International Journal of Applied Optimization Studies
IJAOS
İSÜ İSTİNYE ÜNİVERSİTESİ İSTANBUL

A Two-Phase Local Search Approach for Optimizing University Course Scheduling under Academic Constraints

Kourosh Mokhtari[1,*], Fariba Goodarzian[2], Seyydeh Atefeh Mousavi Abandansar[3]

[1]Microelectronics Institute of Sevilla, 41092 Seville, Spain
[2]Edinburgh Business School (EBS) and School of Social Sciences, Heriot-Watt University, Riccarton, Currie EH14 4AS
[3]Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran
*E-mail (corresponding author): kuroshmokhtari@yahoo.com

ABSTRACT

Efficiently scheduling university courses while accommodating a plethora of academic constraints poses a substantial challenge for educational institutions. This paper addresses the intricacies of course scheduling, acknowledging the presence of multiple restrictions, encompassing class and faculty constraints. Existing methodologies have proposed solutions to this complex optimization problem. In this research, we introduce a novel two-phase local search approach designed to tackle the university course scheduling problem. Our approach unfolds in two distinct phases. Initially, we generate a feasible solution to the scheduling problem. Subsequently, in the second phase, we enhance the solution's quality through the application of refined optimization techniques. We have implemented this method across diverse datasets and meticulously scrutinized the outcomes. Our empirical results underscore the efficacy of the proposed two-step approach in delivering high-quality solutions to the intricate problem of university course scheduling.

Paper type: *Research Article*

Received 2022-03-04
Revised 2022-06-02
Accepted 2022-06-05

Keywords:
Innovative method;
Scheduling problem;
Optimization;
Academic constraints.

1. Introduction

Today, scheduling is one of the essential necessities of human life. Generally, a schedule is a plan for performing work, specifying the allotted time for each part and the person performing the task. A timetable is a structured schedule of events with the times at which they occur. Looking around us, we see that many activities require scheduling, the result of which is shown by a timetable. In general, two elements must be considered for a timetable: 1) two events do not occur simultaneously in the same place, and 2) the sufficiency of the available resources must be considered for all events at any time. In

simple terms, the problem must be feasible. Nevertheless, these two elements are general terms, and there are often several constraints in actual problems. For instance, some events should occur before other events or some events require a minimum time between the first occurrence and the start of the second.

Solving the problem of scheduling university courses is a difficult task because of the magnitude of the issue and the different structures and natures of each problem. Various techniques have been proposed to evaluate and solve this issue, including methods based on graph coloring and integer programming. However, heuristic and metaheuristic approaches have been more emphasized in recent decades. For example, Doğan and Yurtsal (2021) solved the problem using a genetic algorithm. They proposed a genetic algorithm with a biased selection strategy for decoding chromosomes, improving the proportionality of the population. In another study, Lach and Lübbecke (2012) presented an integer programming model and used a two-stage approach based on three parameters: time, classroom, and course. Lü and Hao (2012) proposed an adaptive tabu search algorithm for solving curriculum-based course timetabling problems. They also evaluated four neighborhoods based on three criteria to assess the percentage of neighborhood improvement, enhancement strength, and search steps. In the latest research, Rezaeipanah et al. (2021) developed a hybrid algorithm for the university course timetabling problem based on the genetic algorithm, incorporating some local search approaches. The results confirmed the effectiveness and superiority of the proposed algorithm in solving the course timetabling problem.

2. Problem Statement

A university course schedule is a tool used to organize various events, such as course sessions and teacher assignments, within specific time slots and classrooms, while adhering to a set of constraints to achieve specific objectives. Carter and Laporte (1998) have conceptualized academic course timetabling as a complex allocation problem, involving the assignment of students and teachers to lectures, classrooms, and time slots in a manner that eliminates conflicts among these elements.

In essence, university course timetabling problems can be categorized into two main constraint types: hard and soft. In order to generate a high-quality schedule, it is imperative that the problem is inherently feasible and that any violations of soft constraints are kept to a minimum. Some examples of hard constraints typically addressed in these problems encompass the following:

1. Ensuring that only one course is assigned to a room at any given time.
2. Limiting the allocation of a teacher to just one course during a specific time slot.
3. Verifying that classrooms have sufficient capacity to accommodate the designated group of students.
4. Distributing lectures across different time slots throughout the day.

On the other hand, several soft constraints are commonly considered when tackling these problems:

1. Distributing intensive courses for both students and teachers evenly throughout the week.
2. Avoiding scenarios where students have only one lecture on a given day.
3. Taking into account all tabu (restricted) time slots and pre-allocated periods for teachers, students, and classrooms.

In summary, university course timetabling involves the intricate task of creating schedules that adhere to a range of constraints, both rigid and flexible, with the ultimate aim of achieving a high-quality outcome.

2.1. Modeling the First Phase of Scheduling University Courses

The objective of the first phase is to minimize the violation of hard constraints. A workable solution can be achieved when there is zero violation of hard constraints.

$$Min \sum (s_{2cdtu} + s_{cd\tau d} + s_{lcdtu}) + \sum_{x_i,j \epsilon X} f_1\,(x_{i,j}) \quad (1)$$

$$\sum_{r \in R} \sum_{\tau \le t < \tau_{d+1}} x_{crt} - s_{1cdtu} = n^c_{max} u_{cd}, c \in C, d \in D, t \in T, x_{crt} \in \{0,1\}, u_{cd} \in \{0,1\} \quad (2)$$

$$\sum_{r \in R} \sum_{\tau \le t < \tau_{d+1}} x_{crt} + s_{2cdtu} = n^c_{max} u_{cd}, c \in C, d \in D, t \in T, x_{crt} \in \{0,1\}, u_{cd} \in \{0,1\} \quad (3)$$

$$\sum_{r \in R} (x_{crt1} - x_{crt2} + x_{crt3}) + s_{cdtd} = 1, c \in C, d \in D, \tau_d \le t_1 < t_2 < t_3 < \tau_{d+1} \quad (4)$$

$$\forall x_{i,j}, x_{i,k} \in X, x_{i,j} = c_u, x_{i,k} = c_v : \left(\forall_{c_{rq}}, c_u \notin c_{rq} \vee c_v \notin c_{rq}\right) \wedge (t_{cu} \neq t_{cv}) \quad (5)$$

$$\sum_{r \in R} \sum_{t \in T} x_{crt} = n_c, c \in C \quad (6)$$

$$\forall x_{i,j} \in X, u_{i,j} = c_k, f_1(x_{i,j}) = \begin{cases} \alpha_1 . (std_k - capj), std_k > capj \\ 0 \end{cases} \quad (7)$$

$$\forall x_{i,j} \in X, x_{i,j} = c_k : uav_{i,k} = 0 \quad (8)$$

$$s_{2cdtu} \ge 0, s_{cdtu} \ge 0, s_{1cdtu} \ge 0 \quad (9)$$

In Equations (1)-(9), C is the set of courses, R is the set of rooms, T is the set of time slots, $x_{i,j}$ is the course label allocated to t time and j room, cap_j is the capacity of r_j room, and τ_d is the first time slot in the morning. In addition, u_{cd} will be equal to one if c course is allocated to d day; otherwise, it will be zero. Moreover, n_c shows the number of hours scheduled for each course in a week. In this study, Equation (1) shows the objective function of the first phase, which aims to minimize the level of violation of hard constraints. Equations (2) and (3) demonstrate the minimum and maximum lecture hours per day in accordance with the hard constraints of course scheduling. Equation (4) is related to the intensiveness of the events related to a lecture. Moreover, Equation (5) is related to the interference of courses due to having one teacher after being placed in a course group and scheduling in a time period. In other words, the mentioned Equation must be established in order to prevent the interference

of two courses scheduled in a period. Equation (6) leads to scheduling the number of sessions required for a course in a timetable. Equation (7) is related to holding courses in accordance with the capacity of rooms, and Equation (8) is related to holding courses in accordance with teacher availability.

2.2. Modeling the Second Phase of University Courses Scheduling

The objective function of the second phase is expressed based on a definition presented for solution quality. In general, solution quality is considered by using the penalties that are defined for each time slot and for each course. In this phase, the objective function is to minimize the penalty function.

$$penI = Min \sum_{c\in C}\sum_{t\in T} p_{ct} + \sum_{r\in R} x_{crt} \tag{10}$$

$$\sum x_{crt} \leq \varphi_{qd}, c \in C, q \in Q, d \in D, \tau_d \leq t < \tau_{d+1}, x_{crt} \in \{0,1\}, \varphi_{qd} \in \{0,1\} \tag{11}$$

$$\sum_{d\in D} \varphi_{qd} \leq k_q, q \in Q, \varphi_{qd} \in \{0,1\} \tag{12}$$

$$\sum x_{crt} = 1, \forall c \in F \tag{13}$$

$$\sum_{r\in R}\sum_{t\in v_i}\sum_{\tau=t-d_i+1} x_{c\tau r} = 0, \forall c \in C \tag{14}$$

In Equations (10)-(14), several key variables and their meanings are defined:

- Q represents the number of teachers.
- F denotes the number of allocated courses.
- k_q signifies the maximum number of teaching days for a given teacher.
- p_{ct} quantifies the penalty associated with the level of inconvenience experienced by teacher q when teaching course c.
- In this study, φ_{qd} equals one if teacher q delivers a lecture on day d; otherwise, it is zero.
- Additionally, x_{crt}equals one if course c is scheduled at time t in room r; otherwise, it is zero.

Equation (10) outlines the objective function for the second phase, aiming to minimize the penalty objective function. Furthermore, Equations (11) and (12) pertain to teachers' working hours over the course of a week. Finally, Equations (13) and (14) account for all forbidden time slots and periods that have been pre-allocated to teachers, students, and rooms.

3. Method Description

The problem of scheduling university courses is solved in two stages, in which local search methods are applied. In this study, we use iterated local search methods in both stages.

3.1. Iterated Local Search Method

Iterated local search stands out as one of the most impactful techniques within the realm of single solution methods. This method commences by generating an initial solution, and there are two approaches at our disposal for this purpose. The first approach involves creating a random initial solution, while the second employs a greedy strategy, intentionally generating an initial solution with a worse objective function value. Following the generation of the initial solution, iterated local search is then employed to refine and enhance this starting point. In the subsequent phase of each iteration, a chaos process is introduced to transform the improved solution into a chaotic solution. Subsequently, a local search process is applied to this chaotic solution, potentially replacing the previous solution as the reference point under specific conditions. This iterative process continues until a predefined termination criterion is met.

3.2. Proposed Method for Solving the Problem of Scheduling University Courses

The method proposed for addressing the challenge of scheduling university courses involves two distinct stages. In the initial stage, we strive to establish a feasible solution for the problem, and in the subsequent stage, we enhance the quality of this solution through appropriate techniques. While both stages employ methods based on local search, they diverge in their respective objective functions. To elaborate, the objective function in the first stage centers on minimizing violations of hard constraints. Consequently, the first stage concludes when there are no hard constraint violations. In contrast, the objective function in the second stage focuses on minimizing the breach of soft constraints. In this phase, we endeavor to refine the achievable solution obtained in the first stage by applying local search methods and defining effective and suitable neighborhoods.

3.3. Construction of a Workable Solution for the Problem of Scheduling University Courses

First, our approach introduces a heuristic algorithm to assign classrooms to various courses. Subsequently, the algorithm validates the corresponding timetable at each stage, contingent upon the successful classroom-to-course allocation by the algorithm.

In our problem solution, we allocate time slots to courses. Notably, the algorithm responsible for assigning classrooms to courses relies on a specialized data structure. This structure aids in determining the courses scheduled for a particular period. This data structure, henceforth referred to as the "time-based structure" throughout the remainder of this paper, is essentially a list assigned to each member of the list, designated for a specific time. It encompasses details of the lessons to be conducted during that particular time.

The algorithm initiates its assessment by considering a list of members, each embodying a unique time-based structure. As previously mentioned, each member allocates courses and events to a specific time slot. During the evaluation of each time-based structure, our algorithm systematically processes

courses scheduled for that time slot. It assesses whether the corresponding event for each course is designated for the first or second session. Subsequently, the algorithm proceeds to inspect other time slots associated with that session.

If a room has been previously reserved for any of the events within that session, the same room is allocated to the current event. However, the room is also added to the tabu list for that specific time slot, ensuring it remains unavailable for allocation to any other course. This process is meticulously executed for all courses within the same time slot. In the event that no room is present on the tabu list, a room not on the list will be allocated to the event, provided there are no room assignments for other events within the same session. The algorithm's viability hinges on the availability of a room not present on the tabu list for allocation to the event. Failure to locate such a room renders the algorithm unsuccessful. Conversely, the algorithm successfully allocates rooms to time slots without violating any hard constraints in the problem when suitable rooms are found.

3.3.1. Initial Solution Generation

A random approach is used to generate the initial solution. First, a time slot is randomly assigned to an event of a course. Afterward, the algorithm related to the room allocated, which was explained before, is called. If the algorithm successfully allocates the rooms, the time slot specified will be accepted. Otherwise, the time slot will not be accepted, and another time slot will be randomly generated and evaluated.

3.3.2. Local Search

First, let's delve into the description of neighborhoods employed during the initial phase before delving into the local search method based on these neighborhoods. In this stage, we examine two distinct neighborhoods designed to solve the university course scheduling problem.

The first neighborhood is rooted in substitution and encompasses N_1 and N2 neighborhoods. In N_1, we generate new neighborhoods for a given solution by substituting one of the existing time slots with another. Here, we carefully select an event responsible for violating the constraints related to intensiveness and replace its time slot with another.

Moving on to N_2, we meticulously explore all scenarios that lead to violations of intensiveness-related constraints, considering each individually. Unlike N_1, where the replacement time slot is chosen randomly, N_2 employs a more strategic approach. Time slots are selected with the specific aim of reducing the extent of intensiveness constraint violations.

Now, let's shift our focus to the N3 neighborhood, which is rooted in exchange. Here, we select two events from one or two courses (either one event from one course or two events from two different courses) and swap the time slots assigned to them. Subsequently, the local search method, utilizing the neighborhood displacement structure, is applied to the current solution. Following this step, a second exchange neighborhood is employed on the resultant solution. If the new solution does not surpass the

current one in quality, we revert to applying the first exchange neighborhood structure to the current solution, with the second exchange neighborhood structure being applied to the newly obtained solution.

3.3.3. Chaos

The first event of the courses' sessions is considered to cause chaos in the solution. Afterward, one of the events is selected randomly, and a valid time slot replaces the current time slot. If the first event of a course session changes, then the second exchange neighborhood, which was described before, considers the second event as an event that violates the intensiveness constraints and attempts to replace it with another valid event. Accordingly, all events of that session are recognized to be undoable and are tried to be changed by the algorithm.

3.3.4. Acceptance Benchmark

Following causing chaos in the current solution, the solution obtained replaces the current solution, and the process continues. In the first phase of the proposed algorithm, the cessation condition is to have zero violation of hard constraints. Figure 1 shows the first-step algorithm.

First-step algorithm
Step 1: Allocate rooms to courses.
Step 2: Do the following tasks as long as the secession condition is not established.
Step 2: Apply the iterated local search with the following steps:
2-1: Generate the s_j initial solution.
2-2: Apply local search with the following steps.
2-2-1: Apply N_3 neighborhood on s_j.
2-2-2: Apply N_2 neighborhood on s'_j.
2-2-3: If s'_1 is better than s_j, put $s_i = s'_i$.
2-2-3: Otherwise, apply N_1 on s_j.
2-2-4: Repeat step 2-2-2.
Step 3: Cause chaos in s_j and put $s_i = s'_i$.
Step 4: Assess the solution acceptance benchmark.

Fig. 1. First-step algorithm.

3.4. Improvement of the Solution Obtained from the First Stage of University Courses Scheduling

The neighborhood structures employed in the second stage differ significantly from those in the first stage. In the first stage, we utilized event-based neighborhoods, whereas in the second stage, our focus shifts to course sessions. Specifically, in this stage, we construct neighborhoods based on time slots and rooms. There are two distinct types of room-based neighborhood structures to consider. The first type is rooted in N4 room displacement, while the second is centered around N5 room exchange. Additionally, we have two types of time slot-based neighborhoods. The first, N6, involves displacing two different sessions from two courses, while the second, N7, pertains to transferring time slots

associated with one session to other configurations. In this section, we implement a local search-based algorithm that sequentially utilizes four distinct neighborhood structures. The illustration of the second-stage algorithm is shown in Figure 2.

Second-step algorithm
Step 1: Consider the s_j workable solution.
Step 2: Do the following tasks as long as the cessation condition is not established:
Step 2: Apply local search with the following steps.
2-1: Consider N_i neighborhood (i=4,5,6,7)
2-2: Find the best solution in N_i neighborhood.
2-3: If s'_1 is better than s_j, put $s_i = s'_i$.
2-4: Otherwise, repeat steps 1-2 to 2-3.
Step 3: Select the solution with the highest quality among the solutions.

Fig. 2. Second-step algorithm.

4. Numerical Results

The algorithms proposed for both the first and second steps have been implemented using the Matlab software. Numerical results were acquired using a computer equipped with the Windows 8.1 operating system and 8 GB of memory. This particular software environment was chosen due to its advantageous features; the C programming language, renowned for its speed, was utilized, providing the essential flexibility required to design and implement the proposed algorithm. The numerical results generated by the algorithm for solving the university course scheduling problem are presented in Table 1.

Table 1. Numerical results of the proposed algorithm.

Sample	Number of events	Number of teachers	Number of groups	Execution time of the first phase	A	B	C	D	E
1	120	20	5	4.15	6243	5631	5490	5403	4994
						-10%	-12%	-14%	-20%
2	210	57	9	17.53	12730	10057	9886	9771	9292
						-21%	-13%	-23%	-27%
3	278	57	11	47.73	15306	12212	11560	11232	10561
						-20%	-25%	-27%	-31%
4	384	60	13	74.18	20588	15632	15316	15303	13588
						-25%	-26%	-26%	-34%
5	430	61	15	158.43	23833	17506	16701	16539	14538
						-26%	-30%	-31%	-39%
6	490	62	16	178.32	26340	20518	19766	19435	18438
						-22%	-25%	-26%	-30%

A: Value of the objective function of the second phase; **B:** Value of the objective function after five minutes of running the second-step algorithm; **C:** Value of the objective function after 10 minutes of running the second-step algorithm; **D:** Value of the objective function after 15 minutes of running the second-step algorithm; **E:** Maximum reduction of the objective function after three hours of running the second-step algorithm.

Parameters were determined through a combination of experimental data and trial-and-error methods. Given that the algorithm proposed in this study is inherently randomized, it was executed for a given example ten times, with the resulting mean data recorded in Table 1.

According to Figure 3, an increase in time for the algorithm in the second phase will lead to obtaining a lower objective function value. A high objective function value is obtained when a five-second time slot is considered. Meanwhile, the lowest objective function value is obtained when a 15-second time slot is considered. Therefore, it could be concluded that the increase of the allowed time for objective function can improve its function and lead to obtaining lower objective function levels.

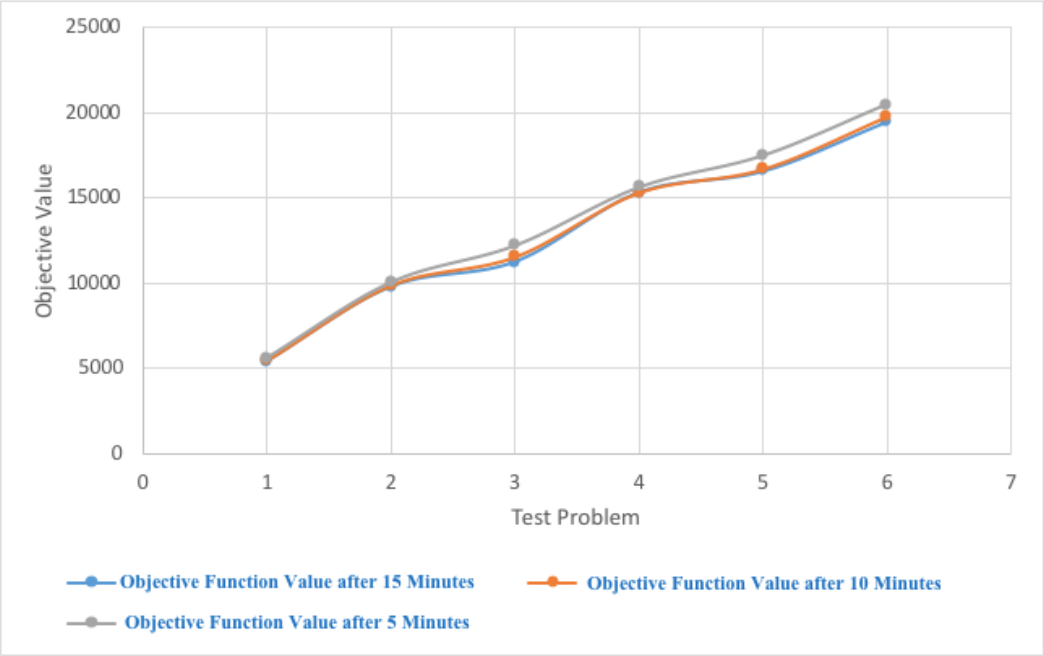


Fig. 3. Comparison of the value of the objective function in different algorithm implementation modes.

5. Conclusion

This study introduces a local search-based algorithm designed to address the complex challenge of scheduling university courses. The algorithm effectively tackles the problem through a two-stage approach. In the initial stage, we attain a workable solution through the skillful utilization of neighborhood structures. Subsequently, in the second stage, we refine and enhance this initial solution using the iterated local search method.

Notably, the application of iterated local search principles begins in the first stage, where the primary objective is to minimize violations of hard constraints. The first stage concludes when there are no hard constraint violations, signifying the achievement of a feasible solution. Here, we detail the algorithm's process for allocating rooms to courses, followed by an explanation of the local search techniques while leveraging the appropriate neighborhood structures of exchange and displacement.

The second stage deploys entirely different neighborhood structures compared to the first stage. While the initial phase primarily relies on events based solely on time slots, the second stage incorporates course sessions and events related to time slots and rooms within its neighborhood structures. Here, a utility function gauges the quality of the solution, and in each iteration, the algorithm works diligently to further refine the solution. Throughout, it diligently considers hard constraints, such as event intensiveness for each course and the capacity of lecture rooms. The problem is effectively addressed by accounting for hard constraint violations in the first stage and implementing a penalty function (penI) in the second stage.

In conclusion, the numerical results strongly affirm the efficacy of this method. This research commenced with a brief overview of the scheduling problem, followed by a concise summary of techniques used to tackle the intricate task of scheduling university courses. Subsequently, we delved into the specifics of the university course scheduling problem, presenting the relevant model. Lastly, we reported the outcomes and insights gleaned from solving this challenging problem.

References

Carter, M. W., & Laporte, G. (1998). Recent developments in practical course timetabling. In Practice and Theory of Automated Timetabling II: Second International Conference, PATAT'97 Toronto, Canada, August 20–22, 1997 Selected Papers 2 (pp. 3-19). Springer Berlin Heidelberg.

Lach, G., & Lübbecke, M. E. (2012). Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research, 194, 255-272.

Lü, Z., & Hao, J. K. (2010). Adaptive tabu search for course timetabling. European Journal of Operational Research, 200(1), 235-244.

DOĞAN, A., & YURTSAL, A. (2021). Developing a decision support system for exam scheduling problem using genetic algorithm. Eskişehir Technical University Journal of Science and Technology A-Applied Sciences and Engineering, 22(3), 274-289.

Rezaeipanah, A., Matoori, S. S., & Ahmadi, G. (2021). A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Applied Intelligence, 51(1), 467-492.