A Solution to the University Timetabling Problem using Graph Colouring

Ahmad Soni
School of Computer Science and Applied Mathematics
University of Witwatersrand
Johannesburg, South Africa
amd_soni@live.co.za

Ritesh Ajoodha
School of Computer Science and Applied Mathematics
University of Witwatersrand
Johannesburg, South Africa
ritesh.ajoodha@wits.ac.za

Kershree Padayachee
Science Teaching and Learning Centre
University of Witwatersrand
Johannesburg, South Africa
kershree.padayachee@wits.ac.za

***Abstract*—With the ever increasing number of students in tertiary institutions and the negative effects of overcrowded and non-optimal classrooms being further understood. It has become increasingly important to develop a system which can correctly and optimally create timetable solutions for each individual student within these tertiary institutions. We note that solutions to this problem have been proposed in various capacities. We will briefly look at some of these algorithms and their results. Our main objective is to investigate further, particularly with respect to students across multiple subjects and faculties. We propose a graph colouring method to solve the problem with a particular focus on hard constraints surrounding different faculties and degrees. Multiple timetables will be generated and cross-validated against each other for accuracy and error reduction. We successfully generate graphs using our algorithm and show an example timetable.**

***Index Terms*—Timetabling, Graph Colouring, Vertex, Greedy**

I. Introduction

The University Timetabling problem is a popular and ever increasingly prevalent NP-Complete problem which has been emerging in the field of Computer Science. With the continuous development of Tertiary Education in most developed and developing countries, there has been a significant increase in the number of students and teachers occupying spaces within these tertiary institutions. With the rise in numbers, so too has the need for an optimal and efficient solution to create a schedule for the allocation of students and teachers based on multiple factors such as their studying disciplines and the time and location of their classes, while avoiding unfeasible conflicts within the system.

A major reason for the need to solve this problem arises from the need for the most optimal teaching conditions in Universities. It has been shown by research done by [1] that overcrowding in classrooms has a direct negative impact on student learning. Additionally, too many students in venues which are too small result in a decrease in learning quality [3] and often weaker students who require more assistance are left ignored. [4]

Due to the popularity of the timetabling problem, there have been many attempts to solve or provide more optimal solutions by different researchers. Examples of such attempts are namely, a graph colouring, but in an exam timetabling capacity by [5], and the genetic algorithm by [12] and [7]. Both of these approaches have been implemented successfully but subject to their own limitations.

We wish to build upon the research done by others by expanding the scope of the graph colouring algorithm. We will apply the algorithm to create a timetable for class scheduling for an entire year, not limited to just the exam period. We will also be using a unique greedy approach to our colouring algorithm which could prove more time efficient for larger universities.

We will generate our graph by making pairings of shared subjects between students as vertices and then connecting these vertices by edges depending on the overlap between students and their subjects. We will then apply our greedy colouring algorithm to colour our generated graph. We will be evaluating our results by means of cross validation and examination of chromatic number over random order colouring of our graph.

Ideally a University should be able to generate a timetable for each student and lecturer which should ideally allocate subjects to times and venues whilst avoiding major conflicts, such as having two different subjects in the same venue at the same time. Currently many Universities do not have an automatic way to generate such a timetable and in some cases, manual computation techniques are being used to create these student and lecturer timetables. This is not ideal as the nature of the Np-Hard problem means that a manual computation will be highly non-optimal and even contain possible errors and overlapping constraints.

The nature of South African Universities, where there are multiple faculties and schools all existing on the same campus is the primary focus for this paper. Due to the large number of students because the University campus is so large, there tend to be a large variety of varying numbers of students per degree and even per subject. This makes an optimization for class sizes a significant problem to solve.

The creation of such an algorithm will contribute by demonstrating the possibility of fully automating the scheduling procedure, providing universities with a inexpensive solution to their timetabling problems and encouraging further research and development in the area.

In the following sections we will explain background information and terminology as well as look into the related work in depth. The Methodology and Results of the research will follow thereafter.

II. Background

In this section, we will discuss Graph Colouring, NP problems and cross-validation. Graph colouring itself is a ubiquitously used algorithm in the real world with multiple applications. Some examples of the applications are scheduling in sport and pathfinding in robotics. While graph colouring can be used to find a solution to our problem, it definitely is not the only solution as proven by the volume of literature on the topic.

A. Np Problems

In terms of complexity, the problem of creating the timetable itself is said to be NP Complete but the optimization of the timetable is NP Hard [2]. A NP problem is defined as a problem that can be solved by a Non-deterministic Turing Machine in Polynomial time. An NP Complete problem ‘X’ is one that can be solved within NP and is reducible to X in polynomial time. A NP Hard problem ’Y, however, can be reducible to ‘Y’ in polynomial time but is not necessarily in NP. This means that the problem cannot be solved in polynomial time until P = NP. These types of problems are computationally intensive to solve without the use of computers and algorithms as they usually involve a large and irregular solutions space.

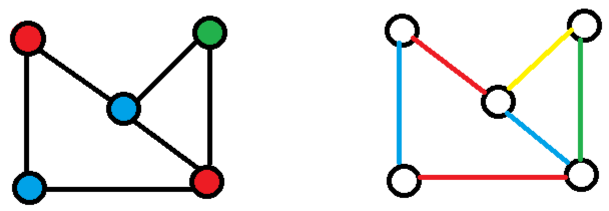
B. Graph Colouring

Graph colouring works by first generating a graph with multiple vertices. In this case, each vertex would represent a subject for which there is at least one student enrolled for. We would then generate relationships between the vertices and connect these relationships via edges. We pick a vertex to begin the colouring process and colour that vertex with one of our available colours.

We then move to the next vertex and colour it with another colour such that there are no adjacent vertices with the same colour. Two adjacent colours which are the same would represent a clash within our constraints for this problem. To optimize this, we traverse all vertices while trying to use as little colours as possible but also adhering to our specified hard constraints and as many soft constraints as possible. The application of this theory has been used in various other practical situations such as automatic sudoku solvers.

Vertex Colouring and Edge Colouring

The two most common forms of Graph Colouring are the Vertex and Edge Colouring algorithms. In Vertex colouring, we colour each vertex of a graph such that no adjacent vertices have the same colour. In Edge Colouring, we assign a colour to each edge instead, such that no adjacent edges have the same colour. For this paper we will be investigating Vertex Colouring.



Vertex Colouring vs Edge Colouring

There is some important terminology to consider when looking at graph colouring algorithms.

- *Proper Colouring* refers to a graph that has been coloured where every edge and every vertex have been assigned different colours.
- *Chromatic number* of a graph G is the minimum number of colours needed for a Proper Colouring of G.
- *K-Colorable* refers to a graph that has been coloured with at most K colours and is a Proper Colouring.
- *K-Chromatic* refers to a Graph G that is coloured with its chromatic number, i.e. Using the minimum number of colours.

Greedy Graph Colouring

The Greedy Graph Colouring Algorithm works on the basis of Brooks’ Theorem.
For every connected graph G that is not an odd cycle or a complete graph

$$x(G) \leq \Delta(G)$$

Greedy Colouring will traverse the graph one vertex at a time and at each vertex it will assign the first available colour which is not in use by any of its adjacent neighbours. Due to how this works, Greedy Colouring does not always necessarily produce graphs which have the minimum number of colours possible. At most for a graph of maximum degree x, where degree is the number of edges connecting to a vertex, a greedy algorithm will produce a maximum of $x + 1$ colours.

C. K-Fold Cross Validation

We will be evaluating the results of our model by using cross-validation procedures. In particular we will be using a specific type of cross-validation known as k-fold validation. In this form of cross-validation we will run our algorithm a fixed number, or k, times. Each of these iterations is called a ‘fold’. We perform analyses on each individual fold, compare them to each other and ultimately we average the folds to get a final fold. In our case, we might not be able to get an 'average' timetable or graph, so we will have to pick the most frequent result. The purpose of this cross-validation is to protect our model against overfitting. The process of running multiple iterations ensures our results are consistent and reduces the overall error of our model.

D. Related Work

1) Graph Colouring Approach to Exam Timetabling: If we look at a University Exam timetabling solution proposed by [5] which is conceptually similar to that of a University Scheduling Timetable. A graph colouring approach was proposed and implemented. The algorithm was used to generate a timetable for foundation students in Multimedia University, Malaysia for trimester 2 in the years 2009/2010.

The algorithm works by following steps required for graph colouring.

Subjects are divided into main foundations, for example: Sciences, Commerce, Law.

The subjects are divided and sorted further by intakes, since the University has multiple intakes in an academic year. A colour is then assigned to each intake. After this is done, a further colour is then assigned based on subjects. An objective function is used to measure how well soft constraints are satisfied and we assume all hard constraints are fulfilled. A conflict matrix is used to check where constraints possibly overlap. Each exam is represented by a vertex and edges represent conflicts [5]. This will allow us to find the minimum number of colours apple to vertices so that no two adjacent vertices have the same colour [6]

The algorithm managed to generate an Exam timetable and improved on the current system offered by the university, which did not consider any soft constraints in its calculations. It was able to schedule exams for a maximum of 1323 students per slot compared to the old maximum of 1263 per slot. However, the final algorithm did end up ignoring most of the soft constraints and some hard constraints, showing that further optimality and research can still be done on the topic with regards to the graph colouring method. Furthermore, this could be extended to university class timetabling, and not just exam timetabling, which would be a lot larger, and more robust.

2) Genetic Algorithm: Another popular algorithm which has been examined in a classroom scheduling approach by [12] and [7] in exam timetabling, is that of the genetic algorithm. The exam timetabling proposed by [7], looked at student data from the engineering department of Ibra college of Technology in the year 2011-2012 but opted for a generalized solution, and the classroom scheduling approach by [12] also opted for a more generalized approach.

Genetic Algorithms are a specific type of evolutionary programming which draw heavy influence from the principles of Darwinnian biological evolution. These algorithms were formalized by John Holland in 1975 and have since gained traction with advances in technology and computational power since then. They are particularly useful in problem solving given large irregular search spaces with multiple solutions [8]

The algorithm functions by generating a selection of feasible timetables and selects the fittest timetable from the selection as its basis for iterating upon. Fitness is determined by the number of soft constraint violations as we assume all hard constraints are always met for a timetable to be considered feasible. It makes use of various genetic operators such as crossovers, mutations and selection in its iterative process. [7].

In both algorithm implementations, the algorithm defines a chromosome, which is an initial sample on which the various evolutionary operators will act on. This chromosome is represented usually as a bit string, but can be represented as a mapping as in the case of [7]. The initial sample or initial population is initialized randomly based on input the data used. The evaluation or fitness function which is used to select the fittest of the feasible time table is given by the equation f = 1/1+x where x is the collective sum of conflicts and constraints. We will now investigate the evolutionary operators

Selection
Selection or Reproduction is the operator which is usually applied first on a population after its fitness value has been calculated. [7]. Selection is the process in which the timetable with the highest fitness function value is picked to be used for the iteration phases.

Crossover and Mutation
In the crossover stage, two strings are compared to each other and recombined to produce a more optimal string [12]. This is done to iterate the chromosomes and generate better ones across generations. After the crossover stage, the fitness is recalculated. If the fitness value is higher than its original string, the algorithm will stop and conclude that an optimal solution has been reached. If the fitness value is lower, however, the Mutation operator will be called. The mutation operator will alter values in the chromosome and then recalculate fitness again. If the fitness value is higher than the original string, the algorithm will stop. If not, however, the algorithm will loop back to the Selection phase.

Both papers produced a satisfactory timetable with 0 conflicts and a fitness score of 1. Indicating that maximum fitness had been achieved. Both algorithms were run using fairly small sample sizes, with the one by [12] consisting of 54 lecturers over 36 courses with 12 venues and 20 periods. This would serve as a good proof of concept for extending the implementation of the algorithm to a University scale, especially in a South African university where students number in the thousands and courses in the hundreds. Both algorithms could stand to optimize further with the implementation of additional soft and medium constraints.

3) Other Algorithms: Some examples of other algorithms which have been used are a Ordering Heuristic Model by [6] which operated by combining graph colouring heuristics with a squeaky wheel optimization framework. A Tabu search method by [10] which uses the metaheuristic local search method to perform its timetabling optimization. A local search procedure with constraint programming was implemented successfully by [11].

III. Research Question

The Question we wish to address here is:

Can a graph colouring approach be used to solve the University Timetabling problem with a specific focus on class scheduling instead of exam scheduling for an entire faculty of students with vastly different majors and subjects?

To solve this question we will create a timetable for every student while adhering strictly to a set of hard constraints and loosely to a set of soft constraints. Hard constraints must always be adhered to and soft can be adhered to when possible but are not necessary. For this stage of the Research we will be investigating if it is possible to generate such a timetable using graph colouring and as such we will not be adding too many additional soft and medium constraints.

The optimization of this problem would ensure that all of the Hard constraints are fulfilled along with as many and soft constraints as possible. Additionally we want to solve the problem by using a graph colouring approach particularly using vertex colouring and a greedy colouring approach. Our most important hard constraints are: A student cannot be assigned more than one room at a given time slot, a student may not have more than one subject at a given time slot and a single room may only be assigned a single subject.

A. Purpose Statement

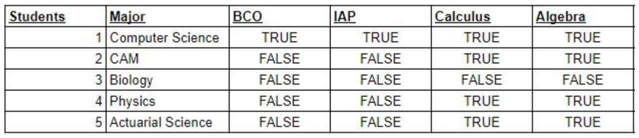
By solving the above problem we would create an applicable solution to the timetabling problem in situations where we are dealing with varying degrees and subjects being offered in the same faculty. This would allow universities to automatically generate timetables for students and lecturers in a much more efficient manner leading to less unnecessary cost overheads and additional time for other University matters.

This research would also seek to deepen the understanding of a graph colouring algorithm, particularly with respect to the timetabling problem. It's contribution to the other timetabling literature could be a helpful addition to further research in the area.

IV. Research Methodology

A. Data

Since the timetabling problem in general is non-specific and can apply to a multitude of scenarios, the data used for testing is entirely synthetic. It was manually curated to be data which could occur in a University setting and not purely random so we could still achieve relevant results. Our data consists of a selection of students and the courses which they are enrolled for in a semester. This is scalable and there is no limit on how many subjects a student can or cannot take. Since we created this dataset, we did not have to do any explicit pre-processing on it except ensuring that it is in the correct format for our algorithm.



Students	Major	BCO	IAP	Calculus	Algebra
1	Computer Science	TRUE	TRUE	TRUE	TRUE
2	CAM	FALSE	FALSE	TRUE	TRUE
3	Biology	FALSE	FALSE	FALSE	FALSE
4	Physics	FALSE	FALSE	TRUE	TRUE
5	Actuarial Science	FALSE	FALSE	TRUE	TRUE

Fig. 1. Sample of Data used

Our Graph Colouring Algorithm was run on three distinct datasets. A trivial case consisting of a small number of students in the same major with the same curriculum. A case where we had multiple students across multiple majors taking fixed curricula. Finally we have a complex case where we added students on a general degree path who could take whichever subjects they wanted to.

B. Graph Structure

Firstly we create a graph consisting of vertices and edges. A vertex is created for each available subject. We iterate through the dataset and create pairings between subjects dependent on how many combinations of True entries we have per student. If there is a True value between two subjects for a student, we create an edge to represent a relationship between the two vertices. Once all edges have been drawn, we consider our initial graph complete.

C. Greedy Colouring

The next part of the Algorithm will traverse each vertex in order and assign an appropriate colour depending on which colours are available. The algorithm works by starting out with a list of available colours available for use and an empty list of forbidden colours. It iterates through each vertex and at each vertex it will check the neighbours of the vertex. If there is a neigbour with a colour, it will add that colour to the forbidden colours list and assign the next available colour in the available colours list. After a vertex has been coloured, the algorithm will proceed to the next vertex and repeat the colouring process.

D. Cross-Validation and Testing

We implemented K-fold cross validation to ensure our generated graphs are accurate. The algorithm was run 10 times for each different test case and the graph which appeared most frequently was manually selected. We also used a shorter algorithm to randomize the order in which the vertices are coloured. We ran this 10 times to check if we would get a consistent chromatic number for each random colouring. This is important, as aforementioned, the greedy algorithm might not necessarily give us a the most optimal solution depending on the order in which the vertices are coloured. This allows us to evaluate the consistency of the algorithm's performance.

E. Limitations

Since we are using a greedy algorithm, our colouring solution might not necessarily be the most optimal.

V. Results and Future Improvements

In this section we will discuss the results of the algorithm as well as future improvements that can be made.

A. Trivial Dataset

The first dataset which was tested was the trivial dataset consisting of a small number of entries from a single major all enrolled for the same subjects. The dataset contains 6 vertices as it has 6 possible subjects a student can be enrolled for.

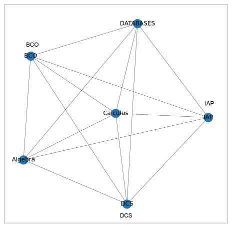


Fig. 2. Trivial Graph

Based on this, we can see the graph has a chromatic number of 6. We now apply our colouring algorithm to the graph.

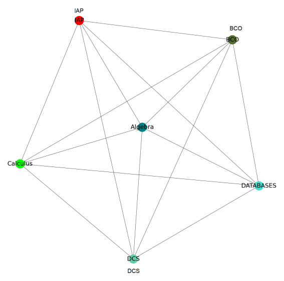


Fig. 3. Trivial Coloured Graph

In this case, 6 different colours had to be used to properly colour the graph.Our random colouring algorithm returned a consistent number of 6 colours, indicating optimal colouring.
The second dataset we wish to look at is the dataset containing students from different faculties on a set curriculum. This dataset contains 12 vertices for 12 possible subjects.

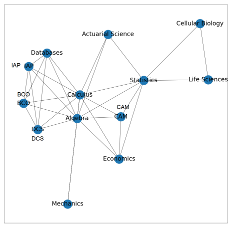


Fig. 4. Set Curriculum Graph

Based on this, we can see the graph has a chromatic number of 12. We now apply our colouring algorithm to the graph.

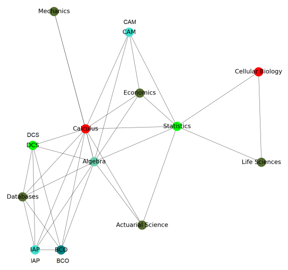


Fig. 5. Set Curriculum Coloured Graph

6 different colours had to be used to properly colour the graph, the same as the trivial case, despite having a much higher chromatic number. Our random colouring algorithm returned a consistent number of 6 colours, indicating optimal colouring.

Finally we wish to look at the most complex dataset where we have added in students who are free to choose whichever subject they would like to take. We have still limited the dataset to contain 12 vertices for 12 subjects.

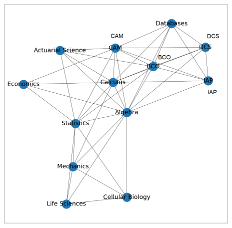


Fig. 6. General Course Students Graph

Based on this, we can see this graph also has a chromatic number of 12. We now apply our colouring algorithm to the graph.

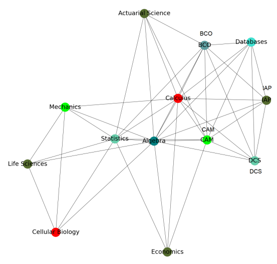


Fig. 7. General Course Students Graph

Here, 7 different colours had to be used to properly colour the graph, which is closer to our chromatic number but still significantly lower. Based on this, our algorithm is colouring quite optimally. Our random colouring algorithm also returned a consistent number of 7 colours showing that the algorithm is being coloured optimally.

Finally we can use our coloured graphs to then generate an appropriate timetable for students:

	Room 0	**Room 1**	**Room 2**
2021-11-14 10:00:00	Databases	Mechanics	Statistics
2021-11-14 12:00:00	None	None	None
2021-11-14 14:00:00	IAP	CAM	Cellular Biology
2021-11-14 16:00:00	Calculus	None	None
2021-11-15 10:00:00	DCS	None	None
2021-11-15 12:00:00	Algebra	None	None

We can see subjects with many students overlapping multiple faculties such as Algebra and Calculus have to be assigned to separate days and other subjects which are relatively degree specific such as Mechanics and Statistics able to be scheduled concurrently with each other. There are no conflicts with our hard constraints detailed earlier, so we can conclude that the algorithm is working as intended.

B. *Future Improvements*

There are many future optimizations and features which can be added to this algorithm to make it more practically useable in everyday situations. Firstly and importantly, we need to develop a method to generate an average graph across multiple iterations of our algorithm. This would give us more reliable results. For futre additions, we could cater for venue sizes and match numbers of students with appropriately sized venues. For example, there would generally be more students taking Algebra and Calculus then there would Statistics or Cellular Biology. We could also add break periods so students have appropriate gaps between their subjects. The problem in general can contain many soft constraints as demanded by a specific institution, all of which would be grounds for future improvements.

VI. Conclusion

The aim of this research was to provide a solution to the University Timetabling Problem using a Graph Colouring approach, particularly with regard to multiple students from multiple degree majors. We accomplished this by creating a non-planar graph of our problem space where subjects were represented by vertices and the relationships between them by edges. We then coloured the graph iteratively, vertex by vertex, using a greedy colouring algorithm. With this approach we were successfully able to generate properly coloured graphs as well as timetables for our synthetic data.

While our greedy algorithm provided satisfactory results, it has yet to be tested on a large, real-world dataset, which may be able to provide further insights into its performance. There is also the possibility of future research where we could use a recursive colouring algorithm instead of our greedy algorithm. A recursive algorithm would check the colours of all previous vertices while assigning colours to its current vertex. This could give us a more optimally coloured graph at the cost of time complexity. We could potentially try to use Edge colouring instead of Vertex colouring, however, for this problem having each subject as a vertex is the more logical option.

VII. Acknowledgment

Thank you to the University of the Witwatersrand for providing the funding and the means necessary for the undertaking of this research paper.

References

[1] Khan, P. and Iqbal, M. (2012). Overcrowded classroom: A serious problem for teachers. University of Science and Information Technology, 49:10162–10165.

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

[3] Carlson B. (2000) Achieving educational quality: What schools teach us. United Nations Publication. ISSN 1020-5179.

[4] Nemrah (2006) Classroom Management and Organization, 2nd Edition. Amman, Dar Yafa Sarhad University of Science and Information Technology, Peshawar, KPK Pakistan

[5] Hussin B, Basari A, Shibghatullah A, Asmai S (2011). Exam Timetabling Using Graph Colouring Approach. IEEE Conference on Open Systems (ICOS2011)

[6] Rahman S, Bargiela A, Burke E, Ozcan E (2009). Construction of Examination Timetables Based on Ordering Heuristics.24th International Symposium on Computer and Information Sciences, ISCIS 2009. 680 - 685. 10.1109/ISCIS.2009.5291905.

[7] Jha S, 2014. Exam Timetabling Problem Using Genetic Algorithm. International Journal of Research in Engineering and Technology. eISSN: 2319-1163 , pISSN: 2321-7308

[8] Colorni A, Dorigo M, Maniezzo V, (1991). Genetic Algorithms and highly constrained problems: The time-table case.Parallel Problem Solving from Nature, Vol. 496, 1991, pp. 55-59.

[9] Rawat S, Rajamani L (2010). A Timetable Prediction for Technical Education System using Genetic Algorithm. Journal of Theoretical and Applied Information Technology, Vol. 13(1), 2010, pp. 59-64.

[10] Lawal H, Adeyanju I, Moidiorra E, Arulogun O, Omotosho O, (2014). University Examination Timetabling Using Tabu Search. International Journal of Scientific and Engineering Research, Volume 5, Issue 10.

[11] Francis K, Manga I, Sajiyus O, (2016). Scheduling Algorithm for University Timetabling Problem. Journal of Computer Engineering,e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. I (Nov. - Dec. 2016), PP 39-43.

[12] Ahmad I,S Sufahani , Ali M and Razili C (2017). A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique. IOP Conf. Series: Journal of Physics: Conf. Series 995 (2018) 012050.