UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

An evolutionary algorithm with simulated annealing-based mutation for automated timetabling problem

Trabajo de integración curricular presentado como requisito para la obtención del título de
Ingeniero en Tecnologías de la Información

Autor:

Jonathan David Freire Valencia

Tutor:

Ph.D. Cuenca Lucero Fredy Enrique

Urcuquí, Enero del 2022

Urcuquí, 23 de diciembre de 2021

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2021-00042-AD

A los 23 días del mes de diciembre de 2021, a las 13:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Miembro No Tutor	Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO
Tutor	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.

El(la) señor(ita) estudiante **FREIRE VALENCIA, JONATHAN DAVID**, con cédula de identidad No. **1805102074**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **AN EVOLUTIONARY ALGORITHM WITH SIMULATED ANNEALING-BASED MUTATION FOR AUTOMATED TIMETABLING PROBLEM** , previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Presidente Tribunal De Defensa	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.	10,0
Miembro Tribunal De Defensa	Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO	10,0
Tutor	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.	9,5

Lo que da un promedio de: **9.8 (Nueve punto Ocho)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

FREIRE VALENCIA, JONATHAN DAVID
Estudiante

Firmado electrónicamente por:
JONATHAN DAVID FREIRE VALENCIA

Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Presidente Tribunal de Defensa

Firmado electrónicamente por:
CRISTHIAN RENE IZA PAREDES

Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Tutor

FREDY ENRIQUE CUENCA LUCERO
Firmado digitalmente por FREDY ENRIQUE CUENCA LUCERO
Fecha: 2021.12.23 21:40:59 -05'00'

GUSTAVO ADOLFO COLMENARES PACHECO

Digitally signed by GUSTAVO ADOLFO COLMENARES PACHECO
DN: C=EC, O=BANCO CENTRAL DEL ECUADOR, OU=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, L=QUITO, SERIALNUMBER=0000316503 + CN=GUSTAVO ADOLFO COLMENARES PACHECO
Reason: I am the author of this document
Location: your signing location here
Date: 2021.12.23 15:58:31-05'00'
Foxit PDF Editor Version: 11.1.0

Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO
Miembro No Tutor

DAYSY MARGARITA MEDINA BRITO

Firmado digitalmente por DAYSY MARGARITA MEDINA BRITO
Fecha: 2021.12.23 14:44:57 -05'00'

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Jonathan David Freire Valencia**, con cédula de identidad **1805102074**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Enero del 2022.

Firmado electrónicamente por:
JONATHAN DAVID FREIRE VALENCIA

Jonathan David Freire Valencia
CI: 1805102074

Autorización de publicación

Yo, **Jonathan David Freire Valencia**, con cédula de identidad **1805102074**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Enero del 2022.

Firmado electrónicamente por:
JONATHAN DAVID FREIRE VALENCIA

Jonathan David Freire Valencia
CI: 1805102074

Dedication

"To my family, whom I love deeply. To the Yachay Tech community, with whom I have learned so much and now have eternal gratitude and appreciation. "

Acknowledgments

I would like to thank my parents, to whom I owe everything, for their effort and support to allow me to study up to this point. To my brothers for being part of my life. To the rest of my family, who have always supported me to get ahead in one way or another.

My thanks also to those who were part of my life at Yachay Tech University. To my professors, not only for their teaching but for constantly motivating us to improve. To my advisor Fredy Cuenca for all the knowledge he has given me throughout my career and for permanently resolving and being attentive to my concerns about this research. To the great friends that I made at the University, especially Joha, Daya, Cami, Jenny, Jordan, and Gabo, for always give me their support and from whom I learned a lot.

Abstract

Nowadays, solving the University course timetabling problem has become necessary to improve efficiency when assigning schedules and handle large amounts of data about students, teachers, classrooms, etc. We present a method to solve this problem by using an evolutionary algorithm with simulated annealing-based mutation. The problem consists of assigning students and events (classes or laboratories) in a schedule with timeslots and classrooms, subject to hard and soft constraints. The algorithm developed in this work consists of two phases. In the first phase, we will assign the most significant number of events that meet the hard constraints using different heuristics. Later, in the second phase, the evolutionary algorithm with simulated annealing-based mutation was implemented. The purpose is to try to meet the hard constraints that could not be fulfilled in the previous phase and to fulfill the most significant number of non-mandatory conditions. In the end, it is expected to have a schedule with all events assigned, fulfilling all the hard constraints and as many soft constraints as possible.

Keywords: Evolutionary algorithm, heuristic, university course timetabling problem, hard constraints, soft constraints.

Resumen

En la actualidad el poder resolver el problema de asignación de cursos en una Universidad se ha vuelto muy necesario para mejorar la eficiencia a la hora de asignar horarios y manejar grandes cantidades de datos sobre estudiantes, maestros, aulas, etc. En este trabajo se presentar un método para resolver este problema mediante el uso de un algoritmo evolutivo con una mutación basada en el algoritmo de recocido simulado. El problema consiste en asignar un conjunto de estudiantes y eventos (clases o laboratorios) en un horario tomando en cuenta las aulas disponibles, sujeto a restricciones obligatorias y restricciones no-obligatorias. El algoritmo desarrollado en este trabajo consta de dos fases. En la primera fase se intenta asignar la mayor cantidad de eventos que cumplan con las restricciones obligatorias mediante el uso de distintas heurísticas. Posteriormente en la segunda fase se implementó el algoritmo evolutivo con una mutación basada en el algoritmo de recocido simulado, cuya finalidad es intentar cumplir las restricciones obligatorias que no se hayan podido cumplir en la fase anterior y cumplir la mayor cantidad de restricciones no-obligatorias. Al final se espera tener un horario con todos los eventos asignados cumpliendo todas las restricciones obligatorias y la mayor cantidad de restricciones no-obligatorias posibles.

Palabras Clave: Algoritmo evolutivo, recocido simulado, heurísticas, problema de asignación de cursos de universidad, restricciones obligatorias, restricciones no-obligatorias.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In a world where planning is part of a person's daily life and vital for the correct management of companies and institutions, it is impossible not to come across with some activity during the day in which timetabling is not involved. Airline, metro, train, and ferry companies, to name just a few examples, offer precise timetables to their customers so that they can plan their trips and buy their tickets at their convenience. Although it goes unnoticed by people, the scheduling of most timetables is a laborious process to the point that it is unthinkable that today many of these schedules can be developed in a short time without the use of a computer.

Universities are no exception in the need of scheduling. As shown in Figure 1.1, there are mainly two university-related timetabling problems for which various solutions have been postulated over the years. The first problem is the university examination timetabling problem (UETTP), which, as its name suggests, consists of creating a weekly schedule for programming a set of midterm/final exams. This work focuses on the second type of problem, which is called the university course timetabling (UCTTP), which consists of allocating a set of events such as lectures, students, teachers, and features in predefined timeslots and rooms, where a set of constraints must be satisfied [3]. Two or three times a year, universities must publish semester or summer study timetables. It is necessary to place students and teachers in their respective lectures, respecting restrictions such as guaranteeing classrooms' availability and avoiding the clash of schedules for both students and professors.

Creating a university timetable can be solved manually by trial and error in the case of small universities. Nevertheless, this process could take weeks to be completed. On the other side, trying to solve the problem manually is unthinkable in most universities that can host thousands of students in a single academic term. When using a computer, it is necessary to have an algorithm capable of creating a timetable in the shortest possible amount of time, which meets all the university's restrictions. The implementation and evaluation of such an algorithm are the core of this thesis.

To create an efficient university timetable, the algorithm must fulfill two types of constraints. Hard constraints must be satisfied in their entirety, e.g., "A student cannot attend two or more lectures simultaneously." On the other hand, there are soft constraints that do not necessarily have to be met entirely; however, it is desirable to fulfill them as much as possible [4], e.g., "Avoiding students having to attend three or more meetings in successive timeslots."

There are several techniques for the resolution of UCTTP. The algorithms used to solve this problem can be divided into mathematical models, heuristic algorithms, and multi-criteria techniques. Mathematical models search for a (sub)optimal timetable employing a deterministic search. Some examples of these models include Graph Coloring Algorithms, Integer or Linear Programming methods, and Constraint Satisfaction Programming; On the other hand, heuristic algorithms follow simple rules based on common sense. Unlike the mathematical models, heuristics are used to involve a certain degree of randomization. Some examples of heuristic algorithms include Simulated Annealing, Genetic Algorithms, Ant Colony Optimization, Memetic Algorithms, Harmonic Search Algorithm, Partial Swarm Optimization, Artificial Bee Colony Optimization, Tabu Search Algorithm, Variable Neighborhood Search, Randomized Iterative Improvement with Composite Neighboring algorithm, and Great Deluge Algorithm. There are also multi-criteria and multi-objective techniques, which can be an application of various algorithms of the two techniques presented above to obtain better results [3].

This work aims to propose and implement an algorithm that will try to solve the UCTTP. The algorithm consists of two main phases. In the first phase, different heuristics will be used to find a feasible schedule, that is, to meet all the hard constraints. In the second phase, we use an evolutionary algorithm with a simulated annealing-based mutation to comply with the most significant amount of soft constraints and hard constraints if necessary. Once implemented, different algorithm parameters such as temperature, fitness function, and crossover will be tuned to determine the optimal values that allow a better solution to be reached.

The rest of this document is organized as follows: Chapter 2, Theoretical Framework, shows a revision of chronological techniques used to solve the University Course Timetabling Problem (UCTTP) and various types of scheduling problems. Chapter 3, the Methodology, describes the procedure performed to achieve the desired results. In Chapter 5, Results and Discussion, the results obtained will be shown and contrasted with the results of other works to evaluate the algorithm's performance. Additionally, we will study the impact of the parameters on the quality of the resulting timetable. Finally, Chapter 6 will show the conclusions and recommendations that can be deduced from the research.

1.2 Contribution

The contribution of this work lies in the second phase of the proposed strategy, the optimization phase. We attempt to optimize timetables by using a Genetic Algorithm whose mutations are based on a well-known heuristic, Simulated Annealing. To the best of our knowledge, a similar integration of heuristics has already been applied to problems such

as robotic task point ordering and combinatorial optimization problems, but never to the Course Timetabling Problem.

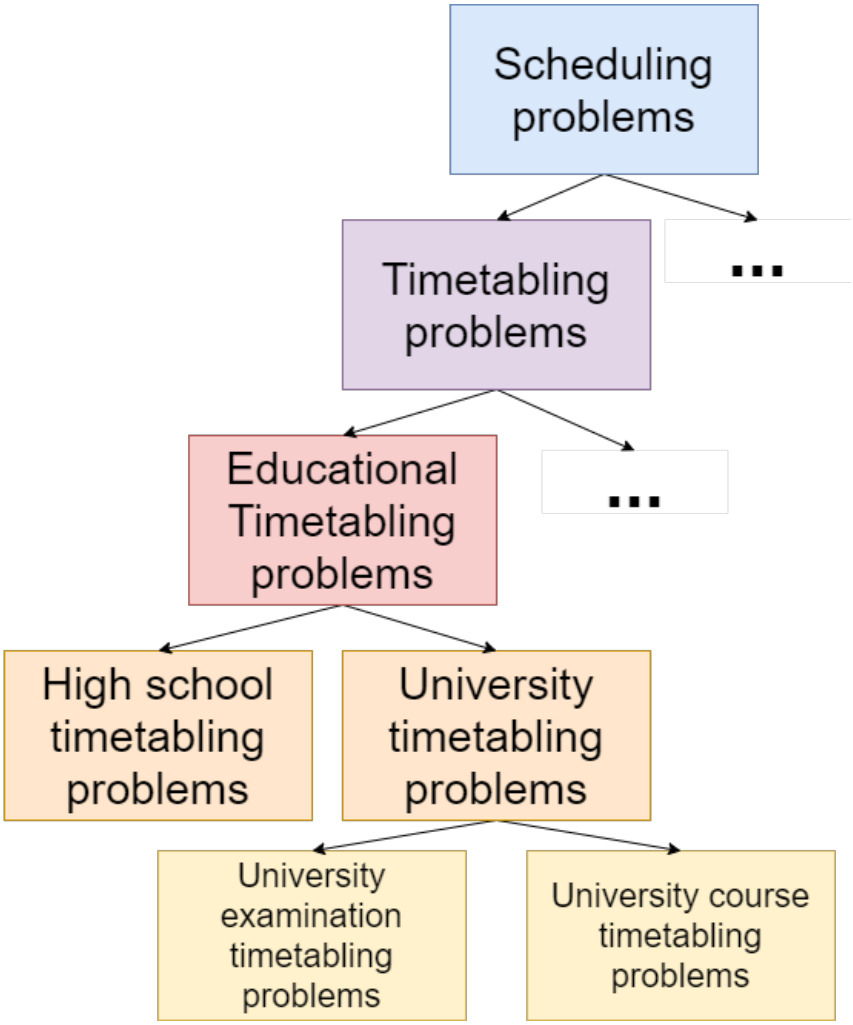

Figure 1.1: Scheduling problems.

Chapter 2

Theoretical Framework

The variety of techniques that have been used for tackling the UCTTP can be classified into three main groups: Mathematical models, Heuristic approaches, and Intelligent and agent-based approaches.

It is common for many researchers and practitioners to search for a feasible timetable (i.e. a timetable that meets all hard constraints) by using deterministic, mathematical models, as is the case in [5, 6]. Once such a feasible timetable is obtained, an optimization phase is used to improve the quality of the initial timetable through a set of successive permutations. Such optimization used to be implemented with heuristic algorithms.

In the beginning, this problem was defined as a graph problem $G =< C, T, \hat{R} >$ where the nodes are classes C and teachers T linked by $\hat{R}$ parallel edges, and the objective was to color the nodes in such a way that no two adjacent edges have the same color [7]. After that, several definitions have been proposed to describe the problem. This chapter presents a description of different algorithms to solve this problem in different categories according to its classification showed in [3].

2.1 Mathematical models

This category includes theory-based technique, Integer/Linear programming method, and constraint satisfaction programming [8]. One of the first techniques applied to solve Timetabling problems was graph coloring. In the studied papers, this problem is proposed as a set of events (nodes) connected through edges (rooms), and the main objective is that the resulting graph has the least possible chromatic number, that is, to occupy the least amount of time slots [8]. This method was used to describe conditions that are both necessary and sufficient for a solution to more general class teacher timetable problems [9]. Subsequently, There is also an approach to real problems on timetabling through graphs, specifically on the class teacher model (i.e., a problem where the only variables that are taken into account are lectures and teachers) and course scheduling [8]. Later on, some techniques used to color them are shown to obtain a feasible timetable; It should be noted that the distinction between feasible and optimal solutions is raised. Since then, there

have been several works using this approach, and over the years, new technologies have been developed in order to improve results, such as algorithms based on RFID technology, which allows to identify and transmit the information of an object through radio frequency waves [10, 11]. Subsequently, in several works, this technique has been used to obtain a feasible result and later apply optimization algorithms to try to improve this solution [12].

2.2 Heuristic approaches

Heuristic approaches can be defined as simple procedures, often based on common sense, that are supposed to offer a good solution (although not necessarily the optimal one) to complex problems in an easy and fast way [13]. A few of these algorithms will be shown below, and some crucial works on each approach.

2.2.1 Simulated Annealing

The Simulated Annealing Algorithm simulates the physical process of heating a material and then slowly lowering the Temperature to decrease defects, thus minimizing the system energy [9]. The main objective of this algorithm is that given an initial Temperature, which in this case is a variable, try to find a way for the solution variable to approach an optimal value in such a way that the decrease in Temperature is adequate, so that allows to achieve a global optimum value [14]. The pseudocode of this algorithm is described in Algorithm 1.

In the case of UCTTP, we are trying to minimize the value of the objective function that represents a decrease in the violation of constraints. There have been several works in which this has been investigated. For example, in [1] good results were obtained using this method together with an adequate statement of the problem. On the other hand, at the 2003 ITC, Kostuch showed on [12] that it is possible to get excellent results as he reached first place. The key was assigning weights in the objective function and modifying according to a correlation of the temperature decrease with variables that change while the algorithm runs. On the other hand, [15] showed that with the same 2003 dataset, it is possible to obtain even better results using an imperialist competitive algorithm.

2.2.2 Genetic Algorithm

A genetic algorithm is an evolutionary algorithm and therefore is considered a heuristic. A genetic algorithm is based on the Darwinian principle of organisms trying to survive their predators by adapting to their environment. The members who best adjust to the environment will have the best chances of survival, which implies that their children can inherit the characteristics that allow adaptation, thus ensuring this species' survival. The genetic mutations that allow adaptation occur randomly in members of species, so there will be cases in which the inherited genes will be better and they may not. Each individual generated by the Genetic algorithm is called a chromosome and is a possible solution to

the optimization problem. Each chromosome is made up of genes that represent decision variables. The fitness values of individuals determine their ability to survive. Each generation contains a mixture of a parent population, which contains surviving individuals (chromosomes) from the previous generation and their children. Children, which represent new solutions, are generated by genetic operators that include crossing and mutation. The higher the fitness value, the greater the chance of survival and reproduction [14].

Algorithm 1: Simulated Annealing Pseudocode

```
1  Begin
2  Input paramaters and initial data
3  Generate initial solutions X and evaluate its fitness function
4  Let β = number of new solutions to reach termal equilibrium
5  while termination criteria are not satisfied do
6      for j = 1 to β do
7          Generate a new solution X^new and evaluate its fitness value
8          if the new solution X^new is better than the old one X then
9              Put X = X^new
10         else
11             Evaluate P(X, X^new)
12             Generate Rand from the range [0, 1] randomly
13             if P(X, X^new) > Rand then
14                 Put X = X^new
15     Decrease the temperature
16 Report the solution X
17 End
```

The pseudocode of this algorithm is represented in Algorithm 2. First, we initialize the parameters and provide a set of initial individuals. Next, we initialize the main loop, where each iteration will be a new generation of solutions. In general, the acceptance criterion is determined by fulfilling a fixed number of generations. The fitness value of each individual is evaluated, and the parents are determined through a selection method. Subsequently, a crossover phase is carried out. A couple of parents are chosen, and a new solution is generated, inheriting characteristics and is expected to be better than the previous generation. Then, in the mutation phase, some individuals are modified randomly to have a varied population. In the end, we will have a new population made up of parent and child solutions that will be the starting point for a new generation.

In finding a solution for university course timetabling, there have also been a great variety of papers relating different approaches. In the case of [12], for example, the performance of the algorithm is analyzed by comparing the effects of various fitness functions and the effects of a stochastic local search. It shows that adequate handling of these metrics can lead to excellent results. Another investigation is shown in [16] where the idea is to make distributed model exchange an island's local best individual with another island. As a result, this model can run faster and decrease the violations of the restrictions allowing it to reach a more quickly optimal value.

Algorithm 2: Genetic Algorithm Pseudocode

```
1  Begin
2  Input paramaters and initial data
3  Generate initial solutions X and evaluate its fitness function
4  Let R and M be the number of parents and the size of the population
5  Generate M initial possible solutions
6  while termination criteria are not satisfied do
7      Evaluate fitness value of all solutions
8      Select the parent population with a selection method
9      for j = 1 to R do
10         Generate Rand from the range [0, 1] randomly
11         if Rand < P_C then
12             Parent j is known as an effective solution
13         else
14             Parent j is known as an ineffective solution
15     for j = 1 to (M − R)/2 do
16         Select two solutions randomly with the uniform distribution from effective
             parents of the parent population
17         Generate two new solutions with the crossover operator
18         Put newly generated solutions into the children population
19     for j = 1 to M − R do
20         for i = 1 to N do
21             Generate Rand from the range [0, 1] randomly
22             if Rand < P_H then
23                 Replace the decision variable i from solution j X_{j,i} using the
                     mutation operator
24     Set population = parent population + children population
25 Report the population
26 End
```

Another exciting research is the use of memetic algorithms. This algorithm is defined as a subject in computer science that considers complex structures such as the combination of simple agents and memes, whose evolutionary interactions lead to intelligent complexes capable of problem-solving [17]. In this type of algorithm, what is sought is that the local search together with the population algorithm works cooperatively to achieve a better search process. In the Qaurooni [18] approach, this algorithm was used to try to solve the UCTTP. As a result, it can be seen that with the approach applied in this algorithm, it is possible to obtain good results for the use of small and medium instances.

2.2.3 Ant Colony

Ant Colony is a technique based on ants' behavior when looking for a path between the colony and a food source. The ant colony optimization algorithm consists of agents called artificial ants searching for reasonable solutions to a given optimization problem. The problem consists basically in finding the best path on a weighted graph. Ants incrementally build solutions by moving on the graph. The solution construction process is stochastic and biased by a pheromone model, a set of parameters associated with graph components related to nodes or links whose values are modified at runtime by the ants [19].

The pseudocode of this algorithm is described in Algorithm 3. As can be seen, we have to initialize the parameters and provide an initial solution. For each iteration, we first evaluate the fitness function of the paths found by each ant. Later we calculate the value of the number of pheromones for each path. This value is the one that tells us the best path to take into account. Subsequently, we select one of the values of each decision and construct a solution, which is going to be the order in which the edges of the graph should be followed. At the end of the loop, we select the ant with the best solution.

Using this technique for solving UCTTP, it is possible to obtain better results than other algorithms, and above all, these algorithms can handle problems with multiple heterogeneous [20].

Algorithm 3: Ant colony Pseudocode

```
1  Begin
2  Input paramaters and initial data
3  Let M be the population size and N the number of decision variables
4  Let D_I be the number of possible values for decision variable i
5  Generate M initial possible solutions randomly
6  while termination criteria are not satisfied do
7      Evaluate fitness value of all solutions
8      for i = 1 to N do
9          for d = 1 to D_i do
10             Update pheromone concentration of possible value d for decision
                variable i
11             Evaluate probability of possible value d to be selected
12     for j = 1 to M do
13         for i = 1 to N do
14             Randomly select a value for decision variable i among possible values
                based on their probabilities
15 Report the ants or solutions
16 End
```

2.2.4 Tabu Search

Tabu search is a local search heuristic that relies on specialized memory structures to avoid entrapment in local minima and achieve an effective balance of intensification and diversification.

Formally, Tabu Search is an improvement of the local search methods. The basic concept of local search is that the move is always from a worse solution to a better one. The optimum achieved with the local search is primarily a local optimum rather than a global optimum since the algorithm always moves towards an improved neighbor solution close to the current one. Tabu search solved convergence to local optima experienced by allowing movements to no improving solutions when there is no better solution near the current solution. Also, it takes advantage of the principles of artificial intelligence by making search movements based on memory structures that prevent repetitive movements and help to explore the decision space of the optimization problem more thoroughly [14].

Algorithm 4: Tabu Search Pseudocode

```
1  Begin
2  Input paramaters and initial data
3  Let X' be the best point and X the current search point
4  Generate a search point X randomly and evaluate its fitness function
5  Set X' = X
6  while termination criteria are not satisfied do
7      Generate neighbor points around the searching point and evaluate their fitness
         values
8      if all neighbor points are tabus and cannot satisfy the aspiration criteria then
9          Stop algorithm and report the best point X'
10     Select the best neighbor point which is not tabu or satisfies the aspiration
         criteria
11     Put X − the selected point
12     if X is better than X' then
13         Set X' = X
14     Update the tabu list
15 Report the best point
16 End
```

The pseudocode of this algorithm is described in Algorithm 4. We first initialize the corresponding parameters and provide an initial and optimal solution, which will be the same at the beginning. We enter a loop where: First, we generate neighbors around my solution and evaluate its fitness function. The algorithm is stopped if all the neighbors are tabus, or worse solutions, and do not meet the aspiration criterion. Now select the best neighbor and make it the current solution. If that solution is better than my best solution, I make it my best solution. Later I update my tabu list and repeat the same process until reaching a completion criterion. Finally, I report the best solution found.

As a result of comparing this algorithm with other heuristic approaches, it can be seen that Tabu Search obtains excellent results and similar to those of Genetics algorithms for

medium instances. In contrast, for large instances, feasible solutions are obtained, but not optimal solutions [21].

2.3 Intelligent and agent-base approaches

The use of hybrid approaches, especially in a single or multi-agent with artificial intelligence, is relatively new and has shown excellent results. A system with agents in which one or more agents can solve the problems must be correctly implemented. It is necessary to carry out tests to determine the number of agents necessary to solve the problem. In the case of timetabling problems, there are publications where several tests have been carried out, obtaining generally good results for the various problems raised in terms of speed [22, 23]. However, it is also possible to solve an agent-based system using reinforcement learning techniques [24].

Chapter 3

State of the Art

3.1 Comparison with other works

In the case of [25], they tried to solve the problem with these datasets using Group-Based Operators, paying particular attention to the resolution of hard constraints. In this case, the author divided the datasets into low, medium, and high difficulty. Although it does not show the specific result of each dataset, it does mention that the number of unassigned events is between 20% and 40%, which is relatively high in comparison with the results of this investigation where the unassigned events reach only 2% in a few datasets, thus showing that in this case, it is better to use the evolutionary algorithm with simulated annealing-based mutation.

In [1], the authors used Simulated Annealing to solve the optimization phase. Before starting this phase, this algorithm already has a feasible solution for all the datasets obtained by a similar method to the "Step 1" used in this investigation. Their schedule also has the last five timeslots of each day free, which is one of the problem's soft constraints, resulting in a much better schedule (better fitness value) than the one presented here before starting the optimization phase. The results can be seen in Table 3.1 and show a much better solution than that shown in this investigation. However, it is important to take into account the number of iterations used to solve this problem, which in the case of [1] was 1.14×10^8 while in this research was 500.

3.1.1 Competition results

As mentioned above, this dataset was used for a competition, so the results are available on the official site [26]. Unfortunately, only the results of the best solutions are exposed, so a fair comparison cannot be made with the various algorithms used in the competition. As in [1], in all cases, the solution is given in 2 phases, always obtaining a feasible schedule and not assigning events in the last five timeslots to reduce the number of soft constraint violations. As a result, as shown in Table 3.1, it can be seen that the violations of the soft constraints are lower than those of this investigation.

Dataset	Soft constraints violations Comparison				
	A1	A2	A3	SA	Our proposal
COMP01	45	85	63	16	385
COMPO2	25	42	46	2	373
COMPO3	65	84	96	17	430
COMPO4	115	119	166	34	736
COMPOS	102	177	203	42	702
COMPO6	13	6	92	0	672
COMPO7	44	12	118	2	613
COMPO8	29	32	66	0	629
COMPO9	17	184	51	1	405
COMP10	61	90	81	21	378
COMP11	44	73	65	5	432
COMP12	107	79	119	55	515
C0MP13	78	91	160	31	557
C0MP14	52	36	197	11	732
COMP15	24	27	114	2	652
COMP16	22	300	38	0	502
C0MP17	86	79	212	37	717
C0MP18	31	39	40	4	422
C0MP19	44	86	185	7	760
C0MP20	7	0	17	0	519

Table 3.1: Comparison of soft constraints violations from the competition (A1, A2, A3), the algorithm provided by [1], and the results of the current research.

3.2 Latest research on UCTTP

Since 1963, several investigations have been carried out in the field of UCTTP. As shown in the previous section, the problem of scheduling has been solved mainly through the use of hybrid metaheuristic algorithms since they solve this problem in a more optimal way. In recent years a few researchers have tried to solve specific problems that exist in general when carrying out the optimization phase. Below we explain a few advances in the algorithms used in recent years to solve specific problems when solving UCTTP.

One of the main problems of UCTTP is the search for a neighbor timetable in the optimization phase. In other words, find a timetable that meets at least the hard constraints. In recent years, several solutions to this problem have been proposed, and one of the most recent is described in [27]. According to this research, in general, there is no way to explore the entire domain of possible solutions because there are a large number of solutions, and many of them cannot be used unless the hard constraints are met. This solution also proposes to solve the problem of finding a neighbor timetable as a combinatorial problem, that is, to move an event to a random position. In this way, a new neighbor can be found in a more reasonable time.

In [28], an algorithm was implemented to solve a Paechter dataset from 2007 [29]. They used a hybrid algorithm, a mix of Hill Climbing, Great Deluge and Simulated Annealing. Although no information was shown on obtaining a feasible schedule, several analyses were carried out on the parameters involved and how to optimize them to obtain better results. Something remarkable about this research is the development of a variable called "Neutrality", which determined the degree of randomness of some metaheuristic algorithms. For example, it was determined that the Neutrality of the Simulated Annealing algorithm is 50%, which means that for every bad solution generated, a good one is generated.

The algorithm shown in [30] was also used to solve the Paechter datasets of 2007. This algorithm also developed some techniques to solve the problem of finding neighboring times. In this case, the metaheuristic algorithm MAX-MIN Ant System was used, where the evaluation was modified in such a way that schedules that cannot achieve a good score in the iteration are quickly abandoned. In this case, feasible solutions were generated for all cases. In the optimization part, they obtained an efficient algorithm between time and quality of the schedule, while without using the intelligent evaluation function developed in this work, it is possible to reach a global minimum at the cost of a long execution time.

In [2] a novelty technique is applied to solve the UCTTP. Although it uses a genetic algorithm for the optimization phase, this work presents an exciting way to solve the problem of finding a neighboring time. Here they propose using a clustering algorithm that identifies the candidate schedules for reasonable solutions and generates many possible solutions around them as sown in Figure 3.1. This is an entirely new idea that could solve this problem in the future since it is one of the most sensitive parts of the process. Although this algorithm was tested on only one dataset, preliminary results show that execution time can significantly improve, which converts it into a fantastic alternative for future works.

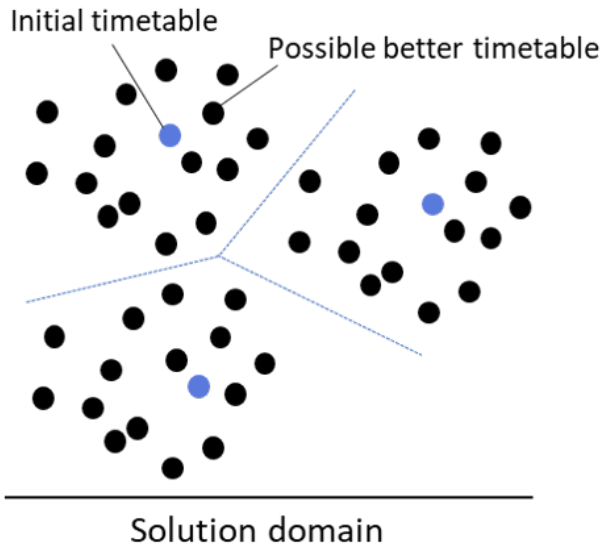

Figure 3.1: Clustering method to find a neighbor timetable described in [2].

Chapter 4

Methodology

4.1 Problem description

The University Course Timetabling Problem (UCTTP) consists of allocating objects (teachers, students, courses) in predefined timeslots and rooms fulfilling hard and soft constraints. The objects to allocate are not always the same and depend on the specific case of each dataset. In our case, the objects to take into account are students, events, and features described as follows:

- A student: Person authorized to take one or more courses at the university.
- Event: An event is a course or lab that a student may attend.
- Feature: A feature is a characteristic of a classroom, e.g. a projector

The objects mentioned above must be allocated into a set of timeslots and rooms. The problem consists of 45 timeslots corresponding to 5 days of 9 hours. The number of rooms, as well as the number of students, events, and features, changes in each dataset. In this case, 20 datasets will be evaluated.

When constructing a timetable, the proposed algorithm must meet the following constraints.

Hard constraints:

- No student attends more than one event at the same time.
- The room is big enough for all the attending students and satisfies all the features required by the event.
- Only one event is in each room at any timeslot.

Soft constraints

- A student must not have classes in the last slot of the day.
- A student must not have more than two classes consecutively.
- A student must not have a single class on a day.

4.2 Dataset Description

The increasing interest of various universities to solve the UCTTP problem has led to several datasets available for use. A significant advance in the automatic creation of UCTTP datasets is the software developed by Ben Paechter, which has been used for various competitions related to this topic [26]. One of these competitions is the "International Timetabling Competition" celebrated for the first time in 2002 by the Practice and Theory of Automated Timetabling organization (PATAT).

The competition provided 20 datasets, also called "instances" [26]. Each of the instances contains the necessary information to solve the problem. The data of each of these 20 datasets are described in Table 4.1 and specifies the following information:

- The number of events, $numEvents$, that need to be scheduled, the number of rooms available, $numRooms$, the number of features to be considered, $numFeatures$, the number of students already registered, $numStudents$. Besides, it is known that there are 45 timeslots available in which events should be assigned.
- An array called $RoomCapacity$ of size $numRooms$ where each element a_i represents the maximum capacity of i room.

$$RoomCapacity = \begin{bmatrix} a_0 & a_1 & \cdots & a_{R-1} \end{bmatrix}$$

- A boolean matrix $StudentEvent$ of size $numStudents \times numEvents$, where each element $a_{i,j}$ can be 1 if student i attends to event j, or 0 otherwise.

$$StudentEvent = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,E-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,E-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{S-1,0} & a_{1,1} & \cdots & a_{S-1,E-1} \end{bmatrix}$$

- A boolean matrix $RoomFeature$ of size $numRooms \times numFeatures$, where each element $a_{i,j}$ can be 1 if room i include the feature j, or 0 otherwise.

$$RoomFeature = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,F-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,F-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{R-1,0} & a_{1,1} & \cdots & a_{R-1,F-1} \end{bmatrix}$$

- A boolean matrix $EventFeature$ of size $numEvents \times numFeatures$, where each element $a_{i,j}$ can be 1 if event i requieres the feature j, or 0 otherwise.

$$EventFeature = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,F-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,F-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{E-1,0} & a_{1,1} & \cdots & a_{E-1,F-1} \end{bmatrix}$$

All datasets can be downloaded from the official website of "The First International Timetabling Competition" `http://sferics.idsia.ch/Files/ttcomp2002/IC_Problem/node3.html` in a compressed file with a .zip extension. Each dataset is a plain text file containing the information, so it is first necessary to preprocess it according to the same website's instructions to obtain the arrays and matrices mentioned above.

4.3 Solution description

The ultimate goal of this research is to obtain a schedule represented as the $TimeslotRoom$ matrix where the rows represent the timeslots and the columns represent the rooms. Each place in the schedule, i.e., cell in the matrix, can contain an event or contain a -1, which means no event.

$$TimeslotRoom = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,R-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,R-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{T-1,0} & a_{1,1} & \cdots & a_{T-1,R-1} \end{bmatrix}$$

In order to obtain the schedule, we propose an evolutionary algorithm with simulated annealing-based mutation, which will be in charge of minimizing the number of violations of soft constraints and, much more importantly, hard constraints. In order to apply this algorithm, it is essential to have a previous solution with the most significant number of events already assigned in the schedule. That is why a series of heuristics has previously been applied to assign the most significant number of events before using our main algorithm.

4.3.1 Step 1: Obtaining an initial timetable

Before creating a schedule, two matrices were created that will help eliminate the hard constraints. The first is the $EventEvent$ boolean matrix, where its rows and columns are

events and each element $a_{i,j}$ represent if the couple of events can be assigned in the same timeslot or not. In this way, we fulfill the first hard constraint of the problem "No student attends more than one event at the same time."

$$EventEvent = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,E-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,E-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{E-1,0} & a_{1,1} & \cdots & a_{E-1,E-1} \end{bmatrix}$$

The *EventRoom* boolean matrix where its rows are events and its columns are rooms is another important matrix to fulfill hard constraints. The $a_{i,j}$ element of the matrix represents if the $i-th$ event is compatible with the $j-th$ room. In this way, the remaining two hard constraints will be fulfilled, "the room is big enough for all the attending students and satisfies all the features required by the event" and "only one event is in each room at any timeslot."

$$EventRoom = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,R-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,R-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{E-1,0} & a_{1,1} & \cdots & a_{E-1,R-1} \end{bmatrix}$$

To generate an initial schedule, start with a blank schedule and a list with the unassigned events. The objective now is to fill this schedule with these events fulfilling the hard constraints, that is, consulting the *EventEvent* and *EventRoom* arrays to determine whether or not an event can be assigned to a place on the schedule. To make the assignment, the four heuristics that were also used in [12] and described below were used. If an event could not be assigned to any available space, it becomes part of a list of unassigned events.

- H_1. Choose a place in the time slots with the fewest events assigned.
- H_2. Pick a random place.
- H_3. For each unassigned event, delete the events of a random timeslot and then assign the remaining events.
- H_4. Apply H3 repeatedly.

The heuristics shown below were applied in the same order in which they are listed. After carrying out this process, it is expected to have a schedule where all its events stay assigned and fulfill all the hard constraints.

4.3.2 Step 2: Optimizing the initial timetable by Evolutionary algorithm with simulated annealing-based mutation

The Evolutionary algorithm with simulated annealing-based mutation algorithm is the one that has been proposed to carry out the optimization phase of this problem; the main objective will be to fulfill as many soft constraints as possible and, if possible, assign the unassigned events to the schedule.

To carry out this algorithm, the four mechanisms on which a traditional evolutionary algorithm is based were taken as support: evaluation, mutation, crossover, and selection. The algorithm has been implemented as shown in the Algorithm 5. The implementation of each of these mechanisms to resolve UCTTP is described below.

Algorithm 5: Evolutionary Algorithm with Simulated Annealing Base mutation

```
1 setInitialpopulation
2 set Temperature, numIndividuals, numGenerations
3 for i = 0 to numGenerations do
4     applyParallelSA(population)
5     applySelection(population, childPercentage)
6     applyCrossover(population)
```

Initial population, *setInitialpopulation*

As a result of the previous phase, i.e. Step 1, an initial schedule was obtained where all events are probably assigned. Now it is necessary to create an initial population which means a new set of schedules. Heuristic 4 shown above was applied to the initial schedule to obtain all the individuals. This process was repeated multiple times in order to obtain an initial population of timetables while ensuring genetic diversity.

Parallel Simulated Annealing, , *applyParallelSA*

This process aims to simulate the mutation and evaluation of traditional Genetic Algorithms. This function receives a timetable and returns a better timetable, i.e. with less soft constraints violated.

The SA algorithm consists in that for each iteration, some neighbors of the current state s are evaluated, and probabilistically, it is decided between making a transition to a new state s' or staying in-state s. The way these elements are calculated is described below:

- **Temperature (T)**: The temperature decreases geometrically as $T_{i+1} = \beta \times T_i$ until it reaches a final temperature. Each new value of T represents a new iteration of

the Algorithm. The initial and final temperature change in each generation of the Evolutionary Algorithm.

- **Acceptance probability**: The acceptance probability is calculated as $p = e^{-\Delta F/T}$ where ΔF is calculated as $|s - s'|$.
- **Neighbors**: Each neighbor is a new schedule based on the current state to which a swap events process was made.
- **Swap events**: Process where the schedule of my current state is taken, then 2 of its events are chosen, and their positions are exchanged as long as the feasibility of the schedule is not compromised. The result is a new schedule.
- **evaluation**: Each new schedule is evaluated employing a fitness function, which is calculated as $F = WH \times UE + WS \times \sum SCV$, where WH is the weight for the hard constraints, WS is the weight for soft constraints, UE is the number of unassigned events, and SCV is the number of violations to soft constraints.

This algorithm runs in parallel, which means that each process of apply SA to an individual is handled as a different thread at the processor level, which results in a significant improvement in time compared to doing it sequentially.

Selection, *applySelection*

Selection is an essential step in an evolutionary algorithm because it allows each generation to be better than the previous one. We select a specific percentage of the best individuals of the population based on its fitness value. Then we delete the remaining individuals that will be replaced in the next step.

Crossover, *applyCrossover*

Crossover is an important phase that allows the creation of new individuals. The process consists of taking two random individuals of the population who will act as parents. Subsequently, an identical copy of parent one is made, and events from parent two are added to that copy in the same positions as parent 2. As a result, as shown in Figure 4.1, there is a new schedule that can be better than its parents.

From this process, it should be noted that two problems arise in the child schedule. First, we need to handle the event already assigned by parent one before adding the event from parent two. The second problem is that there will be duplicates between the events of parent one and events of parent two. The solution for both problems is to add the duplicated event and the already assigned event to the list of unassigned events so that at the end, a reconstruction process is applied where they are assigned, fulfilling feasibility, or keeping as unassigned events.

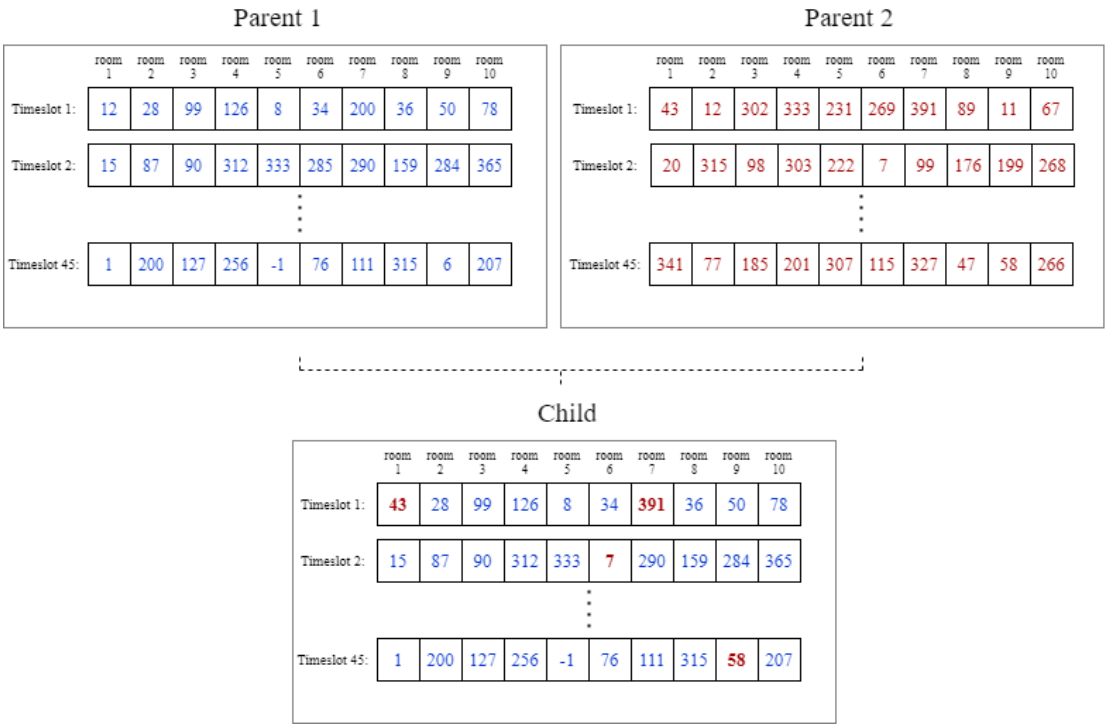

Figure 4.1: Crossover example to obtain a new individual.

Dataset ID	Data description					
	Number of events	Number of rooms	Number of features	Number of students	Average classroom capacity	Average events per student
COMP01	400	10	10	200	10.40	17.76
COMP02	400	10	10	200	10.40	17.23
COMP03	400	10	10	200	10.80	17.71
COMP04	400	10	5	300	15.30	17.43
COMP05	350	10	10	300	17.30	17.77
COMP06	350	10	5	300	17.90	17.77
COMP07	350	10	5	350	20.60	17.48
COMP08	400	10	5	250	12.80	17.58
COMP09	440	11	6	220	10.36	17.35
COMP10	400	10	5	200	10.70	17.78
COMP11	400	10	6	220	11.40	17.41
COMP12	400	10	5	200	10.30	17.58
COMP13	400	10	6	250	12.60	17.69
COMP14	350	10	5	350	20.30	17.42
COMP15	350	10	10	300	17.40	17.58
COMP16	440	11	6	220	10.72	17.75
COMP17	350	10	10	300	17.20	17.67
COMP18	400	10	10	200	10.50	17.56
COMP19	400	10	5	300	15.30	17.71
COMP20	350	10	5	300	17.50	17.49

Table 4.1: Description of the 20 datasets.

Chapter 5

Results and Discussion

5.1 Experimental Setup

5.1.1 Equipment Description

The algorithm designed to solve this problem was developed in Python 3.6.9, and the libraries used include *concurrent.futures*, which was used for the parallelization of the Simulated Annealing step, *numpy* for the handling of lists, matrices, and the generation of random numbers. *itertools* to perform permutations, very useful in the preprocessing phase; *networkx* for fast handling of the maximum matching algorithm. Regarding the equipment used to test the algorithm, a virtual instance was used with a processor Intel® Xeon(R) CPU @ 2.30GHz × 16, 62.8GiB of RAM, a Graphic card llvmpipe (LLVM 10.0.0, 256 bits), and as Operative System uses Ubuntu 18.04.5 LTS.

5.1.2 Experiments

In order to test the algorithm's performance, several experiments have been created that will allow us to analyze better the behavior of the different parameters and important elements of the algorithm and know how to modify them to obtain the best results.

In general the hyperparameters used in each experiment are described below al least that it change according to the experiment:

- Initial Temperature (T_0): 3
- Final Temperature (T_f): 0.005
- Number of generations: 500
- Population Size: 12 individuals
- Temperature decrease: $T_{i+1} = T_i \times 0.99^{counter}$

- Percentage of new children in the population: 20%
- Crossover changes: 5 events

Changes in Temperature

Temperature is a significant parameter of Simulated Annealing since it allows modifying the state of an individual even if it was worse than the previous one. Theoretically, it is known that the higher the temperature, initially the individual will have more negative changes, which causes more diversity in the long term. 4 experiments will be carried out where the initial Temperature T_0 will be 100, 50, 10 and 3.

Crossover

Crossover is a process in which two individuals mix to create a new one. This process has already been described previously. The objective is to determine if it causes a significant difference in results. Then the algorithm will be executed with crossover and without crossover.

Fitness Function

The fitness function is another significant parameter since it is the one that measures how good the algorithm is and directly affects the acceptance probability in the Simulated Annealing algorithm and the selection in the Evolutionary Algorithm. According to [25], in Simulated Annealing, it is better to use a fitness function of maximization to one of minimization. Then two formulas will be used to prove that. F1 adds the number of Violations of hard and soft constraints, so the objective is to minimize it, and F2 is a function of the form $1/x$, so the objective is to maximize it.

$$F1 = WH * UE \ + \ WS * \sum SCV$$

$$F2 = \frac{1}{WH * UE \ + \ WS * \sum SCV}$$

5.1.3 Parameters

In order to measure the quality of results in all the experiments, these parameters will be taken into account:

- Unassigned events: Represents the number of unassigned events in a schedule.
- Soft Constraints Violations: Shows the total of soft constraints violated

5.2 Experimental observations

Once the experiments were carried out, important conclusions could be reached about the operation of the algorithm. In the first place, in most cases, it is possible to obtain a feasible schedule. That is, all the hard constraints were met without leaving out any event. On the other hand, in those datasets where feasibility was not achieved, the number of unassigned events is very low compared to the number of total events. In fact, in the worst case, the amount of unassigned events is less than 2% of the total. Fulfilling of soft constraints, on the other hand, showed very similar behavior in all experiments. It can also be seen that as the generations advance, it is much more difficult to decrease the number of soft constraint violations.

5.2.1 Testing with different Initial Temperatures

In the first set of experiments, we evaluated the consequences of having different initial temperatures. For the twenty datasets, four initial temperatures (T_0) were used: $T_0 = 100$, $T_0 = 50$, $T_0 = 10$, $T_0 = 3$. Figure 5.1 shows the datasets where it was not possible to obtain a feasible timetable. As can be seen, there are between 2 and 4 unassigned events for each experiment where feasible solutions could not be found, which represent a minimal number compared to the total number of events. The worst-case happen with a $T_0 = 10$ with up to 1.43% of unassigned events. The datasets where a feasible schedule could not be found are COMP07, COMP12, COMP14, COMP19, and they all have in common a very low value of features (e.g. projector in a room) compared to the other datasets, so it can be said that a low number of features generates a more difficult problem to solve.

Table 5.1 shows the number of soft constraints violated by each dataset in each experiment. On average, the lowest number of violations (i.e. the best timetable ever found) was achieved with a $T_0 = 3$, while the highest number of violations was reached with a $T_0 = 50$. It shows that with a low initial temperature, better results can be achieved. In connection with the above, there are not too many negative changes when starting the algorithm with a high temperature, as shown in Figure 5.2. The case of COMP06 at To=100 is an exception. This antecedent shows that mixing an evolutionary algorithm with parallel simulated annealing tends to generate good solutions without the temperature being relevant as it would happen if only Simulated Annealing were used.

5.2.2 Testing with different Fitness functions

Two experiments were carried out regarding the variation of the fitness function, using a minimization function $F1$ and a maximization function $F2$, both described previously. About the fulfilling of hard constraints, it can be seen that it is much better to use maximization than minimization, since as seen in Figure 5.3, in the case of maximization, it only failed in 1 of 20 datasets to find a feasible solution, being the best result of all the experiments carried out.

Regarding the violation of soft constraints, it can be observed in Table 5.1 that it is

Dataset	Soft constraints violations in each experiment							
	$T_0 = 3$	$T_0 = 10$	$T_0 = 50$	$T_0 = 100$	Min FF	Max FF	NoRec	Rec
COMP01	385	410	387	426	385	428	385	414
COMP02	373	417	431	390	373	447	373	406
COMP03	430	510	525	465	430	506	430	506
COMP04	736	920	842	891	736	738	736	830
COMP05	702	851	750	746	702	669	702	775
COMP06	672	605	666	611	672	650	672	719
COMP07	613	718	719	615	613	712	613	802
COMP08	629	527	529	545	629	538	629	539
COMP09	405	451	479	417	405	477	405	494
COMP10	378	410	481	429	378	437	378	446
COMP11	432	514	473	510	432	488	432	493
COMP12	515	479	533	547	515	510	515	507
COMP13	557	661	621	665	557	605	557	652
COMP14	732	653	702	671	732	740	732	741
COMP15	652	606	718	603	652	679	652	706
COMP16	502	461	480	439	502	509	502	444
COMP17	717	740	852	723	717	746	717	770
COMP18	422	407	441	434	422	398	422	424
COMP19	760	655	696	687	760	716	760	811
COMP20	519	555	479	484	519	614	519	544

Table 5.1: Soft constraints violations in each experiment.

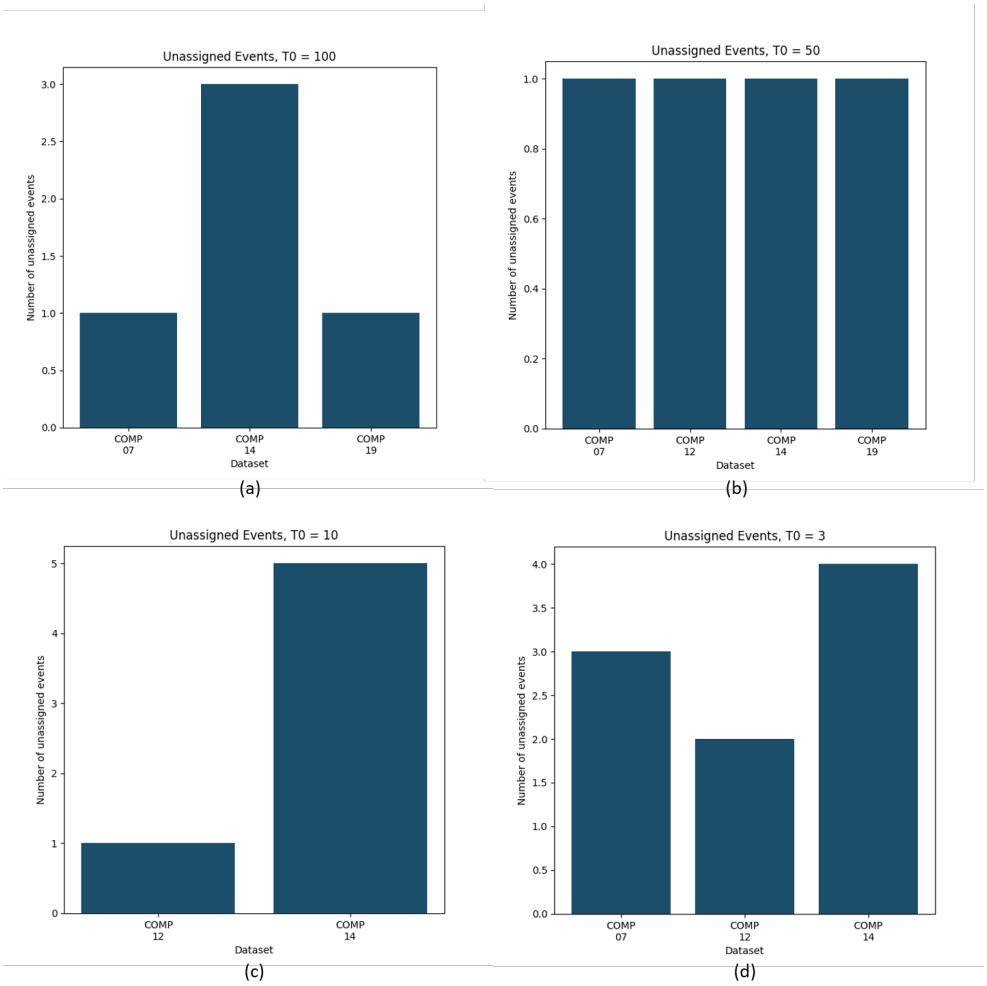

Figure 5.1: Unfeasible datasets obtained with different initial temperatures. (a) $T_0 = 100$, (b) $T_0 = 50$, (c) $T_0 = 10$, (d) $T_0 = 3$.

better to use minimization in most cases, although it can also be noted that there are not many differences. Something interesting is what is shown in Figure 5.4 where the convergence of the fitness maximization function does not seem to arrive at an asymptote as if it is the case of the fitness function of minimization, which raises the suspicion that with the increase of generations, better results can probably be achieved. It can also be seen that the use of the fitness maximization function produces a more significant change, both negative and positive, in the number of soft constraints violated throughout the Generations, which is possibly due to more significant changes in acceptance probability of the parallel simulated annealing algorithm.

5.2.3 Testing the use and not use of crossover

The objective of this experiment is to determine the impact of crossover in the final solution. Two experiments were carried out where crossover is used in one while not in the other. Regarding the hard constraints, no difference was found between the two experiments

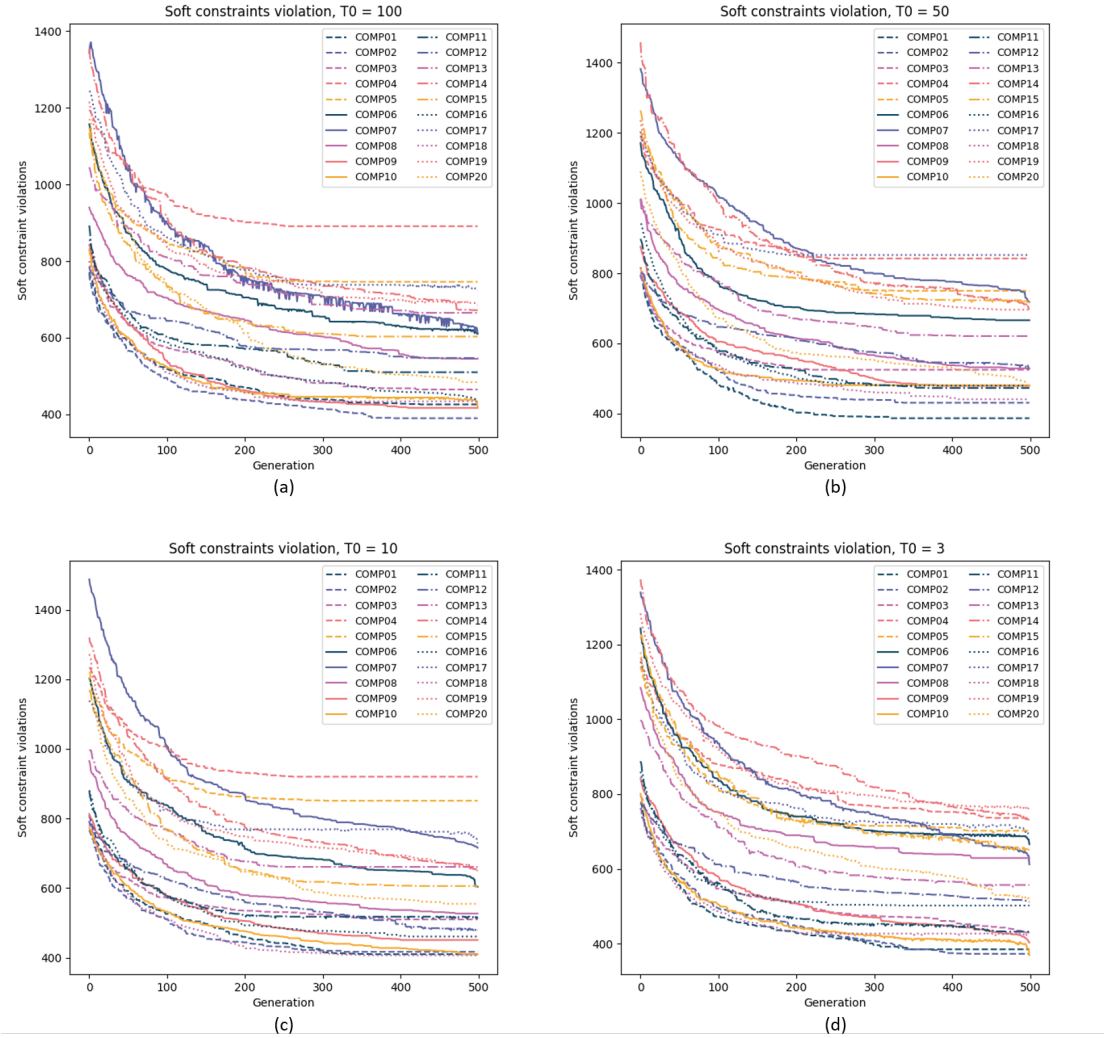

Figure 5.2: Fitness value when using different T_0. (a) $T_0 = 100$, (b) $T_0 = 50$, (c) $T_0 = 10$, (d) $T_0 = 3$.

because in both cases, the amount of non-feasible datasets was 3, as shown in Figure 5.5, representing 15% of the total datasets. Also, in both cases, the number of unassigned events was less than 1% for these not feasible schedules.

The violations to the soft constraints described in Table 5.1 show a notable improvement in the use of crossover compared to not using it. In the case of convergence, as shown in Figure 5.6, it is possible to observe that there is a tendency to generate better results in both cases, and there are almost no negative results, which shows that this mechanism tends to generate only better solutions.

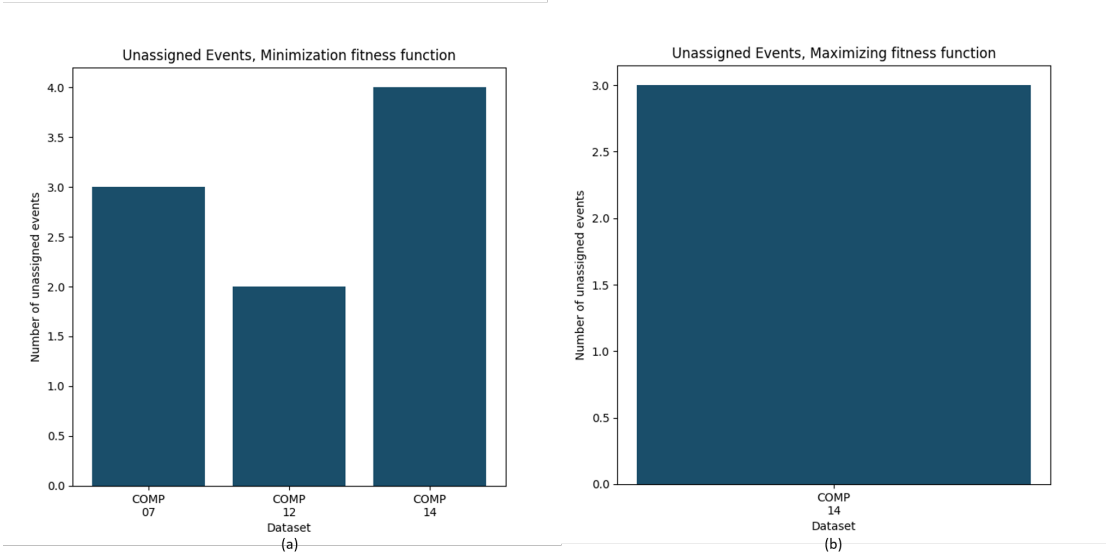

Figure 5.3: Unfeasible datasets. (a) using a minimization function, (b) using a maximization function.

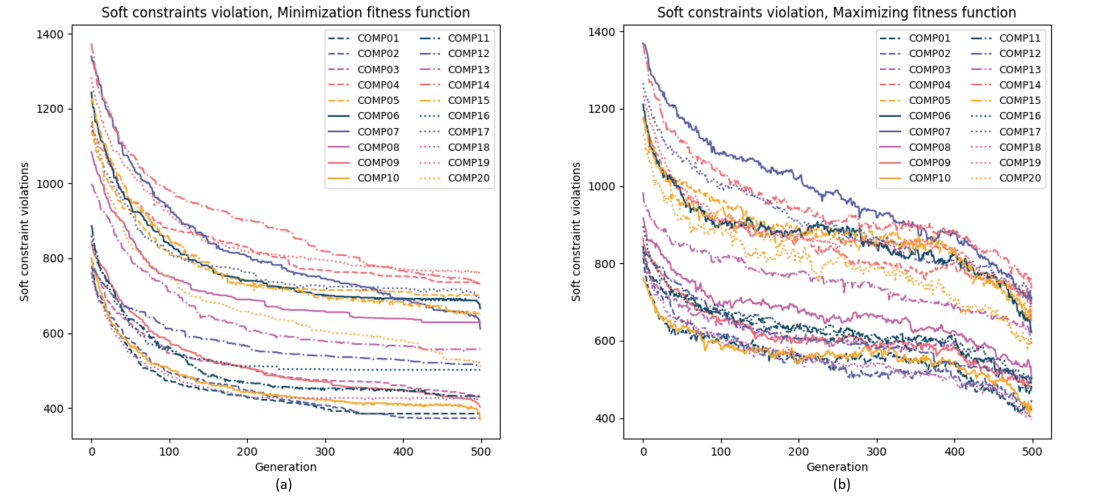

Figure 5.4: Fitness value when using different fitness function. (a) using a minimization function, (b) using a maximization function.

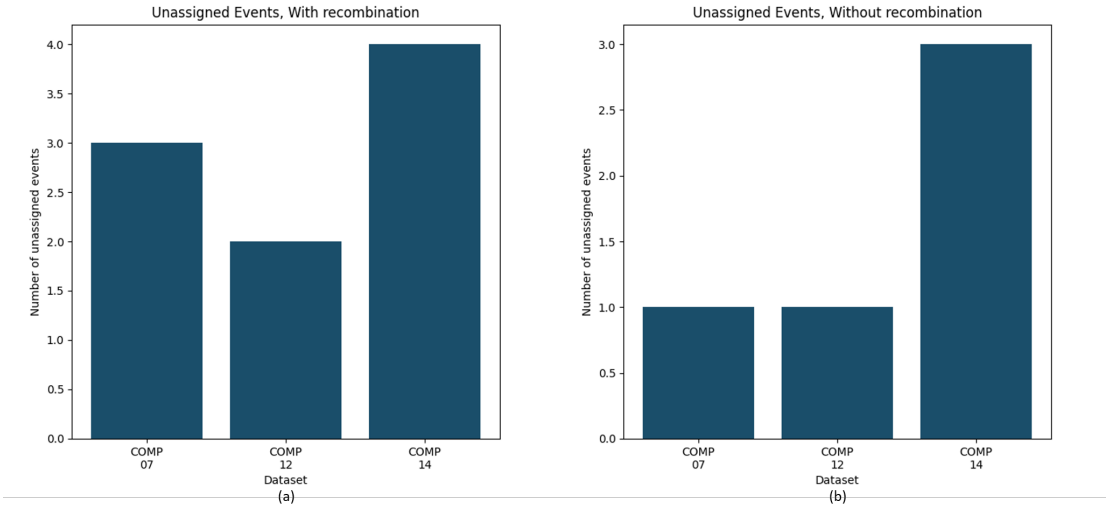

Figure 5.5: Unfeasible datasets. (a) using crossover, (b) not using crossover.

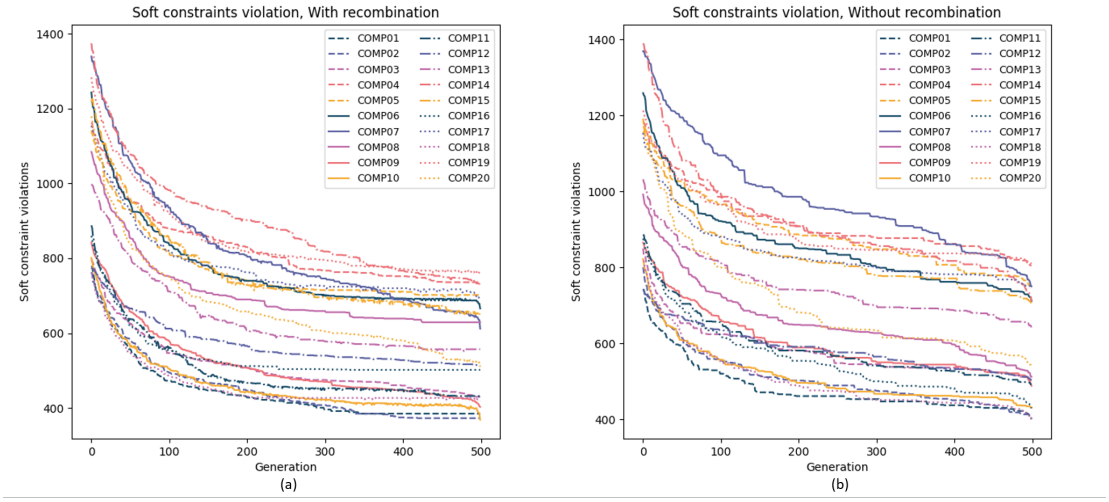

Figure 5.6: Fitness value when using and not using crossover. (a) using crossover, (b) not using crossover.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

1. In the present work, an evolutionary algorithm with a simulated annealing-based mutation was implemented. The quality of the resulting schedules was evaluated by measuring the number of assigned events and the violations of the soft constraints in the resulting schedule. In order to evaluate the efficiency of this algorithm, different experiments were carried out testing the most critical parameters, such as Temperature, fitness function, and recombination phase.

2. In the experiments where the Temperature was varied, it was possible to observe that a low temperature is necessary to achieve better results solving hard and soft constraints. Mixing an evolutionary algorithm with parallel simulated annealing generates reasonable solutions without the Temperature being relevant as it would happen if only Simulated Annealing were used.

3. The experiments where the fitness function was changed showed that this parameter considerably modifies the final results since the results vary depending on whether a maximization or minimization function is used. Although we obtained better results when using a minimization function, the curve's tendency suggests that a maximization function might eventually lead to better results if the number of iterations is significantly increased.

4. As to the use of crossover, the experiments showed that it has a positive impact on the resulting schedules. Specifically, it was detected that in 17 of the 20 datasets, it was better to use crossover and also that the convergence towards a new solution is slower when crossover is not used.

5. Once the algorithm presented in this work is compared with other similar algorithms, we concluded that more changes are needed to reach solutions such as those obtained by [1]. Although good results have not yet been obtained, it was possible to fulfill the majority of the hard constraints of the schedules with shows that this algorithm can be an option to be considered.

6.2 Future Work

1. Although the algorithm can generate a feasible schedule, the optimization phase still needs to be improved. According to the results, the optimization phase can converge towards better results, but it tends to stop at local minimums so that the best possible schedule cannot be obtained. It is necessary to improve the implementation of this algorithm through techniques that improve the convergence of the algorithm. For example, the search for "neighboring schedules" in the mutation phase is done randomly, but it would be better to look for a deterministic way that eliminates repeated schedules. This and many other improvements could expand the search domain of schedules, thus allowing a faster and better optimization.

2. In general, the entire algorithm used to solve the University course timetabling uses dozens of parameters that have not been taken into account individually. In the future, it is expected to be able to carry out a more exhaustive evaluation with each of these parameters to determine their interference in the algorithm's performance and be able to adjust them most appropriately.

3. For this work, we used the Ben Paetcher dataset [26], which provides data that allowed testing the algorithm. However, it is necessary to be able to apply this algorithm in real cases. This problem has many variants [31], mainly because the course timetabling process varies between countries and institutions. Hence, it is challenging to generate an implementation that solves this problem for all cases. Therefore, it is necessary to carry out future research on the variations of course-timetabling in different Universities and refactor the current implementation so that it is not difficult to add or remove new variables for other datasets.

Bibliography

[1] S. Ceschia, L. Di Gaspero, and A. Schaerf, "Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem," *Computers and Operations Research*, vol. 39, 04 2011.

[2] S. P. Erdeniz and A. Felfernig, "Ocsh: Optimized cluster specific heuristics for the university course timetabling problem," ser. ICIST '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3200842.3200858

[3] A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers and Industrial Engineering*, 07 2015.

[4] R. Lewis and B. Paechter, "An empirical analysis of the grouping genetic algorithm: the timetabling case," vol. 3, 10 2005, pp. 2856 – 2863 Vol. 3.

[5] A. Dandashi and M. Al-Mouhamed, "Graph coloring for class scheduling," *ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010*, pp. 1–4, 2010.

[6] H. A razak, Z. Ibrahim, and N. Mohd Hussin, "Bipartite graph edge coloring approach to course timetabling," 03 2010.

[7] C. Gotlib, "The construction of class-teacher timetables. proceedings of ifip," *Congressh*, vol. 62, p. 73–77, 1963.

[8] D. de Werra, "An introduction to timetabling," *European Journal of Operational Research*, vol. 19, no. 2, pp. 151–162, 1985. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:ejores:v:19:y:1985:i:2:p:151-162

[9] M. Solgi, O. Bozorg-Haddad, and H. Loaiciga, *Meta-heuristic and Evolutionary Algorithms for Engineering Optimization*, 11 2017.

[10] V. Nandhini, "A study on course timetable scheduling and exam timetable scheduling using graph coloring approach," *International Journal for Research in Applied Science and Engineering Technology*, vol. 7, pp. 1999–2006, 03 2019.

[11] S. Abdullah, E. Burke, and B. Mccollum, *Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem*, 01 2007, vol. 39, pp. 153–169.

[12] P. Kostuch, "Timetabling competition - sa-based heuristic," 2003.

[13] S. H. Zanakis and J. R. Evans, "Heuristic "optimization": Why, when, and how to use it," *Interfaces*, vol. 11, no. 5, pp. 84–91, 1981. [Online]. Available: https://EconPapers.repec.org/RePEc:inm:orinte:v:11:y:1981:i:5:p:84-91

[14] O. Bozorg-Haddad, H. A. Loaiciga, and M. Solgi, *Meta-heuristic and evolutionary algorithms for engineering optimization*, ser. Wiley series in operations research and management science. John Wiley & Sons, 2017. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=be63b50c9e43b8840f375c5cb5eb2b74

[15] K. Murray, T. Müller, and H. Rudová, "Modeling and solution of a complex university course timetabling problem," 01 2016, pp. 189–209.

[16] A. A. Gozali, "Asynchronous island model genetic algorithm for university course timetabling," 2014.

[17] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 02 2012.

[18] D. Qaurooni, "A memetic algorithm for course timetabling," 01 2011, pp. 435–442.

[19] "Scholarpedia ant colony optimization," http://www.scholarpedia.org/article/Ant_colony_optimization, accessed: 2020-11-12.

[20] K. Socha, J. Knowles, and M. Sampels, "A max-min ant system for the university course timetabling problem," vol. 1-13, 10 2002.

[21] O. Rossi-doria, M. Sampels, M. Birattari, M. Chiar, 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 metaheuristics on the timetabling problem," 03 2003.

[22] H. Asmuni, "Fuzzy methodologies for automated university timetabling solution construction and evaluation," 01 2008.

[23] R. Lewis, B. Paechter, and O. Rossi-Doria, *Metaheuristics for University Course Timetabling*, 01 1970, vol. 49, pp. 237–272.

[24] A. Mondal, "A survey of reinforcement learning techniques: Strategies, recent development, and future directions," 01 2020.

[25] R. Lewis and B. Paechter, "Finding feasible timetables using group-based operators," *Evolutionary Computation, IEEE Transactions on*, vol. 11, pp. 397 – 413, 07 2007.

[26] "PATAT international timetabling competition," http://www.scholarpedia.org/article/Ant_colony_optimization, accessed: 2020-11-12.

[27] H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A review of optimization algorithms for university timetable scheduling," *Engineering, Technology Applied Science Research*, vol. 10, pp. 6410–6417, 12 2020.

[28] T. Feutrier, M.-E. Kessaci, and N. Veerapen, "Investigating the landscape of a hybrid local search approach for a timetabling problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1665–1673. [Online]. Available: https://doi.org/10.1145/3449726.3463175

[29] "PATAT international timetabling competition 2007," http://www.cs.qub.ac.uk/itc2007/, accessed: 2021-11-21.

[30] J. Sakal, J. E. Fieldsend, and E. Keedwell, "Learning assignment order in an ant colony optimiser for the university course timetabling problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 77–78. [Online]. Available: https://doi.org/10.1145/3449726.3459534

[31] A. Rezaeipanah, Z. Abshirini, and M. Zade, "Solving university course timetabling problem using parallel genetic algorithm," *Journal of Scientific Research and Development*, vol. 7, pp. 5–13, 10 2019.