Safwan M. Shatnawi, Fawzi Albalooshi, Khaleel Rababa'h / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
www.ijera.com Vol. 2, Issue 4, July-August 2012, pp.1638-1644

Generating Timetable and Students schedule based on data mining techniques

Safwan M. Shatnawi[1]
Administrative and Technical Programs
College of Applied Studies
University of Bahrain

Fawzi Albalooshi
Quality Assurance Authority for Education and Training (QAAET),
The Higher Education Review Unit (HERU)

Khaleel Rababa'h
Administrative and Technical Programs
College of Applied Studies
University of Bahrain

Abstract

In this paper we introduce a new technique for generating an Academic Programme timetable based on clustering method using data mining techniques for dynamically forming students' clusters; then we use a heuristic function to find the optimal solution. The FP-tree based generated clusters are used as an initial solution to the timetable problem; then we generate more optimized clusters using color mapping algorithm; as a result a students' timetables also generated, all hard constraints are satisfied, and soft constraints are considered while generating the timetable and the student schedule. We tested our approach against real data obtained from the College of Applied Studies in University of Bahrain.

Keywords: *Automated Timetabling, Course Scheduling, data mining, FP-Tree, and Cluster, Color mapping*

1. INTRODUCTION

Course scheduling is one of the challenging time consuming problems facing institutions belonging to the NP-complete class of problems. Our main challenge is to be able to automatically time table two year associate degree programs' courses' so that students belonging to different programs can easily register for courses with no timetable clashes for the semester they are studying for. Hard constraints are: the students can only be scheduled to one event in a time slot; event rooms meet all required features and their capacity is respected; no more than one event is allowed per room and per timeslot. Soft constraints include: to avoid scheduling classes in the last timeslot of the day; to avoid scheduling more than two classes in a row for a student; and avoid scheduling one class in a day for a student.

A general survey paper by [1] investigates examination and course timetabling providing up to-date important information and citations for further research and possible implementations of automated timetabling for use in educational settings. A more comprehensive survey carried by [2] confirms that a general polynomial bounded algorithm for solving time tabling problems cannot be found. However there has been large interest in applying meta-heuristic-based algorithms that are general purpose algorithms. The algorithms were mainly based on three techniques; graph coloring; constraint-programming; and integer programming. The author further categorizes the algorithm into one-staged, two-staged, and algorithms that allow relaxations. In one-staged optimization algorithms hard and soft constraints are allowed to be violated and the algorithm then aims to search for a solution that satisfies both as much as possible examples of such algorithms are those reported by [3] and [4]. Two-staged optimization algorithms attempt to satisfy the hard constraints until an optimization is reached. The soft constraints are then considered for optimization as much as possible. Examples of such algorithms are those proposed by [5] and [6]. The third class of algorithms allows relaxations of some constraints without violating the hard ones. Examples of such are proposed [7] and [8]. In his survey [9] concentrated on the common solution approaches including graph coloring, integer programming, network flow techniques, tabu search, rule-based approach, and constraint logic programming.

A 3-phase approach succeeded to become the winning entry in the International Timetabling Competition is presented by [10]. At first the timetable is created using graph coloring and maximum matching algorithm. In the second phase an attempt is made to satisfy more software constraints using simulated annealing (SA), and in the third phase the solution quality is improved by swapping individual events between time slots using SA. Burke an active researcher in this field who published many articles on timetabling presents a generally applicable approach for constructing examinations and course timetables using a generic hyper-heuristic approach [11]. The authors test their approach on a simulated benchmark proposed by the Meta-heuristic Network [12] for course timetabling problems and compare it with other state of art approaches. Reference [13] investigates automated timetabling to come-up with a solution for the Purdue University class timetabling problem. He investigates a number of common solutions and presents an iterative forward search algorithm that

became the basis of his winning entry in the 2007 timetabling competition [14] organized by the timetabling research community.

Clustering technique is another approach used for timetabling; clustering is the process of finding classes of objects that share common characteristics [15]. Clustering is mainly based on splitting the events into clusters or groups were each cluster is scheduled in the same timeslots without having conflicts. Clustering methods satisfy the soft constrains using additional optimizing rules for obtaining good solution, in order to use this approach the courses are grouped into fixed clusters at early stages of the algorithm; the formation of the cluster is done manually based on some predefined rules which leads to poor quality timetable [16].

An FP-tree is a compressed representation of the input data (transactions). It is constructed by reading the data set one transaction at a time and mapping each transaction onto a path in the FP-tree [15]. FP-tree technique has been used in many applications such as clustering and document organization. In this paper we customize a FP-tree algorithm to dynamically generate clusters from the students transactions (courses to be registered for the coming semester)

2. WEEKLY TIME TABLE

The week consists of five study days each staring at eight in the morning till five in the evening. Lectures and laboratory sessions are regularly timetabled either on Sunday (U), Tuesday (T), and Thursday (H) sessions or Monday (M) and Wednesday (W) sessions. A course can have one or more sections depending on the number of students that are expected to register for it and each section can take up to 25 students. For example if 50 students are expect to register for a course two sections for the course need to be offered. They can be at the same or different timings. A section is timetabled at fixed times over the study week. For example, a three weekly lecture course can be timetabled nine to ten on UTH so that the start and end of sessions is at the same time on different days of the week. Weekly contact hours for a course section can vary but the norm is five (two lecture and three laboratory sessions) and three (three lecture sessions)

3. DATA PREPROCESSING

3.1 Students Advising and Courses Statistics Generation System

In previous work [16] we developed an online advising system that can be utilized by students, advisors, and course timetable planners. Students are given informative advice on which courses to register for in the next semester and are informed of their remaining graduation requirements; advisors are able to see students' progress towards their graduation requirements; and timetable planners are given statistics on courses and sections requirements for the coming semester.

The main source of information in our above mentioned work was the official university registration system. From it students' academic records and achievements, data were extracted and processed for advising purposes. It was important to maintain two new tables: the first holds details of courses belonging to a program and the other holds details of courses' prerequisites. Essential information for timetabling that was also generated in our same previous work was the generation of course requirements for a coming semester based on existing students' achievements, for more details please refer to [16].

In our earlier work other information was extracted from the university registration system for further processing, but we used the resulted information for the timetable generation approach described in this paper.

In order to obtain the set of courses a student will enroll in a given next semester, we apply the following procedure: for each student, we take the student's transcript including the courses a student currently enrolled in and compare it against the program study plan, we get the list of not yet registered courses and failed courses needed to be re-taken, then for each course in the resulted list we compare it against student's transcript to find if the student satisfied the prerequisites for that course. The end result is the list of courses and number of students expected to enroll in them in the following semester.

3.2 Preparing students data to FP-Tree Algorithm

In order to prepare the data for FP-tree Algorithm, first each next semester courses to be registered by students are considered as transaction; so our transactions are set of courses to be register next by all students; each student represent a transaction; the following model our approach

Let C = {c1,c2,…., cn} be set of all courses offered by the college

Let S={s1, s2,…,sn} be set of all students enrolled in the college

Let T={t1, t2,…,tn} be set of all transactions in the college; where each ti={{si,cj}+}; cj belongs to C and registered by si belonging to S.

Before starting our modified FP-tree algorithm, we prepare the students data by comparing student's transcript to his/her study plan, as a result, we obtained list of courses he/she can register in the following semester, each student represents transaction for the FP-tree algorithm tj, and each course a student can register represents an item ci, in that transaction, then we calculate the number of students who are to register for each course, the list

of courses is then sorted in descending order and called header table in the FP-tree algorithm, next courses in every transaction tj are sorted in descending order based on the number of students to register ci next semester based on the header table index, now the transactions are ready to be processed by the FP-tree algorithm; the following example demonstrates the process of preparing the students data:

Let S ={20100001, 20100002, 20100003, 20100004, 20100005} be list of students in the college

Let C={C1, C2, C3, C4, C5, C6} be list of courses to be register in the following semester

Let T ={ {20100001,C1,C2,C3}, {20100002, C2, C3,C4}, {20100003, C2,C3,C6}, {20100004, C1, C2,C6},{20100005,C2,C5}} list of courses to be registered by each student. Table 1. represents the header table.

Next all transactions in T should be reordered according to the header table index, as a result, we obtained the following:

T={ {20100001,C2,C3,C1}, {20100002,C2,C3,C4}, {20100003, C2,C3,C6}, {20100004,C2,C1,C6}, {20100005,C2,C5}}. Now the data is ready to be processed by the FP-tree algorithm. Table 1. Shows the FP-Tree header table for the aforementioned example

Table 1 FP Tree Header Table

Course	Count		Course	Count
C1	2		C2	5
C2	5		C3	3
C3	3	➔	C1	2
C4	1		C4	1
C5	1		C5	1
C6	1		C6	1

3.3. Information Representation

FP-Tree Structure: Students transactions are represented as tree structure; where each node in the tree contains the course to register for, the number of students who will register this course , links to the parent of the node, and an array of links to the node's children; while the original FP-tree doesn't care about the owner of the transaction (student), we modify this structure since we need to keep track of who will register this course to avoid conflicts in the latter processing of the FP-tree results; so we add an array of students who will register the course next to each FP-tree node.

Weekly Time Slots structure: As we mentioned earlier in section 2 the week is represented by 39 slots distributed as 27 one hour slots for UTH and 12 one hour and half Slots for MW, we represented the week slots into two matrixes the first matrix for UTH slots and the second matrix for MW slots. Since the lab session is spanned over 3 hours, we divide the UTH slots into 3 parts and the MW slots into 3 parts also(as well).

Courses structure: Each course structure contains the course code, credit hours, classes, and lab session information.

Students' information: for the sake of our modified FP-tree algorithm, we need only the *student ID* for processing the transactions; however we store the student basic information like name and program name for generating meaningful reports.

Timeslots: the aforementioned slots are grouped into fixed group where each group will be mapped to one level that is generated by applying our modified algorithm

4. TIME TABLE GENERATION PROCESS

4.1 Phase I: FP-Tree generating

The algorithm scans the transactions and builds the FP-tree structure described earlier; figure 1 describes the FP-tree algorithm used to generate FP-tree structure. To form the clusters we processed the FP-tree generated by the original FP-tree algorithm in order to minimize the clusters generated by the original FP-tree algorithm, the algorithm used to cluster the FP-tree is described in figure 2. This algorithm is described in details in[17]. After modifying fp-tree we obtained the clusters which are displayed in figure 3.

1) FP-Tree Results and analysis

After applying FP-tree algorithm, we obtained clusters; where each cluster consists of a group of courses; clusters are not balanced, i.e. number of courses in each cluster (levels) are not equal, even more the number of students register in each course inside the cluster are not equal, moreover the same course can be found in different clusters with different attributes. If we say that n is the average student in each section; where N >10 and N < room size, then any course in the cluster can have M students, where M is one of the following cases:

M < 10: means that this section cannot be opened and since the number of students are less than the minimum allocation for a section size, but we still can find another cluster that has the same course and we can merge these two courses together to open section, this requires the algorithm to search for another course that can be merged with.

10 < M < N: this section can be opened and no more processing is required.

1. Scan the transaction database for the first time.
   1.1 Find frequent items (single item patterns) and sort them into a list L in decreasing support count.
2. For each transaction ti ε T, order its frequent items ci ε C according to the order in L.
   2.1 The algorithm makes a second scan over the database to construct FP-tree by

reading each transaction in step 2 as a branch on the tree.

Figure XX : FP-Tree Algorithm

M > N: this section should be split over two or more sections; again we have to satisfy all hard constraints before splitting this section.

The generated clusters come from courses that is distributed over FP-tree levels where all courses in a given level can be schedule in the same time slots without getting clashes, however as we mentioned above, the number of students in each course may be divided into two or more sections, as a result, we need to move the generated sections into multiple levels to fulfill the business soft constraint, which states that no more than two sections should be scheduled at the same time slots, so we need to move the new generated sections into different levels without causing any clashes, another issue in the generated clusters that the number of clusters may span over the available.

A. void cluster (fptree) {
1. For each Node in FP-tree starting from root in FP-tree level by level do:
  1.1. For each CH1 in Node's children do
    a. For each CH2 in Node's children do
      If CH1->item == CH2->item then Move (CH2, CH1)
    b. Next CH2
  1.2. Next CH1
  1.3. For each CH in Node's children do:
    1.3.1. Let source = CH
    1.3.2. Find CH->item header table entry
    1.3.3. For each HL in header table links
      a. If (HL->node != source) &&
        (HL->predecessor == source->parent) then
        i. Move(source,HL->node)
        ii. source =HL->node
      b. End If
    1.3.4. Next HL
  1.4. Next CH
2. Next Node}
B. void Move (node source, node target) {
 1. target -> item-count += source -> item-count
 2. For each CH in source's children do:
  2.1. create a link from the target node to CH
  2.2. CH->parent = target
  2.3. delete source's link in the header table
  2.4. delete source
 3. Next CH}

Figure YY: FP-tree based clustering algorithm

week slots, so again we need to minimize the number of clusters without causing any clashes. Hence It is obvious that the FP-tree algorithm cluster the students, but still it is not the optimal solution for the problem or even a solution to the problem; in other words the FP-tree algorithm just redefines the problem in a way or another.

It is guarantee that each course exists in any cluster's level can be registered in the same time without clashing, however we still can minimize the number of the cluster levels (depth of the cluster) by applying a color mapping algorithm or an appropriate searching algorithm; in this paper we used color mapping algorithm.

The depth of each cluster is not guaranteed not to exceed the maximum levels count (depth); where the maximum depth should not exceed the week's slots; the depth of the clusters (levels) in phase II of this methodology are minimized and controlled The number of different registration combination, i.e. different paths in the FP-tree structure can be obtained by the following rules:

Let ***M*** be number of courses that are offered by the college; M any positive integer

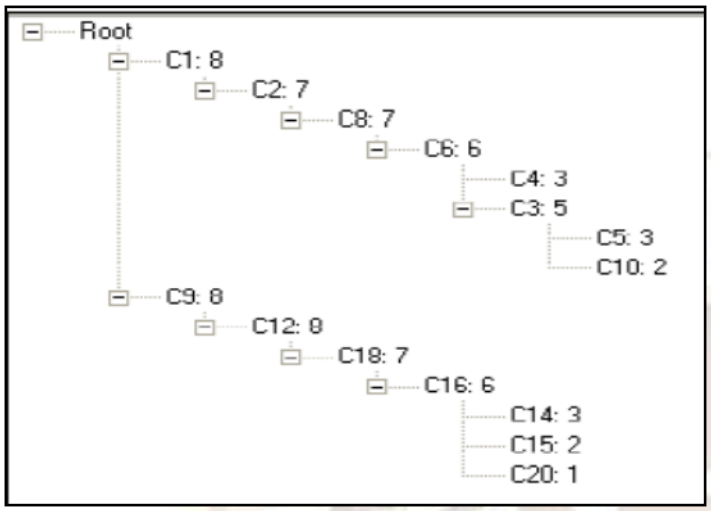
Let ***Z*** be the number of students registered in the college; ***Z*** any positive integer

Let ***N*** be the maximum possible load; ***N*** <=***M***, then the different students registration combinations will be (***SRC***) ={

Case 1:
if ***Z*** > $\frac{M \bullet (M-1) \bullet (M-2).....(M-(N-1))}{N!}$ *then*
SRC= $\binom{M}{N} = \frac{N!M!}{(M-N)! \bullet N!}$
Case 2 : $= \frac{M \bullet (M-1) \bullet (M-2).....(M-(N-1))}{N!}$
if ***Z*** < $\frac{M \bullet (M-1) \bullet (M-2).....(M-(N-1))}{N!}$ *then*
SRC = Z }

Maximum FP-tree depth = ***N***
Maximum modified FP-tree level = ***M***
Maximum levels in the week = time slots /maximum course slot
Optimal number of cluster = ***N***

Figure 3: Obtained Clusters



2) ***Optimality in timetabling***

Optimal solution for timetabling problem is domain dependent, so before searching for the optimal solution we should first define optimality, in our study domain we have several parameters that define optimality, of course the hard constraints is the base to find the solution, but the soft constraints represent the optimal or desired solution, following are some parameters for defining optimality:

- Number of internal time slots fragmentation in the student schedule ; minimize
- Number of sections (classes) of the same course in each time slot; (minimize)
- Utilization of timeslots ; (maximize)
- Utilization of resources (maximize)
- Internal fragmentation in each resource; (minimize)
- Total number of sections; (minimize)

3) Phase II: Processing modified FP-Tree for timetable generating

Starting from the fact that the result of the modified FP-tree algorithm can further be optimized, we will start by minimizing the resulted cluster; a color mapping algorithm is used to minimize the clusters and the nodes (sections) in each cluster; the task of this phase is to minimize the depth of the obtained clusters in phase I; and to form clusters to be processed by phase III; in this process if we found multiple instances of a course in a level, we merge these instances into one instance; again the formed cluster has the same properties described in section 4.1.1. Figure 4 shows the color mapping algorithm used in this phase.

4) Phase III: Generating sections and assigning timeslots.

We obtained courses' clusters where each cluster contains courses that can be registered in the same timeslots without causing clashes in the students' timetable; heuristic function based on the optimality guidelines described in section (optimality in timetabling) is applied to generate optimized timetable. Figure WW shows the screen snap shot for a given program timetable (form). In this phase we assigned room and lab for each section while satisfying the no conflict hard constraints and the number of students less than the room's size which is an easy and direct process.

5. EXPERIMENTAL WORK AND ANALYSIS

In order to evaluate our approach, we tested our approach using real data obtained from the College of applied studies; the sample contains information about 1270 students distributed among 8 academic programs; each student may register up to 5 courses selected among 83

```
Color mapping pseudo code
Courses=N
Colors =0
For each Level(Vertex)
Color i= Li;
Colors ++; courses --;
For all Courses Ci ε C  in Li then  Color i ← Ci
Next level
For each level
Students = number of students in Ci
Section size = M
  If students > section size then
          Split Ci into multiple sections
          For each generated section Si
    Find level Li; Li ∩ Si =0 then
            Color Li ← Si;
level color
Next level
Courses=0;
Colors=N;
```

Figure 4. Color mapping Algorithm

courses, Table 2 shows the statistics of the experimental work; the estimated required sections are obtained based on the assumption that the section size is 25 students; first we obtained the courses to be register next for all students, then we prepared the transaction file where each student next semester courses represents a transaction and based on the number of students who will register each course we built the header table mentioned in section 4.1; next we apply FP-tree algorithm to obtain the transactions' tree; after that we minimize the number of clusters by applying out modified algorithm mentioned in figure 2, the number of obtained clusters was 25 clusters; to minimize the number of clusters we apply the color mapping algorithm mentioned in figure 4, as a result we obtained minimized number of clusters; for our data we obtained 10 clusters; the final step is to

generate sections and then assign each level to a pre fixed time slots; figure 5 shows a snap shot for ACCA221 section 1 class list, the time slots are group in a way that there is no clashes (overlaps) between any two groups of time slots, figure 6 shows snap shot of one time table form; all hard constraints were satisfied (no time clashes; room capacity; number of sections in a given time slot,...) through the use of constraints based approached were any section being generated should not violate any of the hard constraints.

Table 2 Experimental statistics

Item	Statistics	Generated
Number of students	1270	1270
Number of Courses	83	83
Estimated required sections (optimal)	167	183
Student Conflict		0

6. CONCLUSION

Timetable based on intended courses to be registered next semester is generated, our approach goes beyond time conflicts constraints only to enable students to register all courses they want, our approach is student oriented and aims to provide the student with efficient(based on the students definition) timetable, our algorithm has advantages over static clustering algorithms; where the clusters are formed and fixed before finding the solution; whereas our approach allows dynamic formation of the clusters and minimize the search (solution) space for color mapping algorithm. since the obtained clusters less than the available weekly timeslots, the students can have better timetable with minimum internal fragmentation timeslots, which also makes the instructors timetable efficient , the color mapping algorithm makes use of the generated clusters which minimize the search space for the color mapping algorithm.

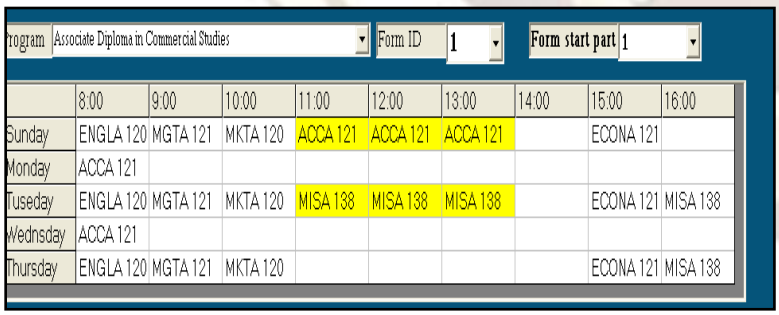


Figure 5. Course /Student Enrollment

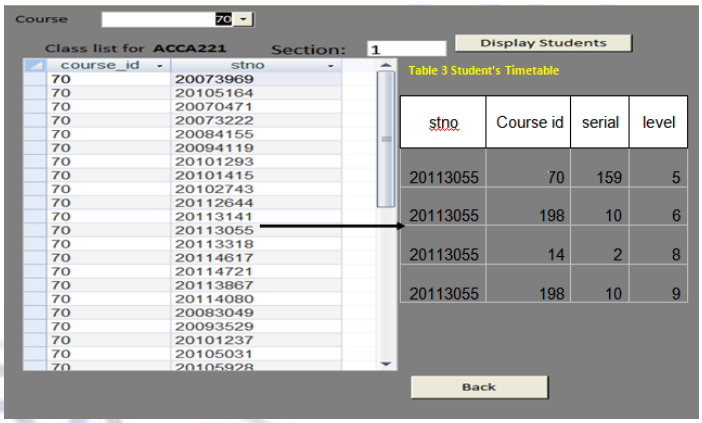


Figure 6. Screen Snapshot of one timetable form.

ACKNOWLEGEMENT

This project is supported by Deanship of Academic research at University of Bahrain , underproject number 23-2010

References

[1] M. B., "University timetabling: Bridging the gap between research and practice (invited paper)," in The 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), Brno, 2006.

[2] L. R., "A Survey of Metaheuristic-based Techniques for University Timetabling Problems," in OR Spectrum, 2008..

[3] C. M. and P. M., "A Multiobjective Genetic Algorithm for The Class/Teacher Timetabling Problem," Practice and Theory of Automated Timetabling (PATAT) III, vol. 2079, pp. 3-17, 2001.

[4] D. G. L. and S. A., "Multi-neighbourhood Local Search with Application to Course Timetabling," Practice and Theory of Automated Timetabling (PATAT) IV, vol. 2740, pp. 262-275, 2003.

[5] C. S. and T. J., "Grasping The Examination Scheduling Problem," Practice and Theory of Automated Timetabling (PATAT) IV, vol. 2740, pp. 233-244, 2003.

[6] A. H. and L. A., "A Tabu Search Heuristic for a University Timetabling Problem," Metaheuristics: Progress as Real Problem Solvers, vol. 32, pp. 65-86, 2005.

[7] E. E., "A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling," Practice and Theory of Automated Timetabling (PATAT) III, vol. 2079, pp. 132-156, 2001.

[8] C. P., W. T. and S. R., "Application of a Hybrid Multi-Objective Evolutionary Algorithm to The Uncapacitated Exam Proximity Problem," Practice and Theory of Automated Timetabling (PATAT) V, vol. 3616, pp. 294-312, 2005.

[9] S. A., "A survey of Automated Timetabling," in *Artificial Intelligence Review*, 1999.

[10] K. P., "The University Course Timetabling Problem with a 3-Phase Approach," *Practice and Theory of Automated Timetabling (PATAT) V*, vol. 3616, pp. 109-125, 2005.

[11] B. E., M. B, M. A., P. S. and Q. R., "A Graph-based Hyper-Heuristic for Educational Timetabling Problems," *European Journal of Operational Research*, vol. 167, pp. 177-192, 2007.

[12] [Online]. Available: http://www.metaheuristics.net/. [Accessed 11 August 2009].

[13] T. Muller, *Constraint-based timetabling. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics*, 2005.

[14] [Online]. Available: http://www.cs.qub.ac.uk/itc2007/ . [Accessed 11 August 2009].

[15] Pang-Ning-Tan, M. Stienback and V. Kumar, Introduction to Data Mining, Addison Wesley, Pearson Education, 2005, pp. 363-370,487- 490.

[16] F. Albalooshi and S. Shatnawi, "Online academic advising support," in *IEEE International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (EIAE 09)*, 2009.

[17] S. Safwan, A.-R. Khaleel and B.-I. Basel, "Applying a novel clustering technique based on FP-tree to university timetabling problem: A case study," in *International Conference on Computer Engineering and Systems (ICCES)*, 2010.

[18] G. M. White and P. W. Chan., "Towards the Construction of Optimal Examination Timetables," in *INFOR 17*, 1979.