Analysis and Solution of Algorithms for the University Course Timetabling Problem (UCTTP)

Raimel Daniel Romaguera Puig
Group C412

Abstract

The University Course Timetabling Problem (UCTTP) is an NP-Complete combinatorial optimization problem with great practical relevance in academic management. This work addresses the problem from a dual perspective: theoretical and practical. First, a formal proof of its NP-Completeness is presented via a reduction from the graph coloring problem, and the nonexistence of approximation algorithms with strict guarantees for its general variant is analyzed. These theoretical results justify the need for approximate approaches. Consequently, heuristic and metaheuristic methods for its practical resolution are studied and applied. Graph coloring algorithms (Greedy, DSatur, RLF) are evaluated on random graphs, where DSatur demonstrated superior performance. Likewise, the metaheuristics GRASP and Genetic Algorithm are implemented and compared on representative UCTTP instances. Experimental results indicate that both are effective, with GRASP showing notable robustness and speed, while the Genetic Algorithm can achieve lower-cost solutions in some cases, albeit with longer execution time. The research confirms the inherent computational complexity of UCTTP and validates the use of heuristic and metaheuristic methods as viable tools for generating feasible and high-quality timetables in real educational environments.

1. Introduction

The University Course Timetabling Problem (UCTTP) is a fundamental combinatorial optimization problem in academic management, which involves assigning courses [2], [3], [4], professors, and students to available time slots and classrooms, while respecting a set of hard (mandatory) and soft (desirable) constraints. Its practical relevance is undeniable, as a well-designed timetable optimizes resource usage, improves the educational experience, and reduces operational conflicts in higher education institutions.

However, UCTTP is classified as an NP-Complete problem [3], meaning there is no exact algorithm capable of solving it in polynomial time for realistically sized instances. This computational complexity has motivated the search for alternative approaches, such as heuristic [1], [4], [8] and metaheuristic methods [2], [3], [6], which offer feasible and high-quality solutions in reasonable times, albeit without optimality guarantees.

In this project, UCTTP is addressed from a comprehensive theoretical-practical perspective. First, a formal proof of its NP-Completeness is presented, based on a reduction from the graph coloring problem, thus establishing its inherent computational difficulty. Furthermore, the nonexistence of approximation algorithms with strict guarantees for certain variants of the problem [5] is explored, reinforcing the need for approximate and heuristic methods.

In response to this complexity, heuristic approaches are studied and applied, particularly those based on graph theory and coloring, as well as metaheuristics like GRASP and genetic algorithms, which allow for efficient exploration of the solution space. Finally, an experimental analysis evaluating the performance of these algorithms on representative problem instances is included, using metrics such as feasibility, solution quality, and execution time.

This research not only contributes to the theoretical understanding of UCTTP but also provides practical tools for its resolution, balancing academic rigor and applicability in real-world contexts.

2. Problem Definition

The University Course Timetabling Problem (UCTTP) is a combinatorial optimization problem whose main objective is to assign a set of academic events (courses, classes, labs) to predefined time slots and classrooms, fulfilling a set of constraints that guarantee the feasibility and quality of the generated timetable.

2.1 Main Components

- **Events (E):** Activities to schedule (classes, exams, tutorials).
- **Time Slots (T):** Available discrete time intervals.
- **Resources (R):** Classrooms, professors, necessary equipment.
- **People (P):** Involved students and professors.
- **Constraints (C):** Rules classified into hard (mandatory) and soft (desirable).

2.2 Types of Constraints

- **Hard Constraints:** Must be fully satisfied. Examples of these constraints are:
  - A professor cannot teach two classes simultaneously.

  - A classroom cannot host more than one event at a time.
  - Classroom capacity must be sufficient for the students.
- **Soft Constraints:** Their compliance improves quality. Among them are:
  - Professor and student schedule preferences.
  - Uniform distribution of classes throughout the week.
  - Avoiding consecutive schedules for the same student.

2.3 Variants of UCTTP

- **Curriculum-Based (CB-UCTTP):** [3] Based on fixed study plans (curricula) that group courses with common students. The goal is to avoid overlaps between courses of the same curriculum.
- **Post-Enrolment (PE-UCTTP):** [3] Built after enrollment, using real enrollment lists to minimize individual conflicts.

2.4 General Objective

Find a complete assignment that:

1. Satisfies all hard constraints (feasible solution).
2. Minimizes the violation of soft constraints (quality solution).

3. Mathematical Model of the Problem

A mathematical model for the **CB-UCTTP** variant is presented, formulated as a binary integer programming problem.

3.1 Sets

- C: Set of courses.
- Q: Set of curricula. Each curriculum $q \in Q$ is a subset of courses $C_q \subseteq C$.
- P: Set of available time periods.
- R: Set of classrooms.
- T: Set of professors.
- F: Set of preferences (soft constraints).

3.2 Parameters

- $a(c)$: Number of students enrolled in course $c \in C$.
- $\text{cap}(r)$: Capacity (students) of classroom $r \in R$.
- $C(t)$: Set of courses for which professor $t \in T$ is qualified.
- $\text{disp}_t(p) \in \{0, 1\}$: Availability of professor t in period $p \in P$.
- $\text{disp}_r(p) \in \{0, 1\}$: Availability of classroom r in period $p \in P$.

3.3 Decision Variables

For each feasible combination of course, period, classroom, and professor, a binary assignment variable is defined:

- $x_{c,p,r,t} \in \{0, 1\}$, defined for all $c \in C$, $p \in P$, $r \in R$, $t \in T$ satisfying $\text{cap}(r) \geq a(c)$ and $c \in C(t)$.

The variable takes the value 1 if course c is assigned to period p, in classroom r, and is taught by professor t; and 0 otherwise. The conditions $\text{cap}(r) \geq a(c)$ and $c \in C(t)$ guarantee that only combinations feasible in terms of classroom capacity and professor qualification are considered.

3.4 Hard Constraints

The following constraints must be satisfied in any feasible solution of the problem:

$$\sum_{p \in P} \sum_{r \in R} \sum_{t \in T} x_{c,p,r,t} = 1, \qquad \forall c \in C, \tag{1}$$

$$\sum_{c \in C} \sum_{r \in R} x_{c,p,r,t} \leq 1, \qquad \forall t \in T, \forall p \in P, \tag{2}$$

$$\sum_{c \in C_q} \sum_{r \in R} \sum_{t \in T} x_{c,p,r,t} \leq 1, \qquad \forall q \in Q, \forall p \in P, \tag{3}$$

$$\sum_{c \in C} \sum_{t \in T} x_{c,p,r,t} \leq 1, \qquad \forall r \in R, \forall p \in P, \tag{4}$$

$$x_{c,p,r,t} \leq \text{disp}_t(p), \qquad \forall c \in C, \forall p \in P, \forall r \in R, \forall t \in T, \tag{5}$$

$$x_{c,p,r,t} \leq \text{disp}_r(p), \qquad \forall c \in C, \forall p \in P, \forall r \in R, \forall t \in T. \tag{6}$$

Explanation of constraints:

- (1): Each course must be assigned exactly to one period, one classroom, and one professor.
- (2): A professor cannot teach more than one course simultaneously in the same period.
- (3): Courses belonging to the same curriculum cannot overlap in the same period.
- (4): A classroom cannot host more than one course per period.
- (5): A professor can only be assigned to a course in periods when they are available.
- (6): A classroom can only be used for a course in periods when it is available.

The constraints for classroom capacity and professor qualification are incorporated implicitly in the variable definition domain, so they do not require additional explicit equations.

3.5 Objective Function

Maximize the fulfillment of preferences:

$$\max \sum_{f \in F} x_{c_f, p_f, r_f, t_f} \tag{7}$$

where each preference $f \in F$ is represented as a tuple (c_f, p_f, r_f, t_f) indicating a desired assignment.

4. Computational Complexity: Proof of NP-Completeness

The University Course Timetabling Problem (UCTTP) belongs to the class of NP-Complete problems [2], [3] [4]. This classification implies that no algorithm is known that solves it exactly in polynomial time for all instances. In this section, the NP-Completeness of UCTTP is formally demonstrated via a reduction from the graph coloring problem, which in turn is proven NP-Complete from the 3-SAT problem.

4.1 Preliminaries of Computational Complexity

- **Class P:** Decision problems solvable in polynomial time by a deterministic Turing machine.
- **Class NP:** Decision problems whose solutions can be verified in polynomial time.
- **NP-Complete Problem:** A problem is in this class if it belongs to NP and every problem in NP can be reduced to it in polynomial time (it is NP-Hard).

4.2 The Graph Coloring Problem is NP-Complete

The K-Coloring problem consists of determining whether an undirected graph $G = (V, E)$ can be colored with at most K colors, such that adjacent vertices have different colors.

4.2.1 K-Coloring is in NP

Given a graph G and an integer K, a certificate is a color assignment. Verifying that each edge connects vertices of different colors takes $O(|E|)$ time, so the problem is in NP.

4.2.2 Reduction from 3-SAT to 3-Coloring (NP-Hardness)

To demonstrate that 3-Coloring is NP-Hard, we reduce from 3-SAT, a known NP-Complete problem. Given a Boolean formula ϕ in CNF with n variables and m clauses, we construct a graph G_ϕ that is 3-colorable if and only if ϕ is satisfiable.

Construction of G_ϕ:

1. **Truth vertices:** Three vertices T (true), F (false), and B (base) are created, connected in a triangle.
2. **Variables:** For each variable x_i, vertices x_i and $\neg x_i$ are created. Edges $(x_i, \neg x_i)$, (x_i, B), and $(\neg x_i, B)$ are added.
3. **Clauses:** For each clause $C_j = (l_{j1} \vee l_{j2} \vee l_{j3})$, a gadget of 6 vertices $\{u_j, v_j, w_j, a_j, b_j, c_j\}$ is added with edges connecting upper/lower triangles, literals, T, and among themselves.

Proof of equivalence:

- If ϕ is satisfiable, there exists a truth assignment that allows coloring G_ϕ with 3 colors.
- If G_ϕ is 3-colorable, the coloring induces a truth assignment that satisfies ϕ.

Since 3-SAT is NP-Hard, 3-Coloring is also. This proof extends to K-Coloring for $K \geq 3$.

4.3 The Course Timetabling (CT) Problem is NP-Complete

The decision version of CT, equivalent to UCTTP, is defined: given an instance with courses, periods, classrooms, and conflicts, determine if a feasible assignment exists.

4.3.1 CT is in NP

Given a proposed assignment, verifying all hard constraints (no overlap, capacity, availability) is done in polynomial time. Therefore, CT $\in$ NP.

4.3.2 Reduction from Graph Coloring to CT (NP-Hardness)

A polynomial reduction is constructed from the Graph Coloring (GC) problem to CT.

Given a GC instance: graph $G = (V, E)$ and integer p, a CT instance is built:

- **Courses:** One course K_i for each vertex $v_i \in V$, with one class.
- **Periods:** p periods (equivalent to colors).
- **Classrooms:** Unlimited capacity per period ($l_k = |V|$).
- **Conflicts:** For each edge $(v_i, v_j) \in E$, courses K_i and K_j are in conflict.
- **Additional constraints:** No preassignments or availability constraints.

Proof of equivalence:

- If G is p-colorable, the coloring defines a feasible assignment of courses to periods.
- If a feasible solution for CT exists, the assignment of periods to courses defines a valid p-coloring for G.

Therefore, CT is NP-Hard.

4.4 Conclusion of NP-Completeness

Given that:

1. The Course Timetabling (CT) decision problem is in NP, and
2. CT is NP-Hard (by reduction from GC, an NP-Complete problem),

it is concluded that the Course Timetabling Problem (and by extension UCTTP) is NP-Complete.

Knowing that this problem is NP-Complete implies that, under the assumption $P \neq NP$, there is no polynomial-time algorithm that optimally solves all instances of UCTTP. Furthermore, the inherent complexity validates the use of approximate methods (heuristics, metaheuristics, hybrids) to obtain quality practical solutions in reasonable times.

5. Nonexistence of Approximation Algorithms

The NP-Complete complexity of the University Course Timetabling Problem (UCTTP) not only implies the nonexistence of exact polynomial-time algorithms but also establishes strong limits on the possibility of designing approximation algorithms with strict theoretical guarantees. Using the PCP Theorem (Probabilistically Checkable Proofs) and other complexity results mentioned in [5], it is demonstrated that approximating the chromatic number of a graph (and by extension UCTTP) is even more difficult.

Theorem of Inapproximability of the Chromatic Number [5]: Assuming that $NP \not\subseteq ZPP$ (stronger than $P \neq NP$), there is no polynomial-time algorithm that approximates $\chi(G)$ within a factor $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$, where n is the number of vertices in the graph.

Interpretation: One cannot guarantee an algorithm that returns a coloring using at most $n^{1-\varepsilon} \cdot \chi(G)$ colors, unless $P = NP$. Since UCTTP directly reduces to the coloring problem, the same limit applies: **there is no polynomial-time approximation algorithm that guarantees a factor polynomially close to the optimum**.

The graph coloring problem has classical approximation algorithms, but their approximation factors are high. For the most general cases, the best known algorithms only achieve ratios of the form $n/\text{polilog}(n)$ [9], which represents a marginal improvement over the trivial approximation of n.

This limitation is fundamental: there is no PTAS (Polynomial-Time Approximation Scheme), much less an FPTAS, for the general coloring problem, unless $P = NP$ [9]. This approximation hardness directly translates to UCTTP, severely restricting the performance guarantee that practical algorithms can offer.

Finally, the nonexistence of approximation algorithms with strict guarantees for UCTTP under standard computational complexity assumptions ($P \neq NP$ and $NP \not\subseteq ZPP$) underscores its intrinsically difficult nature. This justifies the use of practical approaches like metaheuristics, hybrid algorithms, and machine learning-based methods, which, although lacking formal approximation bounds, have proven effective in generating high-quality timetables in real scenarios.

6. Resolution by Graph-Based Heuristics

The University Course Timetabling Problem (UCTTP) can be effectively modeled as a graph coloring problem [1], [8], given the structural equivalence between both. In this approach, events (courses, exams) are represented as vertices, conflicts between events (due to shared resources like professors, students, or classrooms) are modeled as edges, and available time intervals correspond to colors. Thus, assigning conflict-free schedules is equivalent to coloring the graph with the minimum possible number of colors.

6.1 Heuristic Graph Coloring Algorithms

Due to the NP-Completeness of the coloring problem, heuristic algorithms that find feasible solutions in reasonable time are used, although they do not guarantee optimality. Three key algorithms are described below.

6.1.1 Greedy Algorithm

The Greedy algorithm [8] processes vertices in a given order (e.g., random) and assigns to each vertex the first available color that does not cause conflicts with its already colored neighbors. Although its simplicity allows obtaining solutions quickly, the solution quality critically depends on the processing order. In the worst case (complete graph K_n), its time complexity is $O(n^2)$.

6.1.2 DSatur Algorithm (Degree Saturation)

DSatur [1] improves upon Greedy by using a dynamic ordering of vertices based on their saturation degree, defined as the number of different colors assigned to their neighbors. In each step, the vertex with the highest saturation degree (and in case of a tie, the highest degree) is selected for coloring. This strategy prioritizes the most constrained vertices, reducing the number of colors needed. DSatur has complexity $O(n^2)$ and is exact for bipartite graphs, cycles, and circular graphs.

6.1.3 RLF Algorithm (Recursive Largest First)

RLF [8] operates by coloring maximum independent sets in each iteration, assigning one color per iteration. Instead of coloring vertices one by one, it recursively constructs the largest possible independent set, removes it from the graph, and repeats the process with the

remaining subgraph. Although its complexity is higher ($O(n^3)$), it tends to produce higher quality solutions by exploring the solution space in a more structured way.

6.2 Experimental Evaluation on Random Graphs

To compare the performance of these algorithms, 50 random graphs were generated for each size $n \in \{20, 40, 60, 80, 100\}$ using the Erdös-Rényi model $G_{n,p}$ with $p = 0.5$ [8]. The results (average μ and standard deviation σ of the number of colors) are summarized in Table 1.

Cuadro 1: Average results and standard deviation of the number of colors for each heuristic.

n	Greedy	DSatur	RLF
20	7.12 ± 0.65	6.04 ± 0.53	6.90 ± 0.64
40	11.24 ± 0.81	9.64 ± 0.71	11.42 ± 0.60
60	14.82 ± 0.93	12.62 ± 0.72	15.50 ± 0.73
80	18.28 ± 0.92	15.70 ± 0.81	19.38 ± 0.75
100	21.12 ± 0.93	18.24 ± 0.76	23.08 ± 0.87

6.2.1 Analysis of Results

In the experiments performed, the DSatur algorithm managed to outperform the Greedy and RLF algorithms by producing higher quality solutions.

7. Resolution using Metaheuristics

Given the NP-Completeness of UCTTP, metaheuristics emerge as powerful tools to find high-quality solutions in reasonable times, albeit without optimality guarantees [7]. These techniques balance exploration (diversification) and exploitation (intensification) of the search space, effectively handling complex constraints.

7.1 Metaheuristics Applied to UCTTP

Among the most used metaheuristics for UCTP are Genetic Algorithms (GA) ([2] [3]) and GRASP (Greedy Randomized Adaptive Search Procedure) ([2] [3] [6]), both experimentally evaluated in this work.

7.1.1 Genetic Algorithm (GA)

Genetic algorithms simulate the process of natural evolution. They maintain a population of candidate solutions (individuals) that are combined and mutated over generations. Selection favors individuals with higher *fitness* (solution quality), allowing the best characteristics to propagate. The main operators are:

- **Selection**: Choice of parents based on fitness (e.g., roulette, tournament).
- **Crossover**: Combination of two parents to generate offspring.
- **Mutation**: Random alteration of genes to maintain diversity.

Although GAs are robust and applicable to various problems, they require careful parameter tuning (population size, crossover and mutation rates) and may prematurely converge to local optima.

7.1.2 GRASP (Greedy Randomized Adaptive Search Procedure)

[6] GRASP is an iterative metaheuristic with two phases:

1. **Greedy Randomized Construction**: Generates a feasible solution through a greedy algorithm with controlled randomness (using a Restricted Candidate List, RCL).
2. **Local Search**: Improves the constructed solution by exploiting its neighborhood until a local optimum is reached.

GRASP stands out for its simplicity, ease of implementation, and ability to produce high-quality solutions with minimal parameter intervention. Advanced strategies like Reactive GRASP dynamically adjust the balance between greediness and randomness based on the performance of previous iterations.

7.2 Experimental Evaluation on UCTTP Instances

The GRASP and Genetic Algorithm (GA) algorithms were evaluated on a set of five representative UCTTP problem instances, selected for their distinctive characteristics: size (small, medium, large), classroom capacity/student ratio, and conflict levels (high curriculum conflicts and low resource availability).

Each experiment was repeated 20 times to ensure statistical robustness. The metrics evaluated included feasibility (compliance with all hard constraints), solution cost (value of the objective function penalizing soft constraints), and execution time in seconds.

7.2.1 Key Results

The average results for solution cost and execution time for each instance-algorithm combination are presented in Table **??**.

- **Instance 1:** High Conflict
- **Instance 2:** High Availability
- **Instance 3:** Large (12 Courses)
- **Instance 4:** Medium (10 Courses)
- **Instance 5:** Small (8 Courses)

Both metaheuristics proved effective for solving UCTTP, offering an adequate balance between solution quality and execution time. GRASP showed particularly robust and consistent performance, with generally lower execution times and reduced variability

Instance	Algorithm	Average	Time
1	GRASP	0.0	7.7
1	GA	0.0	9.5
2	GRASP	10.4	2.3
2	GA	7.5	3.6
3	GRASP	57.6	13.2
3	GA	52.7	16.9
4	GRASP	32.5	5.2
4	GA	39.6	9.7
5	GRASP	31.8	3.2
5	GA	27.3	5.1

Cuadro 2: Comparative results of GRASP and the Genetic Algorithm on the test instances. The average cost reflects the penalty for violation of soft constraints; a value of 0 indicates a feasible and optimal solution in terms of these constraints.

between repetitions in most cases. On the other hand, the Genetic Algorithm required more computation time but achieved better average costs in some instances, such as the small instance (5) and the high availability instance (2).

8. Conclusions

This work comprehensively addressed the University Course Timetabling Problem (UCTTP), confirming its NP-Complete nature and establishing limits on the existence of approximation algorithms with strict guarantees. This inherent complexity justified the exploration of heuristic and metaheuristic methods in practice. Experiments demonstrated that, among graph coloring algorithms, DSatur surpasses the Greedy and RLF heuristics in quality. Likewise, the evaluation of the GRASP and Genetic Algorithm metaheuristics on representative instances revealed that both are effective tools for solving UCTTP, with GRASP showing robust and efficient performance in time.

9. Recommendations

1. **Deepen Advanced and Hybrid Metaheuristics:** It is recommended to investigate the design and implementation of hybrid metaheuristics that combine the strengths of GRASP (fast construction and local search) with those of Genetic Algorithms (population exploration). Exploring other techniques like Simulated Annealing, Tabu Search, or ant colony algorithms is also suggested to compare their performance on a broader set of benchmark instances.

2. **Incorporate Real-World Constraints and Scalability:** For effective technology transfer, it is necessary to extend the models and algorithms to incorporate a richer set of constraints. Likewise, the performance of the proposed algorithms should be evaluated on large-scale instances, typical of sizable universities, optimizing for execution time and memory usage.

3. **Develop a Modular Software Framework:** The development of a modular and configurable software platform for timetable generation is recommended. This tool would allow academic institutions to define their own constraints (hard and soft) and parameterize the different algorithms (heuristic and metaheuristic) studied here, facilitating practical adoption and continuous experimentation.

4. **Explore Machine Learning (ML) Approaches:** Given the combinatorial nature and the large amount of historical data available in many institutions, it is proposed to explore the application of Machine Learning techniques, such as Reinforcement Learning or Neural Networks, to guide the search for solutions, learn from previous successful timetables, or predict conflicts, thus enhancing traditional metaheuristics.

Referencias

[1] A Study on Course Timetable Scheduling using Graph Coloring Approach. International Journal of Computational and Applied Mathematics. ISSN 1819-4966 Volume 12, Number 2 (2017), pp. 469-485

[2] A Survey of Approaches for University Course Timetabling Problem: Perspectives, Trends and Opportunities. Computers & Industrial Engineering. Volume 86, August 2015, Pages 43-59

[3] A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. in IEEE Access, vol. 9, pp. 106515-106529, 2021, doi: 10.1109/ACCESS.2021.3100613.

[4] Constraint-based Timetabling. Ph.D. Thesis. Tomás Muller. Charles University in Prague. Faculty of Mathematics and Physics

[5] Zero Knowledge and the Chromatic Number. Journal of Computer and System Sciences. Volume 57, Issue 2, October 1998, Pages 187-199

[6] GRASP: Greedy Randomized Adaptive Search Procedures. Mauricio G. C. Resende, Ricardo M. A. Silva. February 17, 2009. Revised September 9, 2009

[7] Metaheuristics from Design to Implementation. El-Ghazali Talbi. University of Lille -CNRS - INRIA

[8] El Problema de Coloracion de Grafos. Universidad de Santiago de Compostela. Sergio Pena Seijas

[9] Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. THEORY OF COMPUTING, Volume 3 (2007), pp. 103–128