

University Course Timetabling using Bayesian based Optimization Algorithm

<https://doi.org/10.3991/ijes.v6i2.8990>

Alinaswe Siame (✉), Douglas Kunda
Mulungushi University, Kabwe, Zambia
alinaswe7@gmail.com

Abstract—The timetabling problem has traditionally been treated as a mathematical optimization, heuristic, or human-machine interactive problem. The timetabling problem comprise of hard and soft constraints. Hard constraints must be satisfied in order to generate feasible solutions. Soft constraints are sometimes referred as preferences that can be contravened if necessary. In this research, we present is as both a mathematical and a human-machine problem that requires acceptable and controlled human input, then the algorithm gives options available without conflicting the hard constraints. In short, this research allows the human agents to address the soft-constraints as the algorithm works on the hard constraints, as well as the algorithm being able to learn the soft constraints over time. Simulation research was used to investigate the time tabling problem. Our proposed model employs the use a naïve Bayesian Algorithm, to learn preferred days and timings by lecturers and use them to resolve the soft constraints.

Keywords—Algorithms; University course Timetabling/Scheduling; Constraints; Bayesian decision approach; Learning algorithm

1 Introduction

University course timetabling / scheduling is a very common problem, and our universities here in Zambia are not an exception especially in case where a University has two campuses running numerous number of programs that are inter-related by shared courses, lectures and rooms, simultaneously. This makes the process of developing a timetable or a training schedule a very difficult task that can take human brains many hours to develop and still have errors. The goal of the university course timetabling is to find a method to allocate whole events to fix predefined timeslots and rooms, where all constraints within the problem must be satisfied. Events include students, lecturers and courses where resources encompass the facilities and equipment's of classrooms such as theoretical and practical rooms. Also, timeslots include two main components, namely daily and weekly timeslots which vary from one institution to another. However, each classroom also has its own components including audio-visual equipment's (video projector), number of chairs necessary for courses

allocated to those classrooms (the capacity of theory and practical rooms), number of whiteboards related to each theory and practice classroom.

Himawan, Anand, & Yong [1] introduced an object-oriented automated timetabling system, which incorporated model, knowledge and data bases. The model-base is made up of heuristic procedures to carry out various aspects of scheduling, e.g. how students are to be assigned to a class, or a class to a time-period. The knowledge-based consisting of rules that control the flow in the execution of the procedures, e.g. what action to be taken if no feasible time-period can be found for a class. The database consisting of data needed for the scheduling process; by isolating and managing it via a database management system (DBMS), they allowed maximum flexibility in the specification of scheduling parameters. The adoption of object-oriented methodology enabled them to achieve higher productivity in the development process, but equally important was the significant improvements in the scheduler's performance. For one, methods allowed the localization of constraint checking, e.g. restrictions on the teaching load of instructors was implemented as a method for the instructor object class. Moreover, message-passing enabled object classes to communicate autonomously, eg. when a particular class had been successfully scheduled, the instructor, student and room object instances that are involved would be automatically updated via messages.

The approach taken by Himawan, Anand, & Yong [1] is one that supports flexibility by allowing the human agent to be part of the process, this research has benefited from such an approach in order to develop a more flexible timetabling system. However, we go further to improve our algorithm to allow it to be able to learn from previous human agent trends, in order to help the algorithm be able to provide a more optimal solution even before the human agent can attempt to work on the soft constraints.

This study contributes a design and test of a model with an improved algorithm to perform course timetabling with the flexibility of allowing soft constraints to be altered by the users while maintaining the integrity of all the hard constraints. The algorithm should have the ability to learn from previous user input so as to provide more optimal timing slots as per previous user preference. The learning is done using a Naïve Bayesian approach. According to [42], a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

To achieve this, we have employed the use of Simulation research. Computer simulations can be used in training, teaching or entertainment. When it comes to research, According to [32], [33] simulation can be used to get a better understanding of a phenomenon of interest and for the purposes of prediction.

2 Problem formulation

University time tabling is still problematic especially the need to provide optimal timeslots for lecturers. Many approaches have been taken in academic environments

to address the problem of course timetabling. Typically, student scheduling and course scheduling have been treated as separate tasks[2].

Hamed, Karimpour, & Hadidi [3] add on to say scheduling is one of the problems which so many researches have been conducted on it over the years. The university course-timetabling problem which is an NP-hard problem is a type of scheduling problem. Timetabling process must be done for each semester frequently, which is an exhausting and time-consuming task. The allocation of whole of events in timeslots and rooms performs by the university course timetabling process considering the list of hard and soft constraints presented in one semester, so that no conflict is created in such allocations. In the university course-timetabling problem, the hard constraints should not be violated under any conditions; soft constraints also should not be violated as much as possible.

Dorneles, deAraújo, & Buriol [4] also show how the same problem is found at high school timetabling level as a classical combinatorial optimization problem that takes a large number of variables and constraints into account. Due to its combinatorial nature, solving medium and large instances to optimality is a challenging task.

Formulation of the Object oriented approach to timetabling according to [1] is as follows:

Formally, in timetabling, we are required to schedule p periods, $k = 1, 2, 3 \dots p$, c classes, $i = 1, 2, 3 \dots c$, and t instructors, $j = 1, 2, 3 \dots t$. The relation of lecturers to classes is represented by a requirement matrix $X = (r_{ij})$, whose (ij) th element represents the number of meetings of class i and lecturer j . Let, $A = (a_i)$ be a vector of dimension c representing class availability and $B = (b_j)$ be a vector of dimension t representing availability of instructors. The solution to the problem is then a matrix, $S = (m_{ijk})$, of dimension $c * t * p$ such that

$$\begin{aligned} \sum_k m_{ijk} &= r_{ij} \quad \forall i, j \\ \sum_j m_{ijk} &\leq a_i \quad \forall i, k \\ \sum_i m_{ijk} &\leq b_j \quad \forall j, k \end{aligned} \tag{1}$$

Usually a_i and b_j are 1. The above three equations specify the constraints that have to be satisfied; the necessary and sufficient conditions for such a solution matrix to exist are:

$$\begin{aligned} \sum_j r_{ij} &\leq p a_i \quad \forall i \\ \sum_i r_{ij} &\leq p b_j \quad \forall j. \end{aligned} \tag{2}$$

This method has been by far a more applicable solution to the current problem at hand and thanks to further advancements in computer processing power the human intervention is no longer a computationally expensive approach. Furthermore, our model will be able to provide an interactive component to the timetabling system, which will enable the completion of the scheduling when the automated system has failed to do so. The feature is none existent in [1].

The objectives of this research are to design and test an improved algorithm to perform course timetabling with the flexibility of allowing soft constraints to be altered by the users while maintaining the integrity of all the hard constraints. The algorithm should have the ability to learn from previous user input so as to provide more optimal timing slots as per previous user preference.

3 Related works

Building university timetables is a complex process that considers a big number of resources that are limited and require absolute optimal use to get a workable solution as finding a perfect solution is close to impossible. This section brings out some of the earlier used approaches to solve the problem at hand.

4 Current existing algorithms

Analyzing the effects of solution space connectivity with an effective metaheuristic for the course-timetabling problem presents a powerful two-stage metaheuristic-based algorithm to approximately solve it. This study describes a novel generative hyperheuristic entitled ‘add–delete Lists’ and details their application across two different problem domains that is Tracks 2 and 3 of the 2007 International Timetabling Competition. An Add–Delete list is a ‘ruin-and-recreate’ sequence which is applied to a solution re-presentation [5]. The use of two algorithms one for hard constrains, the other for soft constrains to provide more optimal timeslots in-order to reduce the number of soft constraints.

Artificial bee colony (ABC) algorithm has been successfully used for tackling incapacitated examination and course timetabling problems. Experimental analysis of the proposed technique showed some improvement in performance over previous techniques. The experiments showing the effectiveness of proposed hybridized-ABC are presented to study the influence of the HCO. Note that the role of HCO is important in fine-tuning the search space region of the food sources (solutions) toward the local minima by enhancing the local exploitation capability of the proposed technique [6]. This approach achieves computational optimal standards. However, it still begs the question can the results be easily satisfying the dynamic set of personal preferences of human feelings in this case lecturers.

Multi agent systems have a more general concept and for all types of current systems, including multiple autonomous components are applied to the following features and include: (1) each agent has the ability of solving a problem incompletely, (2) in multi agent

systems there is no general control system, (3) data are as distributed and (4) computations are asynchronous [3]. The multi-agent distributed system, increases the satisfaction of soft-constraints, while also adhering to the hard constraints. However, this approach can slow down the implementation process if the dataset is large and easily cause the agents to delay the process of finding the solution.

Automated Scheduling system for thesis and project presentation using Forward Chaining Method with Dynamic Allocation Resources presents a practical method for modeling and solving a dynamic resource allocation of automatic scheduling problem using forward chaining heuristic approach, in the case of undergraduate Student's Thesis and Project presentations timetable [8]. This system proved to have worked well. However, its application to university course timetabling problem might not exactly be optimal because of the vast array of resource and requirements to be met.

High school timetabling is a classical combinatorial optimization problem that takes a large number of variables and constraints into account. Due to its combinatorial nature, solving medium and large instances to optimality is a challenging task.

The proposed a fix-and-optimize heuristic combined with a variable neighbourhood descent method that produces solutions which satisfy all hard constraints, i.e., feasible solutions [4]. This method works well for a high school timetabling system; however, it is not so feasible to apply most of its concepts to a university timetabling scenario.

Post-publication disturbances such as absence of teachers typically pose a need for schedulers to rapidly implement some minor changes to avoid empty periods in the timetable. To solve this problem, the scheduler has to create a balance between reducing the number of empty periods, keeping the schedule stable, i.e., not deviating too much from the old timetable and being alert on the number of shifts on a specific day and over days [9]. This model has dealt with the issue of changes in timetabling due to disturbances; this is a feature that could be applied to university timetabling situations.

The timetabling problem has traditionally been treated as a mathematical optimization, heuristic, or human-machine interactive problem. The inclusion of expert knowledge allows for solutions that fit the problem context better, while the use of a database enables a more flexible and maintainable system. The object-oriented paradigm allows for a more efficient design and code implementation of scheduling procedures [1].

In the most recent literature in the area of scheduling/ timetabling algorithm development, the following has been observed [10] defined the problem being resolved was concerned with the assignment of lecturers to specific timeslots and rooms. The quality of the solution was measured in terms of a penalty value which represents the degree to which various soft constraints are satisfied. this hybrid evolutionary approach was tested over established datasets and compared against state of the art techniques from literature. This approach Proved to be better than other approached explained in [10]. Similarly [11],[12] make use of hybrid approaches in developing faster more efficient university timetabling algorithms. However, this it was not tested to see if it provided human acceptable results [13] had the goal to satisfy as many of the soft constraints as possible whilst constructing a feasible schedule. They

presented a composite neighborhood structure with a randomized iterative improvement algorithm. This algorithm always accepts an improved solution and a worse solution is accepted with a certain probability. Preliminary comparisons indicated that this algorithm was competitive with other approaches in the literature. Indeed, it produced seven solutions that were better than or equal to the published penalty values. Similar approach is taken by [14], who proposes a metaheuristic solution, more precisely an adaptive large neighborhood search, which is based on repetitively destroying and subsequently repairing relatively large parts of the solution.

However, [13] noted that in their future research, they would be aiming at exploring how the algorithm could intelligently select the most suitable neighborhood structures according to the characteristics of the problems. While [14], noted that adding perturbation to the evaluation of potential insertion positions did not improve the performance significantly. Having noted the new discovery, this research wishes to further test and validate the need to add learnt patterns from previous allocations.

Several other techniques have been proposed in solving the UCTP and its variants. Various techniques have been developed for automated timetables generations. Other popular techniques include graph-coloring algorithms, simulated annealing and tabu search, genetic algorithms (GA) and Particle Swarm Optimization (PSO). Constraint-based Programming [15] and Particle Swarm Optimization (PSO) [16] [17] [18] have become an interesting approach for solving timetabling problems recently [19].

5 Application of Particle Swarm Optimization (PSO) in university course timetabling

[20] Defines PSO algorithm as a multi-agent general meta-heuristic method, that can be applied extensively in solving many difficult problems [21]. The PSO consists of a swarm of particles in the space; the position of a particle is indicated by a vector, which presents a solution. PSO is initialized with a population of M random particles and searches for the best position (solution or schedule). In every generation or iteration, the local best and global best are determined through evaluating the performances in terms of the fitness values of current population of particles. A particle moves to a new position obtaining a new solution guided by the velocity (a vector). Hence, the velocity plays an important role in affecting the characters of creating new solution. There are two experience positions used in the PSO; one is the global experience position of all particles, which memorizes the global best solution obtained from all positions (solutions) of all particles; the other is the individual experience position of each particle, which memorizes the local best solution acquired from the positions (solutions) of the corresponding particle has been at. These two experience positions and the inertia weight of the previous velocities are used to determine the impact on the current velocity. The velocity retains part of prior velocity (the inertia) and drives particle toward the direction based on the global experience position and the individual experience position. Thus, the particles can derive new positions (solutions) by their own inertia and experience positions.

D dimension space represents the search space (the number of dimensions is corresponding to the parameters of solutions) and the population consists of M particles. $X_i = \{X_{i1}, \dots, X_{iD}\}$, X_{iD} be the particle i with D dimension space ($i = 1, \dots, M$). A position X_i has a rate of position change called velocity $V_i = \{V_{i1}, \dots, V_{iD}\}$. The individual experience is a position $L_i = \{L_{i1}, \dots, L_{iD}\}$, the local best position for the ith particle (called pbest). Additionally, $G = \{G_1, \dots, G_D\}$ represents the global best position among all the population of particles achieved so far (called gbest). The PSO algorithm could be performed by the Eq. (3)

$$V_{newid} = W V_{id} + C_1 r_1 (L_{id} - X_{id}) + C_2 r_2 (G_{id} - X_{id}) \quad (3)$$

$$X_{newid} = X_{id} + V_{newid} \quad (4)$$

where w is an inertia weight used to determine the influence of the previous velocity to the new velocity. The c1 and c2 are learning

[20] Introduces a novel meta-heuristic algorithm that is based on the principles of particle swarm optimization (PSO) is proposed for course scheduling problem. The algorithm features include: designing an ‘absolute position value’ representation for the particle; allowing instructors to lecture based on flexible preferences, such as their preferred days and time periods, the maximum number of teaching-free time periods and the lecturing format (consecutive time periods or separated into different time periods); and employing a repair process for all infeasible timetables.

Since the solution space of the course-scheduling problem is discrete, a local search mechanism is incorporated into the proposed PSO in order to explore a better solution improvement.

The experimental results demonstrate that the proposed hybrid algorithm yields an efficient solution with an optimal satisfaction of course scheduling for instructors and class scheduling arrangements. This hybrid algorithm also outperforms the genetic algorithm proposed in the literature.

[19] Proposed approach (hybrid particle swarm optimization with constraint-based reasoning) uses particle swarm optimization to find the position of room and timeslot using suitable objective function and the constraints-based reasoning has been used to search for the best preference value based on the student capacity for each lesson in a reasonable computing time.

The most common variants of educational timetabling problem are the UCTP and Exam Timetabling. Both have similar constraints with the main difference being ETP events can take place in the same room and timeslot as long as the desire constraints are satisfied while in UCTP, only one event can take place in desire room at a prefer timeslot

[22] applied PSO in order to solve the course timetabling problems in this work. To reduce the computational complexity, a timeslot was designated in a particle’s encoding as the scheduling unit. Two types of PSO, the inertia weight version and constriction version, were evaluated. Moreover, an interchange heuristic was utilized to explore the neighboring solution space to improve solution quality. Additionally, schedule conflicts are handled after a solution has been generated.

Experimental results demonstrate that the proposed scheme of constriction PSO with interchange heuristic is able to generate satisfactory course timetables that meet the requirements of teachers and classes according to the various applied constraints.

6 Application of genetic algorithms in university course timetabling

A genetic algorithm starts with a set of random solutions to the problem. The initial solution is randomized and therefore crude. Each solution is called a chromosome and consists of several genes which are values corresponding to certain properties in the solution. These genes can then be used to control the fitness of the chromosome. Based on the chromosomes' fitness, they are crossed with each other to create a new offspring. These offspring are then randomly mutated to create a bigger search space. When an offspring matches a specified fitness condition, this means an acceptable solution has been found and the algorithm terminates. There are two main stages in the genetic algorithm: the selection and the crossover [23] [24].

6.1 Selection

When to select which chromosomes are to be crossed there are a few different ways. Some of these are elitism selection, roulette-wheel selection and tournament selection [25].

6.2 Crossover

It may vary which genes are carried over when two chromosomes are being crossed. To decide this there are few different methods. Some of them are single point crossover, two-point crossover and uniform crossover [25].

Genetic algorithms have been in use to solve the timetabling problem for some time now. The genetic algorithm is usually modified or used in conjunction with other techniques arrive at optimal solutions. The genetic operators are usually modified to find a solution. [26]. Encoding is the first step in formulating a solution [27]. A local search are also applied in some cases [28] [29].

[30] Developed a mixed integer linear program for the UTP. For the analysis, they converted the UTP into a three-dimensional container packing problem (3DCPP) and create a hybrid genetic algorithm (HGA), which has been shown to be efficient in solving the 3DCPP. they also develop a tabu search algorithm based on the existing UTP literature and compare the findings with that of our HGA. The results showed that their HGA obtains a better solution than the tabu search algorithm in a reasonable amount of time.

Information visibility-based university timetabling for efficient use of learning spaces introduces a new approach to construct resilient university timetables using genetic algorithms and data capturing technologies. The contributed approach adds a new dimension to solving the NP-hard timetabling and space allocation problems.

The data of the studied case of the Faculty of Commerce for the first semester in year 2012–2013 follow: In allocating events in a timetable, UGA matches the number of attendees with the closest room capacity that can accommodate them.

The IVUT proposed methodology enabled better use of resources and addressed the university dynamic timetabling problem in a very practical manner considering joint space reallocation decisions. Accordingly, occupancy rates per rooms were improved [31]. The use of RFID has produced a more natural source of information on which rooms are most preferred and most utilized, this also enables the system to easily learn. However, this approach works well in an environment where a timetabling system already runs.

Table 1. : Summary Table

Algorithm /Research	Learning aspect	Human Agent inclusion	Automated Algorithm	Distance Between classes	Reference
Information visibility-based university timetabling for efficient use of learning spaces	Yes	No	Yes	No	[31]
Analyzing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem	No	No	Yes	No	[5]
Artificial bee colony (ABC) algorithm	No	No	Yes	No	[6]
Multi agent systems	No	Yes	Yes	No	[3]
Automated Scheduling system	No	No	Yes	No	[8]
High school timetabling	No	No	Yes	No	[4]
Post-publication disturbances	No	Yes	Yes	No	[9]
Object oriented approach to timetabling	Yes	Yes	Yes	No	[1]
hybrid approaches	No	No	Yes	No	[10], [11], [12]
Adaptive large neighborhood search, which is based on repetitively destroying and subsequently repairing relatively large parts	No	Yes	Yes	No	[14]
Randomized iterative improvement algorithm	No	Yes	Yes	No	[13]
Application of Particle Swarm Optimization (PSO) in university course timetabling	Yes	No	Yes	No	[19], [22], [20]
Application of genetic algorithms in university course timetabling	Yes	No	Yes	No	[23], [24], [28], [29]

The above reviewed approaches have been proven to find good solutions, in spite of their dynamically dissimilar approaches. The above table is a simple summary of the discussed approaches to timetabling. The key areas our analysis focused on are Learning aspect (ability of the solution to make use of previously information), Human Agent inclusion (ability of the human agent to make preferred soft constraint solutions), Automated Algorithm (is the algorithm automated to do the timetabling by itself) and Distance Between classes (ability of the solution to consider the rooms that

are closest to each other be used for particular groupings to avoid time wastage in between classes). Therefore, “Yes” means the approach includes the key are we have analyzed and “No” means it is not considered or it is unclear.

6.3 Methodology

There are many different ways to approach the research that satisfies the requirements needed to obtain scientific results. It is important to consider the expectations and possibilities concerning research in your own field. In this research, we are applying the use of Simulation research.

Simulation research: Computer simulations can be used in training, teaching or entertainment. When it comes to research, According to [32], [33] simulation can be used to get a better understanding of a phenomenon of interest and for the purposes of prediction. They also argue that simulation is valuable for social science as a tool for formalizing theory. As far as [32] are concerned the process of specifying and building the computer simulation which ‘involves being precise about what the theory means and making sure it is complete and coherent, is a very valuable discipline in its own right’. Simulation, they suggest, has advantages over traditional mathematical modelling when the interest is in processes and mechanisms rather than associations between variables. In addition, [33] noted simulation may be particularly suitable when dealing with complex, non-linear phenomena. In research terms, simulations have long been employed in process analysis and evaluation in operations research, investigating diverse topics such as patient appointments in healthcare, supply chain dynamics and production control systems [32].

Research Design: With the above understanding this research will apply the use of the following steps according to [32], [33]:

Research question: Identify a research question that is suitable for study by simulation.

Model design: Model design involves specification of the target to be modelled in the simulation and the selection of an appropriate simulation method. There are a number of different methods from which to choose depending on the problem being investigated. Model design will usually involve some data collection to inform the parameters for the model and the initial conditions for the simulation.

Model building: The next step is building the simulation model. A number of software programs are now available to support specific simulation methods but if no suitable software package is available, you will have to write the program yourself.

Model verification: Verification involves running the simulation and testing whether or not the model is working as it should. If there any problems with the simulation these should be corrected.

Run the simulation: Simulations can be thought of as virtual experiments during which you run a series of ‘experiments’ under different conditions that can be varied as required [34]. [35] Identify five elements to such an experiment: the initial conditions, the time structure, outcome measurement, the number of iterations and any variation in model parameters or initial conditions. Variation allows different assump-

tions to be tested in order to answer the research questions and also to test the sensitivity of the model to changes in parameters.

Model validation: Validation involves confirming that the simulation is a good representation of the chosen target. This can be done by comparing results of the simulation with empirical data. Validation can be a challenging process due to the nature of simulation and potential limitations on available empirical data. Nevertheless, as [36] highlights, it is important that the model is sufficiently credible that people are confident to act on the insights it produces. Credibility, he suggests, is established over time by the model-building process, the actions of the researchers and the insights offered by the simulation.

Findings and conclusions: As with other research designs, your findings and conclusions should be formulated in response to the research questions and the results should be disseminated. [32] Note that providing enough detail for the study to be replicated while avoiding burying the reader in detail can be a particular challenge when reporting simulation research.

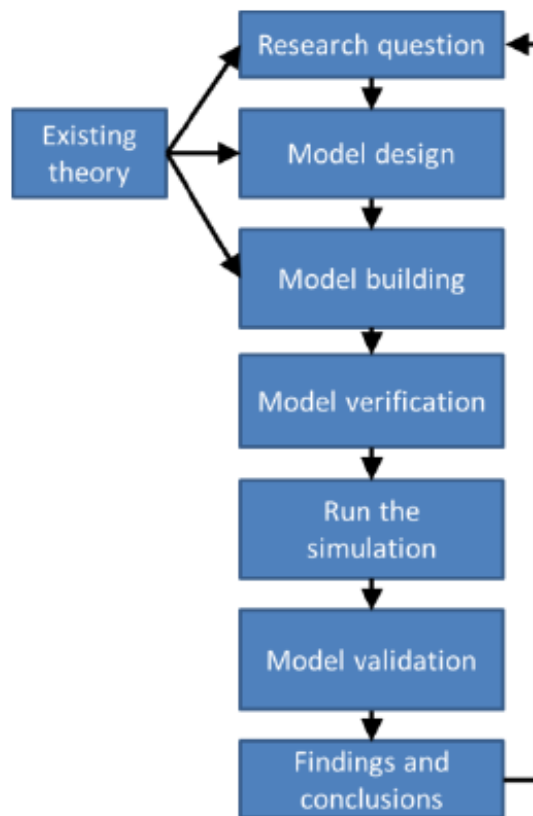


Fig. 1. Simulation Components

7 Proposed Optimal time tabling Model

7.1 Heuristics for time slot assignment

Using a Local Solution Search which finds the best suitable time slot, based on the following: The least capacity capable available room time slot is selected, in the order of time for the first Two or Three times slots and for the last Two or three, a random day besides the day the first two or three earlier inserted time slots appear.

7.2 Generic local search algorithm:

- Generate an initial solution $\rightarrow s_0$.
- Current solution $s_i = s_0$.
- Pick $s_j \in V(s_i)$.
- If $f(s_j) < f(s_i)$, then $s_i = s_j$.
- Else, $V(s_i) = V(s_i) - s_j$.
- If $V(s_i) \neq \emptyset$, then go to 3.
- Else, END. Local optimal solution = s_i . [37]

7.3 Assumptions of test

Assumption include:

- The datasets to be used are assumed to be complete and usable practically.
- The simulations are based on the assumption the datasets are being derived from live environments.

[38] gives decisive proof that the issue of producing course plans is an exemplary case of constraint fulfillment. Our concern has emerged with regards to a university in Zambia. It comprises of an arrangement of courses to be planned for 50 timeslots crosswise over five days and Ten (2 hour) durations. Toward the start of another semester, a few courses are offered to understudies (classes) and each course can be separated into different segments as per the quantity of students enrolled. Our issues are ordered into two sorts of limitations: hard constraints and soft constraints.

7.4 Constraints:

Hard constraints must be satisfied in order to generate feasible solutions. Soft constraints are sometimes referred as preferences that can be contravened if necessary. That is, schedule generation can be optimized by finding solutions that violate the minimal number of preferences for lecturers and classes. It has been established that it difficult to impossible to generate course schedules that satisfy all lecturers' preferences [38], especially with limited resources. Moreover, there are 14 hard constraints and 18 soft constraints that have been considered by previous research and also can be found in [39]. The hard constraints in this study are described as follows:

- Each lecturer can only teach one course at a time.
- Each lecturer can teach one or more courses in a semester
- Each course can only be taught by a single lecturer per time period.
- Students in a program can only attend one course at a time.
- Programs can share courses one or more courses in a semester.
- There are limited number of classroom and each classroom cannot be used more than one course at a time period.
- Certain time periods cannot be scheduled for academic activities, such as lunch time and sport.
- In addition, some other soft constraints to be considered:
- Lecturers and students (classes) can indicate their preferences, along with their preferred days and time periods in 50 timeslots across five days and Ten periods.
- Lecturers can choose to maximize the number of teaching free time periods; i.e., lecturers can specify time periods when they prefer not to lecture.
- A course should be scheduled for 2 consecutive hours if the total number of teaching hours is even or is preferred to have 2 consecutive hours.
- Minimizing students' movement between rooms i.e. movement from one campus to the next or one zone of a campus to the next zones within the campus.
- There should be at least a one-hour break between two courses of a lecturer.
- Some teachers prefer certain times or days for teaching i.e. Monday afternoon is reserved for professors' meetings: Do not schedule professors' courses for Monday afternoon.
- Reduce clashes for repeating students.
- Ensure a course is not scheduled on a single day.
- Some courses have to be scheduled in a particular room, such as computer lab.
- When a course group is too large to fit in any single room, it only makes sense to split it into two separate groups, either have the course allocated the same timeslots in different rooms or have them in the same room but different timeslots.

8 Optimization of timeslot picking

The timetable optimization problem can be seen as an extension to the timetable feasibility problem. Obviously, an optimal solution to the optimization problem is also a solution to the associated feasibility problem, but the reverse is generally not true [40].

Our algorithm learns from previously generated sessional timetables, this information is then analyzed and used to generate a more optimal new timetable. The rational is to model an algorithm that will have the ability learn and pick slots more optimal timeslots or more efficient time slots for lecturers.

The variables mainly being considered from previous timetabling information to determine the efficiency are as follows: the days, time slot, class room zones, classrooms, times in between lectures for each individual Lecturers and hours per lecture session. This model builds upon the assumption that as long as the given previous timetables worked well, they should be the best source of optimization information

that can be used to generate a more efficient timetable. Therefore, the more of this information we can have the better the algorithms prediction of the most efficient timeslots for each individual Lecturer.

9 Efficiency variables

Day: This is the weekday in question i.e. Monday, Tuesday, Wednesday, Thursday and Friday (Typical working days). This variable helps to analyze which week days are most favored by each particular lecturer, if found this can help the algorithm pick the most optimal days.

Time (Period): This is the time within each day i.e. 8 AM to 5 PM (Typical working hours). This variable helps to analyze which hours during the day are most favored by each individual lecturer, if found this can help the algorithm pick the most optimal times.

Time in between lectures and hours per lecture: these variables are also key in optimal timing for lecturers, once a lecturer's preferred trend is analyzed, this can significantly increase a lecturer's efficiency, as the algorithm will be able to times that are more favored.

Classrooms and Classroom Zones/ Campuses: Classrooms, Classroom Zones and campus are more influenced by the requirements course being delivered i.e. number of students in the course, type of delivery (With Labs or without). However, it is also interesting to analyze which of these resources or most favored by individual lecturers.

Assuming most of these efficiency variables are satisfactory for over 50% of lectures, the algorithm will be considered to have done a good job and delivered a more optimal and more efficient University Course timetable.

10 Analysis Methods

10.1 Bayesian decision approach

An alternative to using the empirical distribution is to first fit a mode $p(c,x|\theta)$ to the trained data D . Given this model, the decision function $c(x)$ is automatically determined from the maximal expected utility (or minimal risk) with respect to this model, as in Equations below, in which the unknown $p(\text{true}|x)$ is replaced with $p(c|x, \theta)$. [41]

There are two main approaches to fitting $p(c,x|\theta)$ to data D . We could parameterize the joint distribution using:

$$p(c,x|\theta) = p(c|x, \theta) p(x|\theta) \quad \text{Discriminative approach} \quad (5)$$

OR

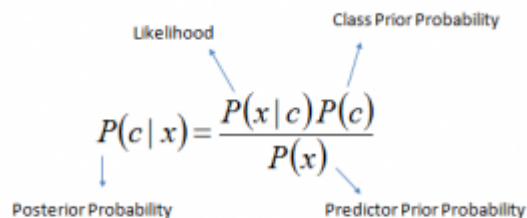
$$p(c,x|\theta) = p(x|c, \theta) p(c|\theta) \quad \text{Generative approach} \quad (6)$$

10.2 Naive Bayes algorithm

It is a classification technique based on Bayes' Theorem with a presumption of autonomy among indicators. In other words, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.[42]

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation 7 below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$


$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c) \quad (7)$$

- $P(c|x)$ is the posterior probability of class (c , target) given predictor (x , attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Advantages

- It is easy and fast to predict class of test data set. It also performs well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

10.3 Application to algorithm

This second approach is being used in comparison to see which approach will give a better result. And this approach is based on picking the most frequent days in their order, then applying the Bayesian decision approach in picking what will be the highest probable times for each day for each lecturer.

10.4 Pseudo Code for training information generation

Below is the pseudo code for generating the learning information via the Naïve Bayes approach.

```
Begin class BayesianDA
  ///declare variables
  declare List<CRoomSlot> cRoomSlotList = new ArrayList<CRoomSlot>();
  declare static CRoomSlot dayFr = new CRoomSlot();
  declare static CRoomSlot timeFr = new CRoomSlot();
  declare static CRoomSlot dayPerTimeFr = new CRoomSlot();
  declare static CRoomSlotDao cRoomSlotdao;
  declare static Bayesian naivebayes;
  declare int day;
  declare int time,frequency;
  declare double probability;

  Start method BayesianDA()
    CREATE object cRoomSlotdao
  End method

  Start method frequencyGen(for lecturer)
    call a list of all lecturer preference cRoomSlotList

    declare the total number of preferences int maxSize

    ///loop throw preferences
    start loop for each preference
      GET the frequency of a day int dayFr
```



```

        GET the frequency of the time int timeFr
        GET the frequency of a particular time on
each day int dayPerTimeFr

        CALCULATE the probability of each day dou-
ble dayfr= dayFr / maxSize;
        CALCULATE the probability of each time dou-
ble timefr= timeFr / maxSize;
        CALCULATE the probability of particular
time on each day double timeperdayfr= dayPerTimeFr / timeFr;

        CALCULATE the naivebayes probability =
timeperdayfr * timefr / dayfr;

        CREATE an object naivebayes // posterior
probability
        SAVE object into the database as a learnt
feature
    End loop
End method
End class BayesianDA
    
```

Fig. 2. : Pseudo code

10.5 Generation of the training information

The table below shows a sample of the stored learning information the algorithm makes use of during the timetabling process.

Table 2. : Sample learning information

Lecturer ID	DayID	TimeID	KeyPar	posterior probability	Day Probability
27	2	2	2-2	0.4	0.2777777777777778
27	2	3	2-3	0.19999999999999998	0.2777777777777778
27	2	5	2-5	0.39999999999999997	0.2777777777777778
27	2	5	2-5	0.39999999999999997	0.2777777777777778
27	3	1	3-1	0.5	0.11111111111111111
27	3	4	3-4	0.5	0.11111111111111111
27	4	2	4-2	0.50000000000000001	0.11111111111111111
27	4	4	4-4	0.5	0.11111111111111111
27	5	1	5-1	0.25	0.22222222222222222
27	5	2	5-2	0.25000000000000006	0.22222222222222222
27	5	6	5-6	0.25	0.22222222222222222
27	5	7	5-7	0.25	0.22222222222222222
207	3	2	3-2	0.5	0.33333333333333333

207	3	3	3-3	0.5	0.3333333333333333
207	4	1	4-1	0.5	0.3333333333333333
207	4	5	4-5	0.5	0.3333333333333333
207	5	2	5-2	0.5	0.3333333333333333
207	5	3	5-3	0.5	0.3333333333333333
183	1	3	1-3	0.5	0.3333333333333333
183	1	4	1-4	0.5	0.3333333333333333
183	2	5	2-5	1	0.1666666666666666
183	3	4	3-4	1	0.1666666666666666
183	4	4	4-4	0.5	0.3333333333333333
183	4	5	4-5	0.5	0.3333333333333333
183	1	3	1-3	0.5	0.3333333333333333
183	1	4	1-4	0.5	0.3333333333333333
183	2	5	2-5	1	0.1666666666666666
183	3	4	3-4	1	0.1666666666666666
183	4	4	4-4	0.5	0.3333333333333333
183	4	5	4-5	0.5	0.3333333333333333

11 Flow Diagram of Timetabling Algorithm with optimization

The flow diagram below, shows the different steps taken by the algorithm to generate the timetable, using the optimization information. And the following are the steps taken:

For each course, the algorithm checks which lecturer is assigned to it. Once a lecturer is found the algorithm then check for previous timetabling information about the lecturer, if no information is found, the course is then assigned a timeslot using the algorithm without optimization in figure 4 below, but if the otherwise is true, the algorithm calculates the learning information and the most suitable preferences according to the Naïve Bayes approach (in do Learn method in figure 3) are used to assign the timeslots.

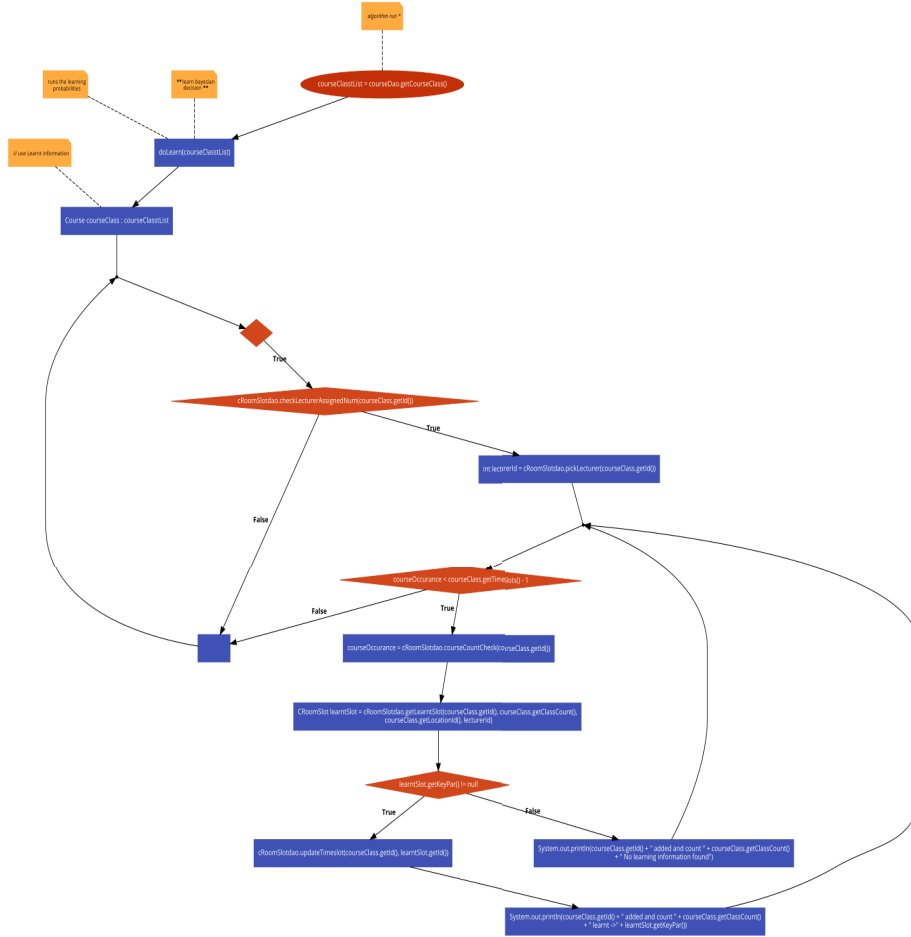


Fig. 3. Algorithm with optimization using learnt information

12 Flow Diagram without optimization

The flow diagram below, shows the different steps taken by the algorithm to generate the timetable, without any optimization information for lecturer preferences. The algorithm here only focuses on ensuring the hard constraints are resolved, with a few easy to resolve soft-constraints being considered.

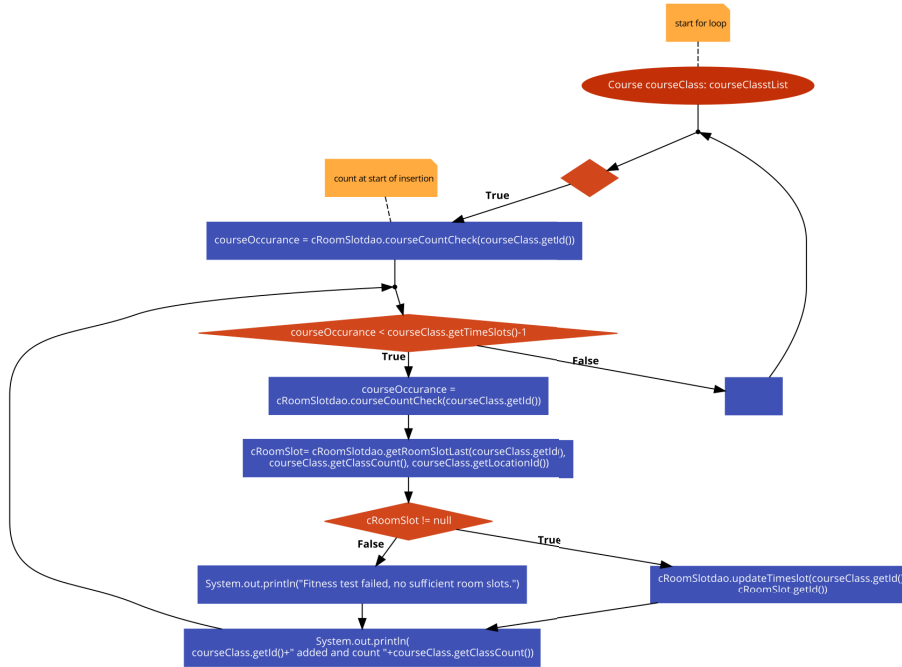


Fig. 4. Algorithm without optimization

13 Conclusion

In this research, we have presented both a mathematical and a human-machine problem that requires acceptable and controlled human input, then the algorithm gives options available without conflicting the hard constraints. The algorithm we have designed has able to learn the soft constraints over time using the Bayesian decision approach that help predict the posterior probability. Our model employs the use a naïve Bayesian Algorithm, to learn preferred days and timings by lecturers. In our future works, we will present our test results, and the impact of our model in how it improves the process of timetabling.

This study used simulation to develop the model, the key strength of simulation as highlighted by [33] Is its ability to support investigation of phenomena that are hard to research by more conventional means. [34] Highlight its potential, for example, to show the outcomes of interacting processes over time or in situations where empirical data are limited. In such situations, simulation can have advantages over analytic statistical modelling of the kind typically used, for example, in correlational studies.

Simulation-based research does, however, face some significant challenges. As [35] point out simulation is vulnerable to misspecification of the model itself. Simulation can be technically challenging and mistakes can be made in developing the computer program. They also note that generalization must be treated with caution beyond the parameters specified in the model. A further problem is that simulation methods

are not as widely understood or accepted as some other, better-known research approaches.

Future work will be to test the model on the real data from three Universities. The tests will involve finding out how long the model takes to generate the timetable, in at least five iterations, then evaluate the efficiency of the generated timetables with interviews from selected lecturers. Finally, we will evaluate the model against its ability to resolve constraints.

14 References

- [1] G. HIMAWAN and V. J. A. N. A. N. D. and Y. W. YONG, “Automated Timetabling Using an Object-oriented Scheduler,” Pergamon, p. 14, 1996.
- [2] C. Head and S. Shaban, “A heuristic approach to simultaneous course/student timetabling,” *Comput. Oper. Res.*, vol. 34, no. 4, pp. 919–933, 2007. <https://doi.org/10.1016/j.cor.2005.05.015>
- [3] B. Hamed and J. K. and A. Hadidi, “A survey of approaches for university course timetabling problem,” in *Computers & Industrial Engineering*, 2014, pp. 43–59.
- [4] Á. P. Dorneles, O. C. B. de Araújo, and L. S. Buriol, “A fix-and-optimize heuristic for the high school timetabling problem,” *Comput. Oper. Res.*, vol. 52, no. Part A, pp. 29–38, 2014.
- [5] R. L. and J. Thompson, “Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem,” *Eur. J. Oper. Res.*, p. 12, 2014.
- [6] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, “University course timetabling using hybridized artificial bee colony with hill climbing optimizer,” *J. Comput. Sci.*, vol. 5, no. 5, pp. 809–818, 2014. <https://doi.org/10.1016/j.jocs.2014.04.002>
- [7] D. Karaboga and B. Basturk, “On the performance of artificial bee colony (ABC) algorithm,” *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008. <https://doi.org/10.1016/j.asoc.2007.05.007>
- [8] C. Fiarni, A. S. Gunawan, Ricky, H. Maharani, and H. Kurniawan, “Automated Scheduling System for Thesis and Project Presentation Using Forward Chaining Method with Dynamic Allocation Resources,” *Procedia Comput. Sci.*, vol. 72, no. Supplement C, pp. 209–216, 2015.
- [9] M. V. and I. F. Vis, *School timetabling problem under disturbances.*, 2016.
- [10] S. Abdullah, E. K. Burke, and B. McCollum, “A hybrid evolutionary approach to the university course timetabling problem,” in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 1764–1768. <https://doi.org/10.1109/CEC.2007.4424686>
- [11] S. L. Goh, G. Kendall, and N. R. Sabar, “Improved local search approaches to solve the post enrolment course timetabling problem,” *Eur. J. Oper. Res.*, vol. 261, no. 1, pp. 17–29, 2017. <https://doi.org/10.1016/j.ejor.2017.01.040>
- [12] E. Norgren and J. Jonasson, *Investigating a Genetic Algorithm-Simulated Annealing Hybrid Applied to University Course Timetabling Problem : A Comparative Study Between Simulated Annealing Initialized with Genetic Algorithm, Genetic Algorithm and Simulated Annealing*. KTH, School of Computer Science and Communication (CSC), 2016.
- [13] A. S., B. E.K., and M. B., “Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem,” *Oper. Res. Sci. Interfaces Ser.*, vol. 39, 2007.
- [14] A. Kiefer, R. F. Hartl, and A. A. O. R. Schnell, “Adaptive large neighborhood search for the curriculum-based course timetabling problem,” vol. 252, 2017.

- [15] Lixi Zhang and SimKim Lau, “Constructing university timetable using constraint satisfaction programming approach,” 2005, vol. 2, pp. 55–60.
- [16] Shu-Chuan Chu, Yi-Tin Chen, and Jiun-Huei Ho, “Timetable Scheduling Using Particle Swarm Optimization,” 2006, vol. 3, pp. 324–327.
- [17] W. Tan, “A new examination timetabling algorithm,” in 16th Proceeding of National Symposium Mathematics and Science, 2008.
- [18] D. R. Fealko, “Evaluating particle swarm intelligence techniques for solving university examination timetabling problems,” 2005.
- [19] H. S. F. Irene, S. Deris, M. Hashim, and S. Zaiton, “University course timetable planning using hybrid particle swarm optimization,” 2009, p. 239. <https://doi.org/10.1145/1543834.1543868>
- [20] D.-F. Shiau, “A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences,” *Expert Syst. Appl.*, vol. 38, no. 1, pp. 235–248, Jan. 2011. <https://doi.org/10.1016/j.eswa.2010.06.051>
- [21] J. Kennedy and R. Eberhart, “Particle swarm optimization,” 1995, vol. 4, pp. 1942–1948.
- [22] R.-M. Chen and H.-F. Shih, “Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search,” *Algorithms*, vol. 6, no. 2, pp. 227–244, Apr. 2013. <https://doi.org/10.3390/a6020227>
- [23] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, Mass: MIT Press, 1996.
- [24] J. Henry Obit, “Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems,” University of Nottingham, 2010.
- [25] R. Fredrikson and J. Dahl, *A comparative study between a simulated annealing and a genetic algorithm for solving a university timetabling problem*. KTH, School of Computer Science and Communication (CSC), 2016.
- [26] A. F. AbouElhamayed, A. S. Mahmoud, T. T. Shaaban, C. Salama, and A. H. Yousef, “An enhanced genetic algorithm-based timetabling system with incremental changes,” 2016, pp. 122–127.
- [27] E. Yu and K.-S. Sung, “A genetic algorithm for a university weekly courses timetabling problem,” *Int. Trans. Oper. Res.*, vol. 9, no. 6, pp. 703–717, Nov. 2002. <https://doi.org/10.1111/1475-3995.00383>
- [28] S. Abdullah and H. Turabieh, “Generating University Course Timetable Using Genetic Algorithms and Local Search,” 2008, pp. 254–260.
- [29] S. Yang and S. N. Jat, “Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 41, no. 1, pp. 93–106, Jan. 2011.
- [30] E. A. A. and G. A. El Khayat, “An information visibility-based university timetabling for efficient use of learning spaces (IVUT),” *Egypt. Inform. J.*, p. 11, 2016. X. Feng, Y. Lee, and I. Moon, “An integer program and a hybrid genetic algorithm for the university timetabling problem,” *Optim. Methods Softw.*, vol. 32, no. 3, pp. 625–649, May 2017. <https://doi.org/10.1080/10556788.2016.1233970>
- [31] N. G. and K. G. Troitzsch, *Simulation for the social scientist*. Open University Press, 2005.
- [32] S. Rose and N. S. and A. I. Canhoto, “Management Research: Applying the Principles,” in *Management Research: Applying the Principles*, Reading: Routledge, 2015.
- [33] J. P. Davis and K. M. E. and C. B. Bingham, “Developing theory through simulation methods,” *Acad. Manage. Rev.*, vol. 32, no. 2, pp. 480–499, 2007. <https://doi.org/10.5465/amr.2007.24351453>
- [34] J. R. Harrison, L. N. Zhiang, and G. R. C. and K. M. Carley, “Simulation modeling in organizational and management research,” *Acad. Manage. Rev.*, vol. 32, no. 4, pp. 1229–1245, 2007. <https://doi.org/10.5465/amr.2007.26586485>

- [35] M. Pidd, *Complementarity in systems modelling*. Chichester: Wiley, 2004.
- [36] J. F. Oliveira and M. A. Carravilla–FEUP, “Heuristics and Local Search,” Dalam *Http-paginas Fe Pt~ MacensinodocsORHow ToSolveItConstructiveHeuristicsForTheTSP Pdf Diakses*, vol. 5, 2009.
- [37] D.-F. Shiau, “A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences,” *Expert Syst. Appl.*, vol. 38, no. 1, pp. 235–248, Jan. 2011. <https://doi.org/10.1016/j.eswa.2010.06.051>
- [38] P. Pongcharoen, W. Promtet, P. Yenradee, and C. Hicks, “Stochastic Optimisation Time-tabling Tool for university course scheduling,” *Int. J. Prod. Econ.*, vol. 112, no. 2, pp. 903–918, Apr. 2008. <https://doi.org/10.1016/j.ijpe.2007.07.009>
- [39] R. M. Goverde, *Punctuality of railway operations and timetable stability analysis*. Netherlands TRAIL Research School, 2005.
- [40] D. Barber, *Bayesian reasoning and machine learning*. Cambridge ; New York: Cambridge University Press, 2012.
- [41] “6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python),” *Analytics Vidhya*, 11-Sep-2017.

15 Authors

Alinaswe Siame is with Mulungushi University, School of Engineering Science and technology, Kabwe, Zambia alinaswe7@gmail.com

Douglas Kunda (PHD) is with Mulungushi University, School of Engineering-Science and technology, Kabwe, Zambiadkunda@mu.ac.zm).

Article submitted 13 April 2018. Final acceptance 22 July 2018. Final version published as submitted by the authors.