2019 International Conference on Advancements in Computing (ICAC)
December 5-6, 2019. Malabe, Sri Lanka

Intelligent Timetable Scheduler: A Comparison of Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms

Tiny Wijerathna Ekanayake, Pavani Subasinghe, Shawn Ragel, Anjalie Gamage, Suchini Attanayaka
Faculty of Computing,Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
tinywe@hotmail.com, pavanisubasinghe123@gmail.com, shawnrgl10@gmail.com, anjalie.g@sliit.lk
pavanisubasinghe123@gmail.com

***Abstract*— A Timetable scheduling is a monotonous task and a problem in an educational institute. This is because many rules and constraints are involved, which can be categorized as hard and soft constraints. Mainly, a university must produce two types of timetables, which are examination, and semester timetables. This paper has reviewed the Exam Timetabling problem with Genetic and Graph Coloring algorithms and the Semester Timetabling problem with Heuristic and Iterated Local Search algorithms. Our aim here is to develop a possible and correct solution for each timetabling problem using the above-mentioned four different approaches.**

Keywords— Genetic Algorithm; Graph Coloring Algorithm; Heuristic Algorithm; Iterated Local Search Algorithm; Scheduling; Examination Timetabling; Semester Timetabling

I. INTRODUCTION

A university must produce semester timetables, mid and final examination timetables every year which is repeating twice or thrice throughout every year. Since this is a complex task, it is very difficult to prepare timetables and schedules. If this task is done manually, it takes months and months to generate error and conflict-free timetables. Examination scheduling is done every year twice per semester and the Semester Timetable schedule is done every year per semester. Back then at Sri Lanka Institute of Information Technology (SLIIT), the examination schedule has been done manually by faculty timetable coordinator along with some group involved and there are some problems and issues met in the existing system that they use for semester timetable scheduling. Timetable task depends on all the dependencies such as subject, student, lecturer and hall allocation. To fulfill the requirement, we have decided to develop a web -based application for the timetable task using four different algorithms. The exam timetable components are implemented using Genetic Algorithm and Graph Coloring Algorithm and the semester timetable-schedule component was implemented using Heuristic Algorithm and Iterated Local Search Algorithm. This research presents an "Intelligent Timetabling Scheduler" to help the timetable coordinator in a fast and efficient way. Our research team deeply analyzed the past literature reviews related to these Scheduling systems and got to know that some systems are already implemented separately. According to the researches scheduling is a type of decision-making process used in almost all productions and service sectors. Currently, most of the timetabling and scheduling systems have considered hard and soft constraints along with the hall allocation, and student allocation. In this system, for the Examination Timetabling component, following are considered.

- Manage subjects' constraints for prioritizing the subjects.
- Subjects that cannot be assigned in the weekends
- 4th year coupling modules and common modules.
- Staff allocation according to their modules and preferences.

For the Semester Timetabling component of the system, the constraints identified are;

- No student will be involved in two or more lectures during the same time slot.
- No lecturer will be involved in two or more lectures during the same time slot.
- Only one event will be allocated to rooms and labs at any period.
- Events such as lectures and labs will be coupled generally.

In manual methods, there can be various outputs, according to the user's vision. Collect data, drafting and cross-check the time table is done manually, and it is a time-consuming task. If any changes are done they have to inform all the interested parties about the updates. Therefore, this research, attempts to develop the system in different algorithms and finally choose the most accurate one.

II. LITERATURE REVIEW

A. Constraint based methods

When timetable and scheduling is modeled, it must cover hard and soft constraints. A set of various variables and values such as halls, subjects, staff, and students must be assigned in order to satisfy hard and soft constraints whereas priority is given to hard constraints. Thereby we can conclude the objective of the above method as to minimize the violation of soft constraints and satisfy the hard constraints [3].

B. Sequential Methods

In this method, timetable problem is dealt as graph problem and then they order the events using domain specific heuristics and then assign the events sequentially into valid time slots, so no constraints are violated for each time slot [3]. In Graph coloring algorithm, scheduling problems are divided into so many parts and its content can be huge. Scheduling problem can be broken down to smaller parts using graph - coloring algorithm. In graph, node can represent subject and edge represents conflicts. That's according to the Subject graph. We can also draw so many graphs using graph-coloring algorithm, as conflicts graph [4].

C. Cluster methods

The problem is divided in to number of events / groups. Group is defined so that hard constraints are satisfied along with soft constraints [3].

D. Metaheuristic Methods

Heuristic algorithm is one that is designed to solve a problem in a faster and more efficient way than traditional methods by reducing optimality, accuracy, precision, or completeness for speed. We are using it to bring up an optimistic timetable system based on this algorithm.

In general educational timetable problem, a set of events (e.g. courses and exams, etc.) are assigned into a certain number of time slots (periods) subject to a set of constraints, which often makes the problem very difficult to solve in real-world circumstances [5]. In fact, large-scale timetables such as university timetables may need many hours of work spent by qualified people or team in order to produce high quality timetables with optimal constraint satisfaction [6] and optimization of timetable's objectives at the same time. Given the generality of the algorithm operation, it can further be adapted to more specific scenarios, e.g. University, examination schedule and further be enhanced to create railway timetables. Thus, through the process of automation of the timetable problem, many hours of creating an effective timetable have been reduced eventually [7]. The crossing of the numerical sequences promotes the emergence of new sequences, formed from the first [8].

Iterated Local Search simply checks whether the new solution is better than the already known one. It iterate number of times using a neighbor generation method that creates a solution basing on the already existing one. Several ending criteria types are known for this type of algorithms and can be found in [9]. Novel cooling scheme is adopted [10] which uses different cooling rates for the two different acceptance criteria. Accepting a worse candidate solution is done based on a probability value and can be found in [11], and the problem of room allocation is considered as a bipartite matching problem in [12]. Stimulated Annealing (SA) local search procedure can be found in [11] to obtain a global minimum and when it is reached, the Stimulated Annealing (SA) search will be terminated. The cooling schedule is critical to the efficiency of Stimulated Annealing (SA) and it is used in [10].

Tabu Search is used to improve the local search to reduce the number of times. Tabu search does not use probability so extensively. This algorithm keeps a Tabu list (a list of moves that are not permitted). To minimize the time spent in the local minimum. After each move, it is added to the list to prevent the algorithm from going back. Tabu Search is a short-term memory after spending a tenure the move is removed from the list to assure that the optimal solution will not be prevented by the Tabu list [3].

E. Auto Generate System Based on Expert System (AGSS)

In this, they have proposed an AGSS which provides accessibility to the timetabling committee to arrange the detail by simply loading the information i.e. numbers of lecturers, list of halls, and list of subjects and students. Therefore, AGSS has developed an auto generated class time table using AI based experts with user customization options. They have used different optimization methods they are Particle Swam Optimization (PSO), Genetic Algorithm (GA), Tabu Search (TS), Ant Colony System (ACS), Fuzzy Logic (FL) and Stimulated Annealing (SA) to obtain fast computing abilities. The Expert system is used by considering its abilities to imitate extract and integrate between knowledge and human expert with data [13].

F. Modified Genetic Algorithm Method (GA)

GA**:** is a meta-heuristic algorithm that has an ability to search within a large space are to find a near-optimum solution. GA is one of the evolutionary algorithms inspired by evolution theory. Which searches a better solution for the next generation. In GA, a solution will be encoded to a string or an array value called chromosomes while each variable that influences the solution is called the gene.

G. Fuzzy Time Window (FTW)

Fuzzy is a method dealing with uncertainty and ambiguity variables. Fuzzy logic for FTW: All the subjects in a university will be divided into 2 groups based on its acquirement of thinking portion are exact (Prioritize) and non-exact (Non- Prioritize) subjects. Fuzzy logic can be used to measure how effective and satisfying when it is put in a certain period [14].

III. SYSTEM OVERVIEW

The Fig 1 of shows all the stages Genetic Algorithm Life Cycle below involve when scheduling the examination timetable.

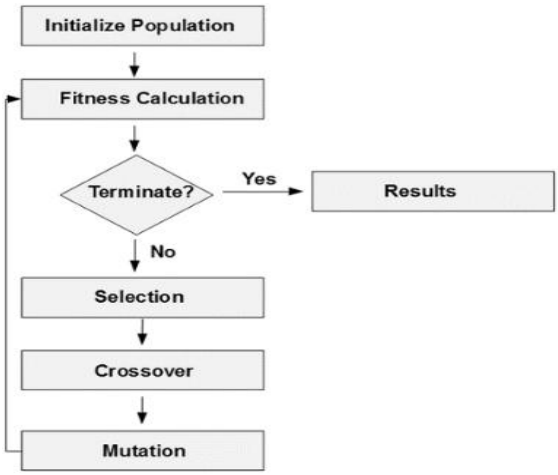


Fig 1 : Life Cycle of the Generic Algorithm

This Fig. 2 below shows the system Architecture of the timetable generation by using Graph Coloring Algorithm.

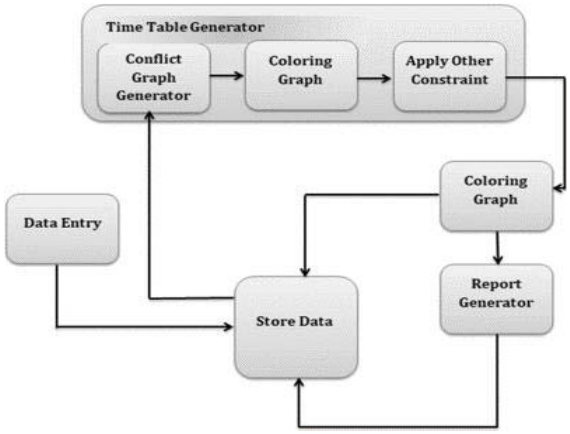


Fig. 2. Graph Coloring System Architecture

Fig. 3 shows the workflow of the timetable generation using Heuristic Algorithms.

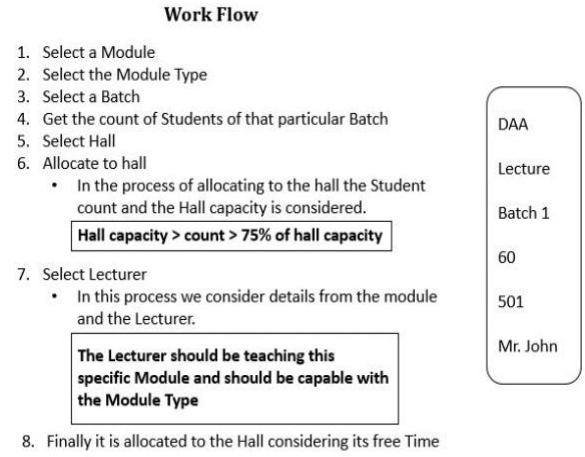
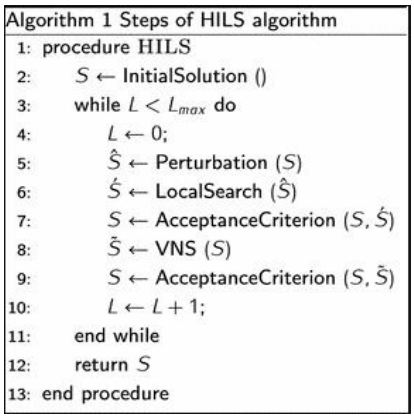


Fig. 3. Work flow of the Heuristic Algorithm

Fig. 4 below shows the steps of the Iterated Local Search Algorithm (ILS).



Algorithm 1 Steps of HILS algorithm

```
procedure HILS
    S ← InitialSolution ()
    while L < L_max do
        L ← 0;
        Ŝ ← Perturbation (S)
        Ś ← LocalSearch (Ŝ)
        S ← AcceptanceCriterion (S, Ś)
        S̃ ← VNS (S)
        S ← AcceptanceCriterion (S, S̃)
        L ← L + 1;
    end while
    return S
end procedure
```

Fig. 4. Steps of the ILS

IV. METHODOLOGY

In this section, the overall methodology of each algorithm is explained accordingly.

A. Genetic Algorithm

a) Generate Initial Population

To generate the initial population the Genetic algorithm keeps running until the initial population is generated. (Population – Hall, Students, Subjects, and Invigilators) Chromosomes are selected, from all population alternatives.

b) Calculate the Fitness Function for each population

Fitness function is calculated for each Population to get the best options such as, the total number of Exam Periods, number of Students are obtained for the subjects assigned to the sessions and examination hall. We can consider when the fitness function is 1.0 then the selected population has no clashes.

c) Selection

Elitism is used for selecting the best individual in a genetic algorithm. The best individual in a given population should directly transfer in order to prevent a decrease of the diversity in the solution without any processing. The roulette wheel method is used for selecting the best individual from a given population and it has a greater chance of being transferred to a new population. All the results are subtracted from the minimum value found in the population. The conformance value of the solution should not be negative. The fitness value obtained is divided by the sum of the fitness values of all the chromosomes, the result is rounded and then multiplied by 100 [3].

d) Crossover

Create two new chromosomes by exchanging the genes between chromosomes. Crossover occurs separately for each row of the chromosome and Crossover pairs are generated randomly for the entire population [3].

e) Mutation

After the cross over is being done in between chromosomes of every generation, group each chromosome which comprises the same genetic structure in the succeeding generations. To make this happen cross over may have to apply several numbers of times until the best chromosome group gets generated. Due to applying of cross over number of times the diversity of new child chromosomes is generated to parent. In order to avoid and replace individuals with the same qualities Mutation is applied to reduce reiteration of individuals.

B. Graph Coloring Algorithm

When solving exam timetable scheduling using graph coloring algorithm, subjects act as vertices. Edges are drawn depending on graph type [15]. One type is a conflicting graph. There should be common modules that cannot have the same time slots that type of conflicts represented in a conflicting graph. This algorithm initially find the maximum degree vertex selected and then find out all its triples [16]. The first and third vertex cannot be adjacent to each other. Then first and third vertex can have the same color. Likewise it continues to colour the whole graph by using the Algorithm for generating best graph,

Input : A weighted Graph
Output : A minimum spanning tree T for G

Let P be a partition of the vertices of G, where each vertex forms a separate set Let Q be a priority queue storing the edges of G and their weights.

T ⟵ ϕ
While Q ≠ ϕ do
(u, v) ⟵ Q. removeMinimumElement ()
If P.find(u) ≠P.find(u)
Add edge(u,v) to T
P.union(u,v) Return T

b) Algorithm for coloring graph

Input: A simple undirected graph G with vertices V (G) = {v1, v2, . . . , vn}. (The list of colors to be drawn from will be {1, 2, . . . , n}; does not necessarily use all of these colors.

Output: A vertex-coloring of G.
For i = 1 to n do

Li = {1, . . . , i} (Li is the list of colors that may be assigned to vi)
For i = 1 to n do

Set ci = the first color in Li (ci is the color assigned to vi)
For each j with i < j and (vi, vj) ! E(G) do

Set Lj := Lj \ ci

Return each vertex, the color it was assigned, and the total number of colors used.

C. Heuristic Algorithm

The algorithms distinguish themselves in the field of methods of optimization and search for the assimilation of the Darwinian paradigm of the evolution of species. The algorithms are processes of convergence [17].

a) In the First step, it takes the details of all the students from the specified batch and then sorts out the count of students in that specified batch.

b) Then it looks for a Hall, which is more suitable to assign that batch with the constraint of having the hall, filled at least to 70%. (Number of Students >= 70% of the hall capacity).

c) Then by sorting out the Module, a lecturer should be assigned to that batch for the Module. When assigning Lecturer, the availability of the Lecturer also should be verified.

d) Then for the final Optimization, a recheck is made to make sure that if there are any possibilities in improving the optimized results. Every time a new constraint is added, or a minor conflict happens, the Algorithm runs again to check that optimal results.

D. Iterated Local Search (ILS)

Iterated Local search algorithm starts with an initial solution and iterates several times to create a candidate solution based on the existing one. If the candidate solution found to be better than the existing one, the existing solution will be updated with the candidate. The search procedure continues until no solution improvement has occurred during several iterations.

a) Generate Initial Solution in ILS

To begin with, creating the initial solution, a random event (E) from the set of unassigned events (E) is selected and added to the temporary created set which is initially an empty set.

This is repeated until no events can be added to E without causing a student-clash with any event already in the selected events. Then the timeslot (t) is taken, and each event will be assigned to a room that comes under the selected timeslot (t). The events that cannot match with a room are removed from the events list (E_t) and added back to E. Initial solution is obtained after all timeslots are taken into consideration for the above process.

b) Improvement Perturbation

Once the initial solution is obtained, Iterated Local Search (ILS) uses an improvement perturbation mechanism to further improve the existing solution. To achieve this, swapping is done and the new solution is checked with the existing one for feasibility with the use of violations of the soft constraints.

We consider two types of swapping which are type one and two respectively; the swapping type one will swap events between two timeslots so that some unallocated events can be put in; and type two will swap events so that events which are related to the same lecturer and student batch can be placed consecutively in the same room. The search procedure continues until no solution improvement has occurred during several iterations.

V. RESULTS AND DISCUSSIONS

For experimental purposes, we have chosen SLIIT, as the case study. Data were gathered relating to the faculty and the student services. We generated a timetable for the examination including a list of students, subjects, halls, and invigilators. The parameters used in the Genetic Algorithm are given here (Population 100, Number of Generation 100, Crossover Rate 0.7, Mutation Rate 0.02).

The Genetic Algorithm improves the solution iteratively. In each run – a generation we kept track of each of the best fitness values. When the maximum number of iterations reached, the global best was the optimal solution. Genetic Operators play an important role in improving the solution. From selection makes, the population filled up with copies of the best individuals. Selection and Crossover gives an optimal solution. Mutation changes within the solution and generates new feasible solutions.

Both Heuristic and Iterated local search have comparably better accuracy for the small group of instances. For the medium and large group of instances. Iterated Local Search (ILS) has always obtained a better number of feasible solutions when it is compared with the heuristic algorithm. However, considering the overall number of instances, Iterated Local Search (ILS) has obtained 56 feasible solutions out of 60 instances, which significantly outperforms the reference algorithm, which is heuristic algorithm. Since we are more focused on finding the algorithm which gives the more accurate solution it is better to choose the iterated local search approach which is more focused on the accuracy of the solution by iterating through many times.

The following sample image shows how the generation is populated against the fitness value and the best solution is chosen once the fitness value equals to 1.0.

In conclusion, we can state that Genetic Algorithm can be used to allocate resources to the examination. However, there are few subjects that are scheduled. In this research, we identified Graph Coloring Algorithm is suitable to identify clashes very clearly also there are no duplicate allocations occur.

Table 1 presents the number of feasible solutions obtained by the Iterated Local Search (ILS) algorithm and the Heuristic algorithm (H) on three different sizes of instance groups. They are named as small, medium and big groups of instances, respectively. In column total instances, the number of total instances in each group is shown in parentheses.

Table 1. Size of the problem

	instance group	total instances	H	ILS
1	small	20	20	20
2	medium	20	16	19
3	large	20	10	17
total		60	46	56

As per Table 1, out of the total small instances, which is 20, both approaches, which are ILS and Heuristic, have obtained same number of feasible solutions, which is a better amount. For the medium group of instances, we can see that ILS has obtained 19 feasible solutions while H has only obtained 16, which is a lesser amount. For the larger group of instances ILS has obtained 17 while H has obtained 10 out of the total of 20. Overall, ILS has been able to find 56 feasible solutions while H has obtained 46 out of 60 instances.

Sample output of our generated timetables are given below,

```
Group: 11
Group Name: YEAR 4 : CS
Hall: D301
Invigilator: Ms. Nisha Jayasooriya
Module: Secure Software Engineering
Time: 20th slot FRI 16:00 - 18:00
--------------------
Batch: 45

Group: 12
Group Name: YEAR 4 : DS
Hall: N301
Invigilator: Ms. Dulani Perera
Module: Object Oriented Programming
Time: 23rd slot SAT 13:30 - 15:30
--------------------
Batch: 46

Group: 12
Group Name: YEAR 4 : DS
Hall: A509
Invigilator: Ms. Geethanjali Wimalaratne
Module: Operating Systems
Time: 10th slot WED 11:00 - 13:00
```

```
Date-Time              Hall    Subject

Year : 1
Semester : 1
20th:8.30-10.30        B502    IPE
22nd:8.30-10.30        B501    MIT
24th:8.30-10.30        B501    ICS
26th:8.30-10.30        B501    CS

Semester : 2
17th:8.30-10.30        B502    PS
17th:10.30-12.30       B501    EAP
20th:10.30-12.30       B501    SPM
22nd:10.30-12.30       B501    ISDM
24th:10.30-12.30       B501    IWT

Year : 2
Semester : 1
26th:10.30-12.30       B501    SE-I
17th:12.30-2.30        B501    OOP
20th:12.30-2.30        B501    DBMS-I
22nd:12.30-2.30        B501    OSSA
```

In this research, we have compared four algorithms,

- Genetic and Graph Coloring, Heuristic and Iterated and local search algorithms.

We were able to find the best algorithm based on running time of the algorithm as a percentage given below,

- Genetic Algorithm (GA) takes 25% of time
- Graph Coloring Algorithm (GCA) takes 11% of time
- Iterated Local Search Algorithm (ILSA) takes 44% of time
- Heuristic Algorithm (HA) takes 20% of time.

Fig 5. Number of clashes (as a %) of the algorithm

Fig 6. Number of clashes of the algorithm

VII. REFERENCES

[1] Antariksha Bhaduri, "University Time Table Scheduling using Genetic Artificial Immune Network", IEEEXplore 17 November 2009 [PDF].

[2] Jumoke Soyemi, John Akinode , Samson Oloruntoba, "Electronic Lecture Time-table Scheduler using Genetic Algorithm", 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress.

[3] Esraa A. Abdelhalim, Ghada A. El Khayat, "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)", Alexandria Engineering Journal, www.elsevier.com/locate/aej.

[4] S. Saharan and R. Kumar, "Graph Coloring based Optimized Algorithm for Resource Utilization in Examination Scheduling," vol. 1201, no. 3, pp. 1193–1201, 2016.

[5] Edmund K Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, Rong Qu, "A Graph- Based Hyper-Heuristic for Educational Timetabling Problems." European Journal Operational Research, 176:177-192, 2007.

[6] W. Legierski, "Constraint-based Techniques for the University Course Timetabling Problem", CPDC, (2005), pp.59 -63.

[7] Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole "An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach" International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012.

[8] José Joaquim Moreira" A System of Automatic Construction of Exam Timetable Using Genetic Algorithms" proDEI, Faculdade de Engenharia da Universidade do Porto Rua Dr. Roberto Frias 4200-465 Porto, Portugal.

[9] T. Padale, H. Mane, M. Bhagat and S. Nangare,"A Literature Review on Timetable Generation Algorithms", *Imperialjournals.com*, 2019.www.imperialjournals.com/index.php/IJIR/article/view/3621.

[10] P. Kostuch, "The University Course Timetabling Problem with a Three-Phase Approach", *Practice and Theory of Automated Timetabling.*

[11] T. Song, S. Liu, X. Tang, X. Peng and M. Chen, "An iterated local search algorithm for the University Course Timetabling Problem", 2019.

[12] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, New Jersey.

[13] Ruth Ema Febrita and Wayan Firdaus Mahmudy, "Modified genetic algorithm for high school time-table scheduling with fuzzy time window", IEEEXplore, 1 March 2018 [PDF], ieeexplore.ieee.org.

[14] Paresh M. Chauhan, Kashyap B. Parmar and Mahendra B. Mendapara, "Solving Time-Table Scheduling Problem by Novel Chromosome Representation Using Genetic Algorithm", 2015International Conference on Circuit, Power and Computing Technologies [ICCPCT].

[15] R. Ganguli and S. Roy, "A Study on Course Timetable Scheduling using Graph Coloring Approach," vol. 12, no. 2, pp. 469–485, 2017.

[16] Vinayak Sapru, Kaushik Reddy and B Sivaselvan, "Time Table Scheduling using Genetic Algorithms employing Guided Mutation" 2010 IEEE International Conference on Computational Intelligence and Computing Research.

[17] Queirós, F. H.: Construção auztomática de Horários de Aulas. Tese de Mestrado, Universidade Portucalense (1995).