International Journal of Science Academic Research
Vol. 06, Issue 03, pp.9553-9555, March, 2025
Available online at http://www.scienceijsar.com

ISSN: 2582-6425

Research Article

A STACK BASED APPROACH FOR A TIME-SLOT SCHEDULING

[1, *]Ashish Seth, [2]Kirti Seth, [3]Sonam Gupta and [4]Pradeep Gupta

[1, 2]Inha University in Tashkent, Tashkent, Uzbekistan
[3, 4]Ajay Kumar Garg Engineering College, Ghaziabad, UP, India

Received 17th January 2025; **Accepted** 24th February 2025; **Published online** 27th March 2025

Abstract

Developing a school timetable presents distinct challenges and limitations for each educational institution, with all schools aiming to produce a high-quality schedule. A conceptual framework for this issue allocates classes and instructors to specific time periods while preventing conflicts. Such a model can be resolved when the workload of teachers and classes fits within the allotted schedule. However, in practical situations with additional complexities, such as specific scheduling needs or requirements for consecutive time blocks, achieving an ideal timetable becomes extremely challenging, and some constraints may need to be loosened to find a feasible solution. Furthermore, while quality is desired in timetables, it is challenging to define or quantify. Scholars have investigated various methods to address timetabling problems, including graph coloring and constraint-based approaches. Emerging educational trends necessitate greater flexibility to accommodate special requirements and elective courses. This article examines simplifying the timetable for educational institutions at all levels. The paper identifies a gap in current solutions and proposes an effective alternative that offers slight improvements in terms of time and space complexity.

Keywords: Time-Slot Scheduling, Stack-Based Algorithm, Timetable Optimization, NP-Complete Problems, Constraint Satisfaction.

INTRODUCTION

Developing a timetable is a persistent challenge for educational institutions at the start of each academic year. The process is labor-intensive, as it must account for various factors such as time slots, classrooms, subjects, faculty, and students. Balancing flexibility with convenience for all parties involved is particularly demanding and requires substantial effort. The creation of a timetable also encompasses several internal complexities. For instance, educational institutions have a limited number of classrooms and instructors. The schedule must ensure that neither teachers nor students are double-booked. Additionally, some classes may need to be scheduled consecutively or kept separate. Furthermore, there may be soft constraints to consider, such as reserving lunch hours, evenly distributing classes, or accommodating faculty preferences for working hours. When applied to larger scales, this problem falls into the category of NP problems, specifically NP-complete. This classification means that all possible combinations must be explored to find an acceptable solution. In computer science, an NP (Non-deterministic Polynomial time) problem is one that can be quickly verified by a computer but may take significantly longer to solve. The Rubik's Cube serves as an example: while it's easy to check if the cube is solved correctly, finding the solution can be time-consuming depending on the complexity of the color combinations. Constructing timetables for larger institutions involves more variables, increasing the effort required to develop a solution. One approach is to examine a subset of the problem or solution spaces. While there isn't a definitive solution, general approaches such as heuristic methods and algorithms like Genetic Algorithms and Tabu Search can be implemented. Based on these concepts, we attempted to design an algorithm to address the timetable problem. This paper aims to simplify timetable creation for educational institutions, identifies gaps in existing solutions, and proposes a more effective approach with improved time and space complexity. The remainder of this paper is structured as follows: A research Gap has been identified by carefully observing the existing solutions. A thorough literature review has been done to closely evaluate various solutions proposed in this field, followed by detailed explanation of our proposed approach and finally the proposed algorithms is evaluatedfor its performance.

Corresponding Author:** ***Ashish Seth,
Inha University in Tashkent, Tashkent, Uzbekistan,

Existing Solutions – Research Gap

Extensive research has been conducted on developing schedules and rosters for university courses and exams. While these areas share some solution strategies, they also have notable distinctions that researchers address independently. Asratian & de Werra, (2002); noted that timetable creation becomes increasingly complex when additional requirements are considered. For example, coordinating classes that require teacher collaboration or scheduling multiple simultaneous courses can greatly complicate the process. In such scenarios, achieving optimal compactness for teacher schedules becomes challenging, ultimately impacting the overall timetable quality. Willemen, (2001) highlighted the issue of consecutive periods, which introduces another layer of complexity; this realistic constraint alone significantly increases the difficulty of efficiently solving timetabling problems.

Daskalaki & Birbas, (2005) proposed different approaches for university and school timetables. In university settings, where many courses involve consecutive lecture blocks, it can be advantageous to temporarily relax the consecutive period requirement and attempt to adjust the schedule later. This method substantially reduces timetable creation time, making it suitable for universities. However, in schools, where consecutive periods are less frequent, this approach is not as necessary. In these cases, prioritizing compact teacher schedules remains crucial for high-quality timetables.

Burke & Rudova, (2007); Burke & Trick, (2005); etc. have suggested a diverse range of timetabling solutions. Similarly, Efthymios Housos *et al,* 2009 proposed a two-phase approach. The first phase uses an integer programming model to assign "work shifts" to teachers, taking into account preferred teaching times. The second phase creates the actual timetable, considering the work shifts but not strictly adhering to them. This approach allows for flexibility while ensuring an optimal overall solution. Core courses are given priority, and teacher preferences are considered during this stage. Seth and Seth (2022) address the challenges of sorting large healthcare datasets and efficiently, proposed a novel multi-layered approach. The findings from their research can be applied to various analytics tasks that require efficient data sorting and processing.

Ashish S, H Aggarwal, and AR Singh (2014) proposed a rule-based approach to estimate the reliability of any systems. Their approach considers various factors to evaluate the reliability of any model, the findings from their research can be applied to timetable scheduling to assess the reliability and to identify potential bottlenecks that may affect to design an effective timetable. Additionally, various other methods have been proposed for these problems, including variations of graph coloring, tabu search, constraint-based techniques, case-based reasoning, and more.

Proposed Solution Algorithmic Steps

This research aims to propose an effective algorithm possible. To address this problem, we employed stacks and lists. We anticipate that solving this problem will be resource-intensive, as it requires prior knowledge of the lessons, their weekly frequency, and the list of groups. Each algorithm possesses unique characteristics, and they continue to evolve over time. For example, algorithms can be utilized to address sorting, comparison, and searching problems. To solve these issues, several established algorithmic techniques are available, including Brute Force, Greedy, and Recursive algorithms. The potential of algorithms, combined with the capabilities of computer science, contributes to an increasingly advanced and efficient future

Proposed algorithmics works as follows

Data Collection: Gather information about the available subjects, their respective lecture hours per week, the number of groups, and the available time slots.

Calculate Maximum Groups per Day: Determine the maximum number of groups that can be accommodated in a day by dividing the total weekly lecture hours by 5 (assuming 5 days of classes per week). Round up the result to get the maximum number of groups.

Time Slot Allocation: Assign subjects to time slots based on the calculated maximum groups per day. Prioritize subjects with higher lecture hours.

Correction: Identify and correct any errors in the allocation process, such as assigning multiple subjects to the same time slot. Overall, the process aims to create an efficient and feasible timetable by considering the available resources (subjects, groups, and time slots) and ensuring that the allocated time slots do not conflict.

The proposed algorithm will be implemented as follows

Step 1: Well, first of all we should take the list of the groups, then we need to take subjects with their count of lessons for per week, and number of time-slots:

Subjects	Groups	Timeslots
OS, 4 lectures per week	1	1: 9:00-10:30
CA, 4 lectures per week	2	2: 10:30-12:00
DB, 2 lectures per week	3	3: 12:00-13:30
LA, 2 lectures per week	4	4: 13:30-15:00
		5: 15:00-16:30
		6: 16:30-18:00

Step 2: For calculation, lets assume that every group come to university 5 days, so we need to know how many lesson groups can enter maximally for per day. In order to know this, we need to add all subjects lecture counts for per week and divide them to 5, because we have 5 days per week, then we take upper-bound of result.

For example: (4 + 4 + 2 + 2) / 5 = 3

Step 3:

Wrong	Correct
CA	CA
OS	OS
CA	LA
OS	OS
LA	CA
DB	DB
. . .	. . .

Now we have 6 slots for per day, we take one group and will make the timetable for this group. We have to make stack for subject which now repeat itself in +-maxLessons.

For example: maxLesson = 3;

Step 4: So now we have stack of lessons for per day and groups list. Well, we create list of object of group which has two fields name and map contains subject in proper day and slot like this:

name: String

Hash<**day**: String, Hash<**slot**: String, **sub**: String>>

Step 5: Then we need to know whether subject is free for asked day and slot, for that also we need to create list of subject objects which contains its name and availability like this:

name: String
Hash<day: String, Hash<slot: String, available: bool>>

Step 6: Now when we set lesson to slot of day we make subject objects availability to false, if it is false already we move to other subject, not other slot, because we want to make easy for students. If there is no available time on this slot for group we can move to other slot.

Step 7: To get completed timetable we need to move stack by max Lessons, because it should be comfortable for next group. We use loops and inner loops to get fully completed timetable.

RESULT AND ANALYSIS

The complexity of proposed algorithm is as follows;

Time complexity:

Well, we have inner loops and outer loops, below you can see the code.

We assumed k-subjects, n-total Lecture Per Week, g-group.
So time complexity worst case is

$$O(k + n * n/5 * k) + O(g * 5 * 6) = O (k(1+n*n)) = O(k*n*n)$$

Therefore, Time complexity: $\mathbf{O(k*n^2)}$.

Space complexity:

Space complexity depends from key value, therefore, O(n) in this example, it is also noticed that for larger key value, if encryption is used then it becomes highly complex to decrypt the same.

Conclusion

The paper presents a novel approach to timetable scheduling using a stack-based algorithm. The proposed approach addresses the challenges of creating efficient and flexible timetables for educational institutions. The algorithm involves data collection, maximum group calculation, subject assignment, error correction, and timetable generation. The analysis shows that the algorithm has a time complexity of O(k*n^2) and its space complexity depends on the key values used. The proposed approach is efficient, easy to implement, and can be adapted to various scheduling scenarios. Future research can explore further optimizations and extensions of the algorithm.

REFERENCES

Ashish S, H Aggarwal, AR Singh. (2014). Estimating Reliability of Service-Oriented Systems: A Rule Based Approach. *International Journal of Innovative Computing, Information and Control ICIC International* Volume 10, Number 3, 1349-4198.

Asratian, A.S., & de Werra, D. (2002). A generalized class-teacher model for some timetabling problem. *European Journal of Operational Research*, 143, 531-542.

Birbas, T., Daskalaki, S., Housos, E. (1999). Rescheduling Process of a School Timetable: The Case of the Hellenic High Schools & Lyceums. Proc. of the 5th International Conference of the Decision Sciences Institute, Athens, Greece.

Efthymios Housos, Sophia Daskalaki. (2009).School timetabling for quality student and teacher schedules. *Journal of Scheduling*

Seth, A., & Seth, K. (2022). The Novel Multi-Layered Approach to Enhance the Sorting Performance of Healthcare Analysis, *International Journal of Reliable and Quality E-Healthcare. IGI Global.* Volume 11, Issue 3

Seth, A., & Seth, K. (2021). Optimal Composition of Services for Intelligent Systems using TOPSIS. *International Journal of Information Retrieval Research,* 11(3), 49–64. doi:10.4018/IJIRR.2021070104

https://www.sciencedirect.com/science/article/abs/pii/0360131586900096

https://www.ijert.org/an-overview-of-the-heuristic-approaches-for-university-course-timetabling-system

https://coderedirect.com/questions/185073/algorithm-for-creating-a-school-timetable

Appendix-1

Implementation of the proposed approach

Classes

```
class Group
-name: String
-subjectAt: Hash<day: String, Hash<slot: String, sub: String>>

class Subject
-name: String
-subjectAt: Hash<day: String, Hash<slot: String, available: bool>>
-void isFree: Boolean

class SubCountForWeek
-name: Stirng
-count: Int
```

Methods

```
subjectStack = generateSubjectStack()
for(group in groupStack.length){
        group.name = groupStack.pop()
        maxLessonsPerDay = subjectStack.length()
        for(day in 1..5)
                int setLessons = 0
                bool booked = false
                for(slot in 1..6)
                        sub = subjectStack.pop()
                        free = sub.isFree(day, slot)
                        if(free){
                                sub.booked(day, slot)
                                group.set(day, slot, sub.name)
                                booked = true;
                                setLessons++;
                        }
                        if(setLessons == maxLessons)
                                break;
                        if(!booked)
                                subjectStack.push(sub);
        subjectStack.moveByMax();

Stack generateSubjectStack(){
        List subCount = [SubCountWeek('OS', 4),
                        SubCountWeek('CA', 4),
                        SubCountWeek('DB', 2),
                        SubCountWeek('LA', 2)];
        int allLessons = getAllLessons();
        int maxLessons = allLessons/5;
        int total = 0;
        List<String> bigList;
        while(total !== allLessons)
                List<String> list;
                count = 0;
                while(count != maxLessons)
                        for(sub in subCount)
                                if(sub.count == 0) ||
(list.contains(sub.name))
                                        continue;
                                else{
                                        list.add(sub.name)
                                        sub.count—;
                                        count++;
                                        total++;
                                }
                bigList+=list;
}

Int getAllLessons(){
        lessons = 0;
        for(lesson in subjectsList)
                lessons+=lesson.count;
        return lessons;
}
```

****