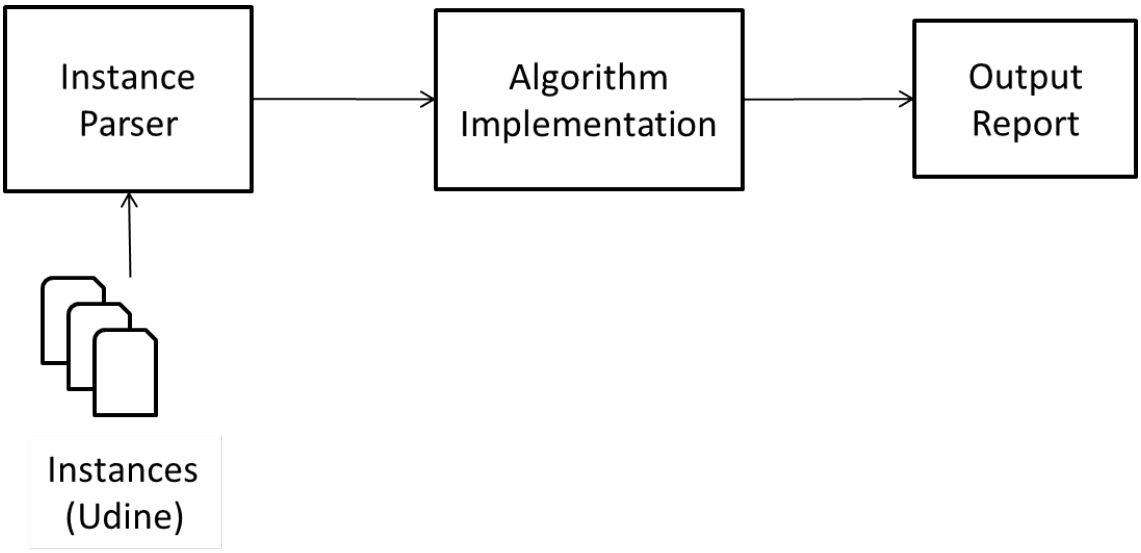


Figure 4.4: General System Architecture.

The instance parser is the component responsible for parsing the problem instances that were provided by the University of Udine for the ITC 2007 Track 3 competition. The instances will loaded in memory. After the instances are loaded, the implemented algorithms modules will be executed. After the completion of the execution, a report file will be generated with the information regarding the best solution and data regarding metrics (e.g. Time executed, CPU time, etc).
The system will be implemented as a normal Java console application. The motivation regarding the usage of Java for the implementation is the availability of third party components that can be useful in the data manipulation and visualization, which can benefit the analysis of the results.
As a general requirement, the application should be able to obtain configuration data that can be used to perform tuning of parameters used in the solver algorithms (e.g. the parameters used in the cooling schedule of simulated annealing algorithm).

# 5

# Project Planning

This project is comprised two phases. In the first phase, which is currently completed, consisted on the research of the timetabling problem, with emphases on the ITC 2007 Track 3 Curriculum-Based Timetable problem, and several meta-heuristic algorithms that are used to create automated solvers. In this phase, we also selected the algorithms that will be implemented and defined the way in which the results obtained with each implementation will be analyzed.
Figure **??** presents the allocation of time to the most relevant tasks were executed in the first phase of this project.

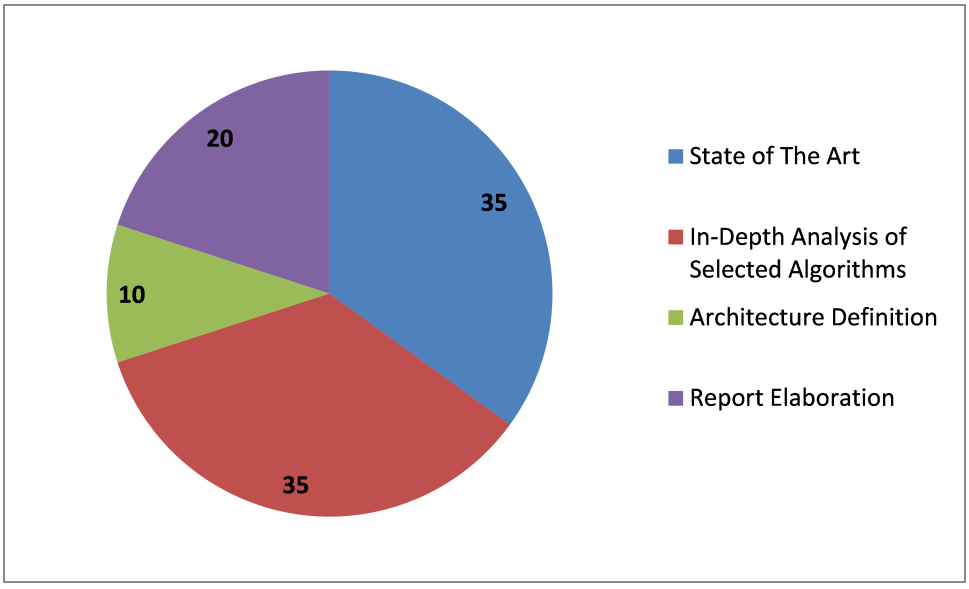


Figure 5.1: Allocation of time to the tasks executed in the first phase.

The following GANT diagram (Figure **??** ) illustrates a detailed planning for the whole year. It includes all the relevant tasks related to the first and second phases of the project devel-

opment. The second phase (and pending) tasks are represented in blue.

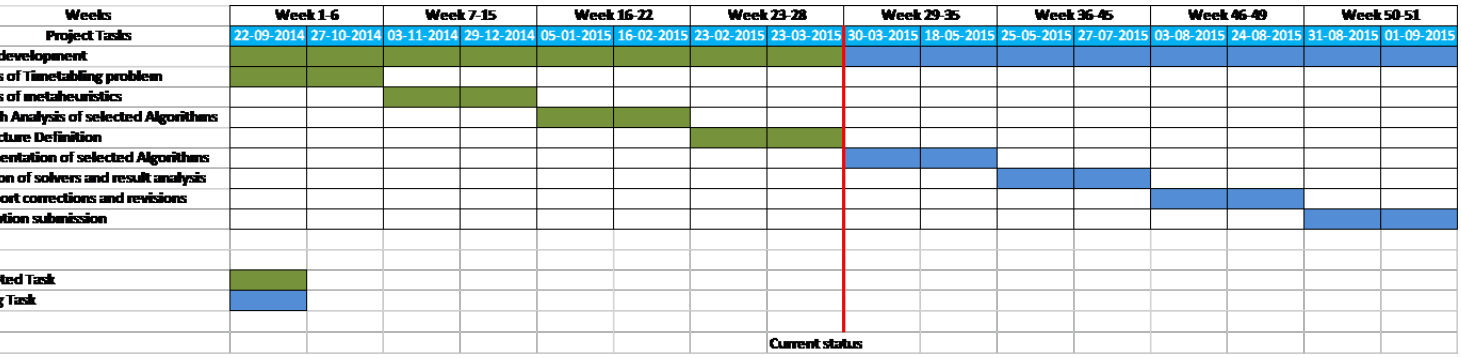


Figure 5.2: GANTT diagram detailing the project plan.

Some changes to the proposed plan may occur during the project development, influencing the duration and final proposed dates for some tasks may. The time distribution and amount of work to be accomplished between the winter and summer periods is not equally distributed. This is because of the courses taken by the author during the winter semester in parallel with the project development, which consumed a lot of time of the winter period.

# 6

# Future Work

In the following project phase, we will initiate the implementation of the algorithms. Upon completion of the implementation, we will initiate the execution of the algorithms over the provided instances and afterwards will analyze the results obtained and compare them with the ITC 2007 winning implementation results.

# References


1. S. Abdullah, E.K. Burke, and B. McCollum. An investigation of variable neighborhood search for university course timetabling. In *The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, page 413–427, 2005.
2. S. Abdullah and H. Turabieh. Generating university course timetable using genetic algorithm and local search. In *Proceeding of the 3rd International Conference on Hybrid Information Technology*, page 254–260, 2008.
3. M. Atsuta, K. Nonobe, and T. Ibaraki. Itc-2007 track2 - an approach using general csp solver, 2007.
4. A.O. Bajeh and K.O. Abolarinwa. Optimization: A comparative study of genetic and tabu search algorithms. *International Journal of Computer Applications*, 31(5):43–48, October 2001.
5. E.K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.
6. E.K. Burke, J. Marecek, A. Parkes, and H. Rudová. Decomposition, reformulation, and diving in university course timetabling. *Comput. Oper. Res*, 37(3):582–597, 2009.
7. E.K. Burke, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.
8. E.K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.
9. M.W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193–202, 1986.
10. M.W. Carter and G Laporte. Recent developments in practical examination timetabling. *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 3–21, 1996.
11. M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9(5):403–432, October 2006.
12. M. Clark, M. Henz, and B. Love. Quikfix. a repair-based timetable solver. In *Proceedings of the Seventh PATAT Conference*, 2008.
13. T.B. Cooper and J.H. Kingston. The complexity of timetable construction problems. *The Practice and Theory of Automated Timetabling: Selected Papers from the First International. Lecture Notes in Computer Science, Springer-Verlag Berlin,*, 1153:283–295, 1996.
14. F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results. In *Proceedings of the Seventh PATAT Conference*, 2008.
15. S. Even, A. Itai, and A. Shamir. On the complexity of timetabling and multicommodity flow problems. *SIAM Journal of Computation*, 5(4):691–703, 1976.
16. M.J. Geiger. An application of the threshold accepting metaheuristic for curriculum based course timetabling. In *Proceedings of the Seventh PATAT Conference*, 2008.

17. Z. Lü and J.-K. Hao. Adaptive tabu search for course timetabling. *European Journal of Operational Research.*, 200(1):235–244, 2010.
18. G. Lach and M.E. Lübbecke. Curriculum based course timetabling: Optimal solutions to the udine benchmark instances. In *Proceedings of the Seventh PATAT Conference*, 2008.
19. D. Landa-Silva and J.H. Obit. Great deluge with non-linear decay rate for solving course timetabling problem. In *The Fourth International IEEE Conference on Intelligent Systems*, pages 11–18, 2008. Varna, Bulgaria.
20. B. McCollum. Itc 2007 – competition rules. Online, 2007.
21. B. McCollum. Itc 2007 site. Online, 2010.
22. B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. Parkes, L. Di Gaspero, R. Qu, and E.K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal of Computing*, 22(1):120–130, 2010.
23. P. McMullan. An extended implementation of the great deluge algorithm for course timetabling. *Computational Science—ICCS, Part I. LNCS*, 4487:538–545, 2007. Springer, Berlin.
24. T. Müller. Itc2007: Solver description. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 2007.
25. S. Petrovic and E.K. Burke. University timetabling. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton, Chap. 45*, 2004.
26. O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L.M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, and T. Stützle. A comparison of the performance of different meta-heuristics on the timetabling problem. *Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science*, 2740:329–351, 2003.
27. A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
28. K. Socha, J. Knowles, and M. Sampels. A max-min ant system for the university course timetabling problem. *The Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002). LNCS*, 2463:1–13, 2002. Springer, Berlin.