University Timetabling Problem

Group E

Abstract

This paper explores an approach to solve the university timetable planning problem through constraint-based reasoning. This timetabling problem is formulated as a constraint satisfaction model, and various techniques like constraint propagation and backtracking are implemented. The dataset from the University of Brasilia is employed to define variables and constraints. The proposed solution integrates constraint programming and optimization with a simulated annealing-like approach for efficiently creating an optimal solution. The study demonstrates the capability of constraint based programming and provides a foundation for exploring hybrid approaches to solve the timetabling problem.

1 Introduction

A. Wren defines timetabling Wren (1996) as follows:

> "Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

Timetabling problem can be defined as a process of allocating a number of events or activities to different resources which can be time (timeslots), space (rooms), and personnel (teachers) which satisfies different constraints for these resources. This problem spans educational institutions, such as schools, colleges and universities, airline industries, hospitals and also sports wherein they are used for educational timetabling, crew assignments, nurse and doctor scheduling and match fixtures in a league. Proposed solutions to timetabling problems date back to the 1980s, but the field still continues to be an active area for research. Recent papers in operational research and artificial intelligence journals highlight the ongoing efforts to tackle the complexities inherent in timetabling with the availability of new and advanced computing facilities and techniques.

This paper focuses on the University Course Timetabling Problem. University Timetabling is an administrative activity for a lot of institutions around the world. It is a very complicated task as it aims to obtain the ideal feasible timetable in each institution based on their specific rules, conditions and structure. As the years go by, these institutions grow rapidly with the number of registered students increasing, meaning more teachers, subjects, classes and hard manual work. The construction of a timetable can be an extremely difficult task and its manual solution can require much effort and the proposed solution may be unsatisfactory and time consuming. Thus, university timetabling problems have attracted the scientific community from a number of disciplines (including Operations Research and Artificial Intelligence) for automating the solutions to help accelerate the search process for the optimal timetable. Several algorithms have been presented over the years to solve the university timetabling problem, however many publications argue regarding the complete

automation and say that human touch is still necessary to guide and reinforce the search process.

There have been multiple approaches to timetabling problems which have been described in literature and tested on real data. We have reviewed these methods and employed some of them in our proposed solution. Here are some approaches and methods presented by Carter and Laporte (1996) : 1) sequential methods, 2) cluster methods, 3) constraint based methods, 4) meta-heuristic methods.

1. Sequential Methods

   These methods first order the events using the heuristics for the domain and then sequentially assign them into valid time periods so that there are no conflicts between events within each period (Carter, 1986). The timetabling problems are approached as a graph colouring method, where the problem is represented as a graph where events (courses) are represented as vertices, while the conflicts between the edges are represented by edges (de Werra, 1985). For example, if students have to attend two events there is an edge between the corresponding nodes which represent this conflict. A conflict-free timetable can be constructed by modeling it as a graph colouring problem. Each time period is represented by a colour in the graph and it is ensured that no two adjacent vertices share the same colour. Various heuristics are proposed for graph colouring used and ordered based on the difficulty of the problem. These heuristics are widely employed since they are easy to implement, however they lack the power of more modern intensive search methods. Nevertheless, their hybridisation with other search methods remains pivotal in modern timetabling research.

2. Cluster Methods

   These methods initially categorize the set of events into groups, ensuring that events within each group satisfy the hard constraints. Then, these groups are allocated to specific time periods to meet the soft constraints. However, such approaches may yield poor timetables in terms of quality.

3. Constraint Based Methods

   In this method, the events in the timetabling problem are represented by a set of variables, to which values (resources like rooms and time-periods) are assigned adhering to some constraints which are defined by us. A set of rules is established to guide the allocation of resources to events. When the propagation of the assigned values leads to an infeasible solution, a backtracking process is initiated (Brailsford et al., 1999). This process reassigns all the values and is continued until solution is achieved, satisfying all constraints required.

4. Meta-Heuristic Methods

   A range of meta-heuristic approaches like simulated annealing, tabu search, genetic algorithms and hybrid approaches have been explored for solving the timetabling problem in the past two decades. There are some very good results reported by using hybrid methods (Burke and Carter, 1998a). Meta-heuristic methods begin with one or more initial solutions and employ search strategies designed to avoid getting stuck

in local optima. While these search algorithms generate high quality solutions, they often come with considerable computational cost and significant tuning of parameters.

This paper, discusses key features of examination timetabling problems and applies the constraint based programming method to solve the timetabling problem. Section 2 describes the course timetabling problem and the constraints involved. The proposed solution involving the constraint based approach and an optimization process is discussed in Section 3. In Section 4, the benchmark problem is discussed and experiments are conducted to showcase the effectiveness of the approach in addressing the problem. The limitations of the work, future research opportunities and the conclusions are given in Section 5.

2 Problem Formulation

Our goal was to define a weekly timetable for university students. We wanted to find a proper time, room and students for all the courses. Each course has a set of acceptable timings and room assignments. This means, for example, that the only rooms that meet specific requirements like the correct size, equipment required, room type, etc. are listed for a course. We have assumed that there are 5 working days, from Monday to Friday for which the timetable needs to be defined. The time slots are divided into 2 hour periods, with each lecture being of the same length (2 hours). The operational hours for the university are 8:00 AM to 6:00 PM, and all classes are to be organised within this time span.

We use the Constraint Satisfaction Problem (CSP) solver to solve this timetabling problem. In a CSP, we have a set of variables with known domains and a set of constraints that impose restrictions on the values those variables can take.

To formally define a CSP, we specify:

- A finite set of variables $X = \{X_1, X_2, \ldots, X_n\}$
- For each variable X_i we define a (finite or infinite) domain $D = \{D_1, D_2, \ldots, D_n\}$
- A finite set of constraints, $C = \{C_1, C_2, \ldots, C_p\}$, where each can involve any number of variables, representing relationships between variables.

We solve the CSP by assigning a value to each of these variables such that all constraints are satisfied.

In terms of a CSP, a timetabling problem can be specified as treating the courses as variables of the problem. The rooms and time slots, instead, are the domain for each variable, while the constraints are various relationships between these variables. The domains and variables together determine a set of all possible assignments (solutions) that can be complete or partial.

2.1 Variables

Let's use the following notation:

- S is the set of student ids, $S = \{S_1, S_2, \ldots S_m\}$. For example, $S = \{100115, 100140..\}$
- L denotes the set of lessons/courses, $L = \{L_1, L_2, \ldots L_n\}$. For example, $L = \{$Primary Teaching, Political Sociology..$\}$.

- T denotes the daily time slots $T = \{T_1, T_2, \ldots, T_{25}\}$. T={Monday 8:00-10:00, ..., Friday 16:00-18:00}.

- R denotes the set of rooms, i.e. the set containing different room addresses. $R = \{R_1, R_2, \ldots R_t\}$. For example, R = {SALA PAT AT 045, FE-5 SALA DE AULA 04...}.

2.2 Constraints

We define constraints as the relationships between these variables.

- C_1: *Student-time clashes:* Each student has at most one lecture at any given time, i.e. a student must have just one course in a particular time slot. That means for a course L_i, if we have a time-slot $T(L_i)$, and for course L_j we have time-slot $T(L_j)$, then $T(L_i)$ and $T(L_j)$ must not be the same if the student of course L_i, $S(L_i)$, is the same student as student for course L_j, $S(L_j)$ (i.e. it's a single student).

$$T(L_i) \neq T(L_j)\ if\ S(L_i) = S(L_j), \quad \text{where} \quad 1 \leq i, j \leq n$$

- C_2: *Room-time clashes:* Each room is occupied with at most one course at any given time. Say for course L_i, we have room $R(L_i)$, and for course L_j we have room $R(L_j)$, then $R(L_i)$ and $R(L_j)$ must not be the same if time-slot for L_i, $T(L_i)$ is same as the time-slot for course L_j, $T(L_j)$, i.e.

$$R(L_i) \neq R(L_j)\ if\ T(L_i) = T(L_j), \quad \text{where} \quad 1 \leq i, j \leq n$$

3 Proposed Solution

The timetabling problem has been approached through an efficient solution which employs the principles of constraint based programming. Leveraging the Google OR-Tools CPSolver, the code formulates the data set for timetabling after pre-processing, as a constraint satisfaction problem (CSP). After pre-processing the data set, by filtering rows with missing information and extracting relevant data required for the code, a conflict dictionary is created to handle course conflicts. The primary objective is to minimize the conflicts and the time taken for all the courses and allocate classrooms, time-slots and resources to courses and students.

Hard and soft constraints are carefully considered during the approach. Hard constraints rigidly warrant the robustness of the solution. They include prevention of overlapping classes, ensuring that no two classes occur simultaneously in the same room. A constraint ensuring that there is no conflict between lectures for students enrolled in multiple courses also helps create a viable solution. Some soft constraints integrated into the code contain a preference of a timetable having consecutive classes on different days to accommodate student preferences and allow for free afternoons. The OR-Tools CP Solver navigates the solution space effectively and finds an optimum solution and timetable for the problem. The solution meets the basic requirements for an institution and also provides a foundation for optimization in the future to meet additional constraints.

3.1 Constraint Dictionary Initialization

In this section, a dictionary is initialized to store conflicts between courses, and this is crucial while handling constraints related to students enrolled in multiple courses. It iterates over each course, and students enrolled in that course are identified. For each student, other courses that they are enrolled in are determined, and then a dictionary entry for the current course is created, mapping to a list of courses with which it conflicts.

3.2 OR-Tools CP Solver

This part contains the core implementation of the timetabling problem using the OR-Tools CP Solver. An instance of the Constraint Programming model is created using the OR-Tools library. This is a representation of the timetabling problem and contains the variables, constraints and an objective function. Variables representing lesson start and end times and intervals are defined, and constraints to ensure no overlap, maintain precedence over lessons and, resolve and avoid conflicts are imposed on these variables to satisfy the requirements of the problem. The Solver is then initialized and executed and it employs various techniques like constraint propagation and backtracking to explore the solution space and find a feasible solution which satisfies the defined constraints. The timetable is obtained with classes assigned and rooms allocated and constraint programming is efficiently used to solve the timetabling problem.

3.3 Soft Constraint Optimization

A soft constraint optimization which follows a simulated annealing-like approach is used to enhance the quality of the solution. After obtaining the initial timetable, a soft constraint evaluation function is defined which assigns a value to each timetable, considering preferences of courses in adjacent time slots and allowing free afternoons. This optimization process explores random permutations of the timetable and evaluates each permutation's value using the soft constraint function. A new working timetable is produced if the new permutation improves the overall value of the soft constraint. This process continues for a set of iterations. The final solution not only satisfies the hard constraints but also optimizes the timetable based on the soft constraints. This method inspired by simulated annealing helps to fine-tune the initial solution and improve the quality of the timetable generated.

3.4 Personalised Timetable Function

A timetable function is designed to generate a personalized timetable for a specific student according to his student ID in a given semester. The function checks the courses the student is enrolled for and populates the timetable with only those courses and the room address in their respective time slots. This function is very helpful for students and gives them an overview of their weekly timetable, according to their course enrollments.

4 Numerical Experiments

The implementation of the proposed solution was conducted in Python using the OR-Tools CP Solver. To evaluate the model, multiple data sets were considered, but some did not

have the information required while some were too large to run in a reasonable amount of time. The analysis was focused on the data set from the University of Brasilia that was obtained online containing data from both semesters of the 2019 academic year. It included the student_id and each course_id for the courses they were enrolled in, and the semester in which the course would be taught. Each course_id had unique course names and the room_address for the rooms present in the university were also extracted. A snippet of the data is shown in Figure 3.

To ensure the quality of the data set some data pre-processing was done to get the required data. Rows containing 'Missing info' in the 'room_address' column were filtered out and only certain room_address were considered. A subset of the data was selected since the data was too large and would take very high computational time. The data was then split according to the semester and the proposed approach was applied on the data. There were 1123 unique courses in Semester 1 and 976 courses in Semester 2 which we have taken into consideration.

The generated timetable follows the assumption of a uniform lecture duration and schedules two lessons per week for each course. The schedule encompasses five working days, from Monday to Friday, with time slots divided into 2-hour periods between 8:00 AM and 6:00 PM. The timetable satisfies constraints related to student, room, and time slot allocations and adheres to the defined constraints, ensuring that each student has at most one lecture at any given time. This is achieved by avoiding clashes between time slots for courses attended by the same student. Similarly, room-time clashes are mitigated by allocating different rooms for courses that share the same time slot.

To provide a more granular view, we implemented a function that allows us to examine the timetables of individual students. This function enables us to check the specific courses, timings, and rooms assigned to each student. An example of a student-specific timetable for both semesters is as shown in Figure 4.

To evaluate the model's performance, we compare the execution time to get the feasible solution and the execution time to get an optimal solution in both semesters. The initial phase of our model involves satisfying the hard constraints to derive a feasible solution. This execution time provides us insight into the rapid deployment of a functional timetable. After a solution is obtained, the optimization process helps us enhance the timetable by addressing the soft constraints to create an optimal solution. By comparing the execution times for both solutions, we gain an understanding into the trade-off between speed and optimality in our timetabling approach. We can see that the execution time for the optimal solution is quite high as compared to the time taken for obtaining the feasible solution and this is due to the iterative function of the optimization process. The soft constraints are applied and the timetable is improved.

Semester	**Feasible Solution**	**Optimal Solution**
Semester 1	6.869 s	50.42 s
Semester 2	1.733 s	34.56 s

Figure 1: Time Taken for Feasible and Optimal Solutions

Figure 2 illustrates the percentage of occupied rooms throughout the week in the solution before optimization. A higher room occupancy is observed at the beginning of the week, showing the enforcement of hard constraints in the initial phase. In contrast to this Figure 4 shows the corresponding percentages in the optimal solution after optimization. Here, the allocation of rooms exhibits improvement, aligning with student preferences. The optimization process focuses on the soft constraints and contributes to a more balanced timetable. This gives us insight into the efficiency of our approach through the optimization phase which enhances the final timetable produced.

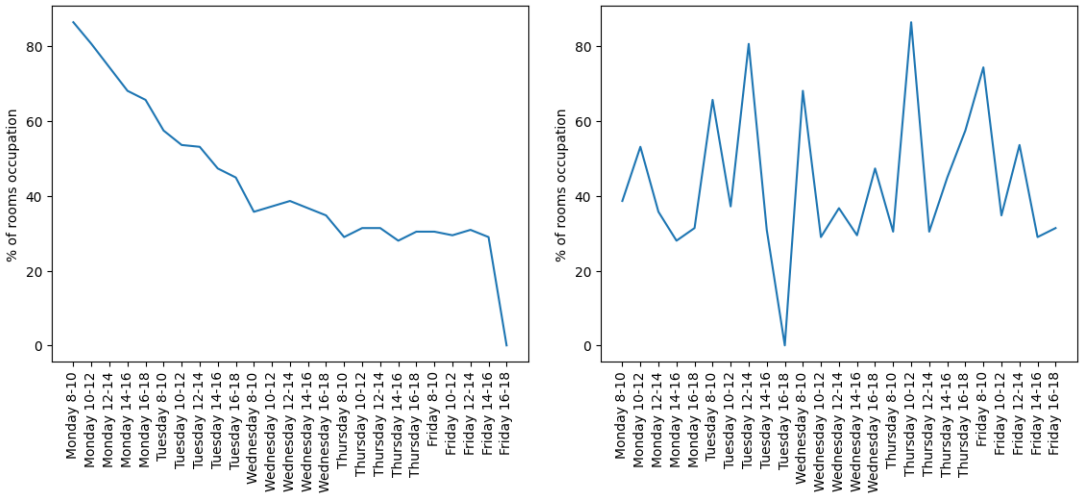

Figure 2: Percentage of Rooms Occupied before and after Optimization

5 Conclusions

This report explored the constraint programming approach to solve the university timetabling problem of the University of Brasilia. It presents an efficient solution using the principles of constraint-based programming. By utilising the Google OR-Tools CP Solver, we formulate the timetabling problem as a Constraint Satisfaction Problem (CSP), addressing both hard and soft constraints with precision. The analysis also considers the trade-off between speed and optimality. Feasible solutions are rapidly obtained, meeting the basic requirements for an institution. The subsequent optimization process refines timetables, providing valuable insights into the execution times for both solutions.

In conclusion, the presented solution not only meets the fundamental constraints of university timetabling but also establishes a foundation for future optimization. The integration of hybrid models and further exploration of advanced optimization techniques could lead to even more refined timetabling solutions. The study provides a valuable contribution to the field, offering a logical approach to timetabling that balances efficiency and optimization.

There is a massive scope for future work in university timetabling done by a CSP solver. Adaptation of more sophisticated models and constraints is essential to capture the intricate

nature of university timetabling, including nuanced factors like lecturer preferences, room characteristics, and specific course requirements.

As we look forward, the exploration of *hybrid models* is a promising direction for advancing the field of university timetabling. By combining the strengths of CSP with other approaches such as meta-heuristic algorithms and simulated annealing, we could refine the performance of our timetabling model. We could also integrate mathematical programming techniques, such as linear programming to increase precision of the model. By adopting these hybrid models, we can aim to create efficient timetabling systems that cater to the evolving needs of educational institutions.

Another direction future research can explore could explore is *real-time adaptive timetabling systems* that can promptly respond to dynamic changes and unforeseen events such as lecture cancellations, health emergencies, etc.

We could also develop a *user-interactive interface for timetabling*, to facilitate teacher-student interaction and provide insights into the impact of scheduling decisions. This would empower administrators and faculty to actively participate in the timetabling process.

One more interesting area to explore would be *Machine Learning integration* for predictive analysis, pattern recognition, and intelligent decision-making in university timetabling.

5.1 Limitations

While using a CSP for timetabling is a powerful technique, there are certain limitations attached to it while solving complex university timetabling problems. There are also certain bottlenecks that are observed due to assumptions taken for the algorithm.

A few drawbacks of the proposed solution considered are:

1. It is assumed that all lectures are of the same duration (2 hours). This might not be the case in many university timetabling problems, however as an initial benchmark model for simplicity, we did not take that into account.

2. We do not have a constraint defined for professors, hence we are not checking the availability of professors to take the lectures. This is also a useful constraint in multiple timetabling problems, but is beyond the scope of this report.

3. The model does not check whether a particular course is assigned to the same room or a different one each time. It is unlikely, but possible, that a course is assigned to a different room in different simulations of the model.

4. Adaptation to dynamic changes in university environments, such as sudden room changes or fluctuations in student enrollments, remains a challenge. Future research should focus on developing real-time adaptive timetabling systems.

5. The presented CSP solver exhibits notable performance on moderate-sized timetabling instances. However, challenges arise when scaling the system to handle large university schedules, prompting further investigation into optimizing the solver's scalability.

References

A. Bashab et al. Optimization techniques in university timetabling problem: constraints, methodologies, benchmarks, and open issues. *Computers, Materials & Continua*, 74(3): 6461–6484, 2023.

E. K. Burke and S. Petrović. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.

M. Carter and G. Laporte. Recent developments in practical examination timetabling. *In Lecture Notes in Computer Science*, pages 1–21, 1996.

M.C. Chen et al. A survey of university course timetabling problem: Perspectives, trends and opportunities. *IEEE Access*, 9:106515–106529, 2021.

S. B. Deris et al. University timetabling by constraint-based reasoning: A case study. *The Journal of the Operational Research Society*, 48(12):1178–1190, 1997.

L. T. G. Merlot et al. A hybrid algorithm for the examination timetabling problem. *In Lecture Notes in Computer Science*, page 207–231, 2003.

A. Wren. Scheduling, timetabling and rostering — a special relationship? *In Lecture Notes in Computer Science*, page 46–75, 1996.

Appendix

student_id	year_semester	course_id	course	weekday	start_time	end_time	room_address
100115	20191	192015	Primary Teaching	Monday	8:00	11:40	FE-5 - SALA DE AULA 04
100140	20191	139190	Social History And General Policy	Tuesday	16:00	17:50	SALA PAT AT 045
100140	20191	134911	Political Sociology	Wednesday	14:00	15:50	SALA PAT AT 061
100140	20191	139190	Social History And General Policy	Thursday	16:00	17:50	SALA PAT AT 045

Figure 3: Benchmark Data

Link to Dataset Used

index	Monday	Tuesday	Wednesday	Thursday	Friday
8-10	Information Systems,SALA BSA SUL B2 41/10	*	Information Systems,SALA PAT AT 125	*	The Object Oriented Modeling,SALA UAC A1 42/30
10-12	*	*	*	Electronics,SALA UAC AT 42/50	*
12-14	Electronics,SALA BSA SUL B2 54/10	*	The Object Oriented Modeling,SALA PAT AT 117	*	*
14-16	*	*	*	*	*
16-18	*	*	*	Operational Systems,SALA DE AULA MAT1	Operational Systems,SALA ICC BT 012/62

SEMESTER 1

index	Monday	Tuesday	Wednesday	Thursday	Friday
8-10	Human Computer Interaction,SALA BSA SUL B1 11/13	*	*	Environment Of Sciences,SALA PJC BT 021	*
10-12	Human Computer Interaction,SALA DE AULA MAT1	*	*	Environment Of Sciences,SALA UAC A1 42/40	*
12-14	*	Operational Systems,SALA ENM DT 43/15	*	Operational Systems,SALA BSA SUL A1 16/22	*
14-16	*	*	*	*	*
16-18	*	*	*	*	*

SEMESTER 2

Figure 4: Student Specific Timetable for both Semesters