AUTOMATED SYSTEMS FOR EDUCATIONAL TIMETABLING PROBLEMS

Nguyen Tan Tran Minh Khang,
Faculty of Information Technology, HCMC University of Science, Vietnam.
Tran Hue Nuong,
Faculty of Mathematics, HCMC University of Science, Vietnam.

TÓM TẮT:

Trong bài báo này, chúng tôi giới thiệu về hai hệ thống web cho phép xếp thời khóa biểu một cách tự động: hệ thống UTS (University Timetabling System) – hệ thống xếp thời khóa biểu cho trường đại học với cách học theo học phần và hệ thống HTS (High school Timetabling System) – hệ thống xếp thời khóa biểu cho trường phổ thông trung học. Bài toán được đề cập đến trong hai hệ thống là bài toán xếp thời khóa biểu cho giáo dục, trong đó, các học phần (môn học) có độ dài lớn hơn một tiết, và có thể được phân thành các cụm tiết khác nhau. Thuật giải Tabu Search được tích hợp vào hai hệ thống để giải quyết bài toán. Hai hệ thống đã được chạy thử nghiệm trên 9 bộ dữ liệu đại học và 3 bộ dữ liệu của hai trường phổ thông.

ABSTRACT:

This paper introduces UTS (University Timetabling System) and HTS (High school Timetabling System) - two web-based systems for solving the educational timetabling problems automatically. These systems support two timetabling problems named curriculum based course timetabling problem and high school timetabling problem, in which courses could have different lengths and could be divided into blocks. These two systems use Tabu Search-based algorithms as the kernel solvers and are applied to nine real-world instances of a university and three instances of two high schools in Vietnam. Results are obtained in reasonable time and the quality is promised for being used in practice.

Keywords: *course timetabling, high school timetabling, Tabu Search, metaheuristic, automated timetabling system*

1. INTRODUCTION

Educational timetabling problems are known to be hard combinatorial problems with many variants based on the concrete requirements of each university and high school. Based on the survey of Schaerf (1999b) and the technical reports of ITC2007 (the second International Timetabling Competition) in McCollum et al. (2007), educational timetabling can be classified into four groups: high school timetabling problems, curriculum based course timetabling problems, post enrolment based timetabling problems and examination timetabling problems. Problems in the high school timetabling group don't consider the room schedule while the others do. The curriculum based course timetabling problems are different to the post enrolment based timetabling on the decision of the conflict between courses: in the post enrolment based timetabling, the students will register the courses they want to study first, then the courses are scheduled in such a way that there's no pair of overlapped courses registered by the same student; while in the curriculum based course timetabling, the timetables are decided not based on the registration of the students, but on the curriculum set up by the institution. The main thing that distinguishes the examination timetabling from the others is its requirement of the

balanced workload of the students during the exam.

Due to the complexity of timetabling problems in the literature and the large number of required constraints in the practice, the tasks of creating timetables should be done automatically by machines, not by hand. Moreover, managing and distributing timetables is also a complicated job, including gathering and updating information of lecturers, students, courses; letting people see their scheduled timetables in appropriate points of view, etc. Therefore, integrated automated timetabling systems that include the appropriate user interface, utilities and efficient algorithms for institutions should be considered an urgent and realistic demand. The high school timetabling system named KTS, which is introduced in Jeffrey (2006) is an example.

This paper introduces two free web-based integrated automated timetabling systems: UTS (University Timetabling System) for curriculum based course timetabling problem and HTS (High school Timetabling System) for high school timetabling problem. The problem model and data instances for testing are taken from popular universities and high schools in Vietnam, the proposed constraints is a little specific, but they may be adapted to some other institutions with a few changes. Moreover, for convenience, there are two off-line applications that have the same functions with the online systems. Input data is represented in XML and could be used to transfer between the off-line and online systems.

The kernel solvers of these systems are based on Tabu Search algorithm – a popular metaheuristic that has been efficiently used for solving many other timetabling problems, for example in Costa (1994), Ramon et al (2002)

This paper is organized as follow: section 2 presents the data models of UTS and HTS, section 3 describes the details of educational problems considered in these two systems, section 4 shows the main features of each system, section 5 presents the general overview of the Tabu Search - based algorithms used in the solver modules of these systems, at last, section 6 presents the experiment results when applying UTS and HTS to twelve data instances of one university and two high schools in Vietnam and section 7 gives some conclusions.

2. DATA MODELS

2.1 Data model of UTS

2.1.1 User's role

In the UTS system, each user will have an account with one of the three roles: lecturer role, manager role and administration role. Users with lecturer role can change their personal information, e.g., their available time for teaching, and view their timetable once it's scheduled. Administrative user can administer other accounts. Managing users can create, manage timetables of their institutions.

2.1.2 Problem's concepts

Each timetabling problem of each institution is presented as a problem instance. The main information of an instance includes the following basic concepts:

- Time concepts: periods, sessions, days, cycles, lectures.
- Resource concepts: lecturers, classes, subjects, courses, rooms, devices.
- Group concepts: class groups, subject groups, curricula, building, device groups, time groups

The time concepts:

Each *period* often last about 45 minutes. A *session* is a group of consecutive periods that forms a part of the day, e.g., morning session includes all the periods in the morning. Users can set the number of periods per session and the number of sessions per day due to the specific instance.

A *cycle* is the total time of a timetable and is often one week. A cycle may be set more than one week based on the special requirement of lecturers, e.g., some

lecturers want to teach every two weeks, not every week, the cycle should be two weeks in this case.

Resource and group concepts:

A *time group* is also a group of periods, but not required to be successive. For example, lecturer A likes to teach in the mornings; in that case, the managing user should create a time group X which consists of all the periods in the morning and assign this time group to the preferred period list of all courses that this lecturer is involved in.

A *class* is a group of students that will attend the same courses, note that in the UTS, the class is used instead of the student concept because this is a curriculum-based problem.

A *lecture* is a part that lasts one period of a course.

A *course* is a group of lectures that have the same lecturer, the same attending classes and subject. The main information of a course includes the lecturer teaching this course, one or some attending classes (classes here are disjoint sets of students at the same level, each class could attend various courses), the subject that this lecturer will teach in this course and the number of lectures in a week that belonging to this course, i.e., the number of periods that this course must hold in a week. Each course may be split into blocks (a block is a group of consecutive lectures of the same course in a day), the size of the blocks must be in the range of [*minConsecutivePeriods, maxConsecutivePeriods*]. For example, the values of *minConsecutivePeriods* and *maxConsecutivePeriods* of the course named A are 2 and 3 respectively, and the number of lectures in a week of this course is 5. Then we could split course A into two blocks, one lasts 2 periods and the other takes 3 periods. Some courses may be *pre-assigned*, i.e., their periods and rooms are pre-determined by people, these courses will not be re-scheduled again but their pre-assigned information (e.g., the assignment of their lecturer, classes) will be considered during the timetabling process of the other courses.

A *room* contains the information of its capability and its location (the *building* that this room belongs to, this information will be considered in the soft constraint of the university problem because each building is often distanced far from the others, so there should be some constraints to eliminate the movement of lecturers and students between these buildings in the same day).

The devices include projectors, microphones, sockets … Devices of the same type are grouped into one device group.

The concepts of *class groups*, *subject groups* are used to define the curricula. A *curriculum* is a combination of one class group and one subject group. All courses that has the pair (class, subject) belong to some curriculum are prohibited to be overlapped.

2.1.3 Problem instance's representation

For the convenience, both of the information of each problem instance and its results after being scheduled by the solver module are saved in an XML file. With this kind of saving format, users could easily switch between the offline and the online system by importing timetables from and exporting timetables to XML files. Therefore, users can use UTS offline applications to create these XML files, and then upload them to the online system to spread them out to the other users throughout the internet.

The structure of the XML data file is designed based on the data model of the problem. It includes two main parts: the input data tags and the output tags (for the result timetables)

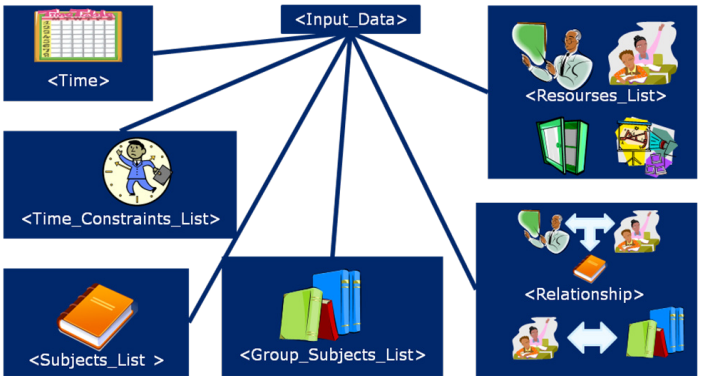

Figure 1: The general view of the tags in the XML file of UTS

The input tag contains 6 components as shown in figure 1 : the *Time* component for the definition of the time concepts, the *Time_Constraint_List* for the available periods information of lecturers, classes and the preferred time group of courses, the *Subject_List*, *Group_Subjects_List* for the definition of the subject groups and the curricula, the *Relationship* for the main information of each courses, and the *Resourses_List* for the list of devices and rooms.

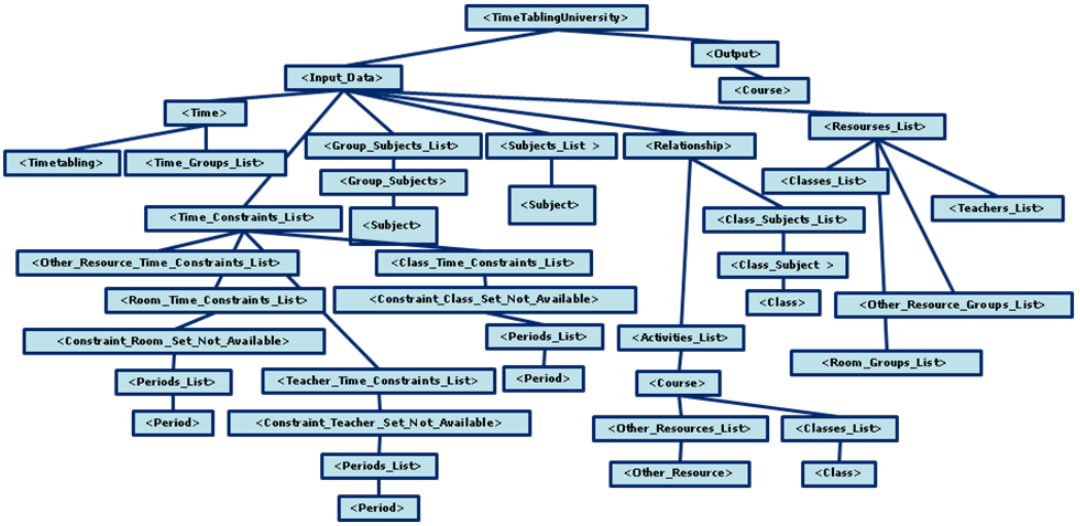

Figure 2: The detailed list of the tags in the XML file of UTS

The output tag contains all the needed information of the result after running the solver on the instance, including the list of period, room and device assignment of all courses. The detailed tree of the data structure in the XML file are listed in figure 2

2.2 Data model of HTS

HTS's data model is very similar to UTS, except that there are no concepts of subject groups, subject-class groups and concepts that are related to rooms and devices.

3. PROBLEM DESCRIPTION

3.1 Problem description of UTS

The curriculum-based university timetabling problem considered in this paper is taken from the Faculty of Information Technology, HCMC University of Science - it involves finding a weekly map between a set of courses and a set of periods, rooms and devices in such a way that satisfies the university's requirements as much as possible.

Our problem has 8 *hard constraints*, i.e., constraints that must be satisfied, and 12 *soft constraints*, i.e., constraints that should be satisfied as much as possible). Each soft constraint is associated with a weight factor.

- Hard constraints (denoted by H):

[H1] The number of a course's consecutive lectures in a day must be in the range of *[minConsecutiveLectures, maxConsecutiveLectures]*.

[H2] Lecturers, classes, rooms and devices could not be assigned to periods at which they're not available.

[H3] All the lectures of all courses must be assigned.

[H4] The pre-assignment of any course must be respected.

[H5] The capacity of a room that is assigned to a course must be equal or greater than the planned number of students attending that course.

[H6] Courses that have the same lecturer or class must not overlap.

[H7] No room or device could be assigned to two simultaneous courses.

[H8] The consecutive lectures of each course in a day must belong to the same session

- Soft constraints: soft constraints are divided into two groups: high weighted soft constraint group (denoted by HS), which includes 1 constraint, and low weighted soft constraint group (denoted by LS), which includes 11 constraints:

[HS1] Courses that belong to the same curriculum should not be overlapped.

[LS1] The movement of a lecturer between two buildings in the same day should be avoided.

[LS2] The movement of a class between two buildings in the same day should be avoided.

[LS3] Courses should be assigned to periods that they're preferred.

[LS4] The idle time between consecutive courses in the same session for a lecturer or a class should be avoided.

[LS5] The number of periods that a class is assigned in a day should be equal or less than *maxStudyingPeriods*.

[LS6] The idle time between consecutive courses in the same day for a lecturer or a class should be avoided.

[LS7] The number of sessions that a lecturer was assigned to should be as small as possible.

[LS8] The number of periods that a lecturer is assigned in a day should be equal or less than *maxTeachingPeriods*.

[LS9] The number of days that a class was assigned to should be as small as possible.

[LS10] In the timetable of a class in session, if there are three periods in which this class is idle, these should be consecutive periods (this is a special constraint requested by the timetabling staff because they need use these idle periods for another purpose).

[LS11] All courses that belong to the same curriculum which are scheduled to the same session should be assigned to the same building at that session.

(Note that each course has its own *minConsecutiveLectures*, *maxConsecutiveLectures* parameters).

The linear objective function $f(q)$ is used to evaluate the quality a timetable q:

$$f(q) = \sum_{i=1}^{n} w_i d_i$$

w_i and d_i are the weight and the violation number of the i^{th} soft constraint, n is the total number of soft constraints.

3.2 Problem description of HTS

The high school timetabling problem of HTS originates from the real-world

problems of two high schools: Ho Chi Minh's high school for the Gifted and Tran Phu high school. High school timetabling problem of HTS involves finding a map between a set of courses and a set of periods that satisfies all the constraints as much as possible.

HTS's hard constraints are similar to the UTS, except the constraints that are relevant to rooms and devices. There're 7 soft constraints, including the LS3, LS4, LS5, LS6, LS7, LS8 of UTS and one more soft constraint: The number of sessions that a course is scheduled into should be in the range of [*minSessions*, *maxSessions*].

4. THE MAIN FEATURES OF THE SUBSYSTEMS

The three subsystems of UTS and HTS are very similar. Therefore, in this section, we only describe the UTS's subsystems. The offline applications have most of the functions that online systems have and could connect to online systems for the convenience. Because of the similarity between them, we just describe the online systems, the details and executable files of offline application could be downloaded from the website of the author *www.fit.hcmuns.edu.vn/~nmkhang*

4.1 The teaching subsystem

When managing user adds a lecturer to the system, that lecturer will be sent a default username and password via his email address. That lecturer could use this account to log in to teaching subsystem via the log in site as shown in figure 3, after logging in, the lecturer could be able to:

1. View and update his individual information (email address, phone number…).
2. Register their available periods (they must give these information to the system for the scheduling task)
3. View their scheduled teaching timetable: lecturers can see the list of all instances that he attends and view his teaching timetables if it has already been scheduled.

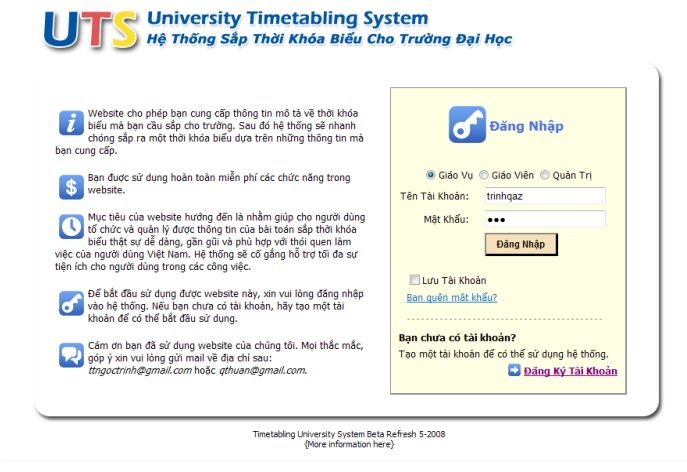

Figure 3: The log in page of UTS

4.2 The managing subsystem

This is the main subsystems. It has all the most important functions to support the automated timetabling task. In managing subsystem, managing users can:

1. Create an instance or upload an instance from XML file, export an instance to XML file and download it.
2. Export results to Excel file (.xls).
3. Schedule instance by calling the Tabu Search solver module (user can also use the offline application to run the solver).

4. Manage and modify information of each instance.
5. Email the result timetable to lecturers.

To create an instance using web-based systems, users must give information of time, people, things, groups and requirement parameters. The main important information is courses' info. This must be create at the last stage - after creating information of time, people, subjects, rooms, devices and groups. Requirement parameters are optional, if users don't set these value, the system will use default ones.

All the instances will be saved in database of the system. But users can exports them to XML files for other purposes. Users can view the result timetables in various points of view: lecturer view, class view, room view, device view, subject-class view and the general view. Results obtained after an instance is scheduled are appended to the original XML file of that instance. Sometimes the result timetables induced by the solver module cannot satisfy the requirements of users. In that case, these results could be exported to Excel files (.xls); this helps users to view results in a popular way and modify them by hands.

After all the information of an instance is supplied, managers can call the solver module to do timetabling automatically. But before the main algorithm (Tabu Search) is called, there will be a checking stage. In this stage, all the information is checked using four criteria to make sure that the input data is valid:

1. The total number of available periods of a lecturer must be not less than the total number of periods that this lecturer has to teach.
2. The total number of available periods of a class must be not less than the total number of periods that this class has to study.
3. Lecturer A of a given course B must have at least one block of B's minConsecutivePeriods consecutive available periods.
4. Three information of a course, including the number of lectures belonging to this course, minConsecutivePeriods and maxConsecutivePeriods should be reasonable (that means there must be at least one way to split this course into valid blocks).

Certainly these criteria are not enough to make sure that the input data is exactly valid, but they can help to eliminate few mistakes in the input before the algorithm runs. If the input data violates at least one of these criteria, the main algorithm will not be called and the system will send a report to users to help them correct the mistakes before recall the solver.

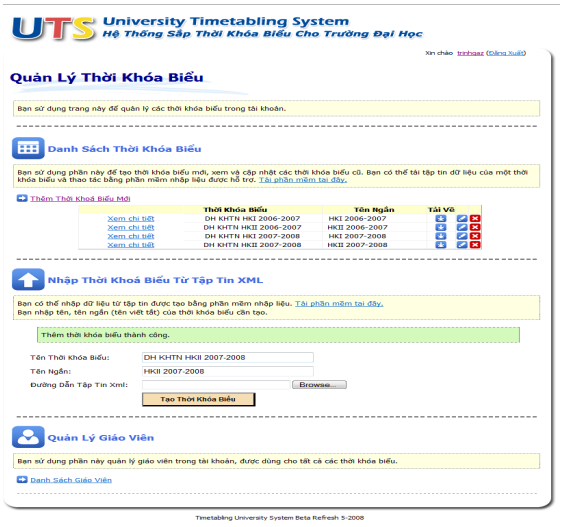

Figure 4: The timetable viewing page in the managing subsystem of UTS

4.3 The administrative subsystem

In administration subsystem, administrators can:

1. Activate or block lecturer and manager accounts.
2. Delete accounts.
3. Send emails to other accounts to remind their passwords.
4. View profiles of all lecturer and manager accounts.

5. THE SOLVERS - TABU SEARCH BASED ALGORITHMS

The solvers in UTS and HTS are adapted versions of the popular Tabu Search metaheuristic. Details of these algorithms are described and published in Khang et. al. (2010a) and (2010b). Both of them have two major phases: constructive phase using greedy algorithm and improvement phase using Tabu Search. There three kinds of moves considered: single move, which changes the periods and room (if needed) of a block of a course, swap move, which exchanges periods of two blocks of two courses, and block-changing move, which changes the block structure of a course. The experimental results shows that these algorithms are fairly efficient and could give good timetables in reasonable time. Details of the results will be presented in the next section.

6. EXPERIMENTAL RESULTS

In this section, a very brief experiment results when testing UTS and HTS solver modules using twelve real-world data instances in Vietnam are presented. University input data are collected from the University of Science and high school ones are taken from the High school for the Gifted and Tran Phu high school in Vietnam. The weights of soft constraints are set to be default values as follow:

- UTS: w(HS1) = 60, w(HS2) = 300, w(HS3) = 100, w(S1) = 15, w(S2) = 5, w(S3) = 5, w(S4) = 5, w(S5) = 2, w(S6) = 5, w(S7) = 10, w(S8) = 10, w(S9) = 5, w(S10) = 10
- HTS: w(S1) = 100, w(S2) = 5, w(S3) = 10, w(S4) = 5, w(S5) = 5, w(S6) = 5, w(S7) = 5

Each instance is tested ten times in computer with configuration of Core2 Duo processor, 1GB RAM, and Windows XP OS. The solvers are implemented in C++ by the authors. Time for running UTS algorithm is small, just less than five minute. However, running time of HTS algorithm is a bit larger (about 15 minutes) due to the big size of input instance, so if users don't want to run this solver throughout the Internet, they could use the off-line application of HTS to run the solver module, and then upload the result to the web-based system.

The objective function's values of each solver's average results, of the handmade timetables (created by the experts and were used in the reality) and the improvement rate of the automated ones versus the handmade ones are shown in Table 1 and Table 2 respectively. We could see that the solvers' results are better than handmade ones, and time for running of the solver is just evaluated by second or minutes, while time for solving these timetabling problem by handmade is often days or weeks.

Table 1: Results of the UTS's solvers

	Average	Handmade	Improvement rate (%)	Running time (s)
Instance 1	115	295	61%	43.8
Instance 2	135	400	66%	56.2
Instance 3	100	255	61%	55.3
Instance 4	110	370	70%	58.5
Instance 5	380	615	38%	206.2
Instance 6	315	510	38%	267.4

Instance 7	835	1155	28%	270.9
Instance 8	745	930	20%	306.7
Instance 9	570	895	36%	249.05

Table 2: Results of the HTS's solvers

	Average	**Handmade**	**Improvement rate (%)**	**Running time (s)**
Instance 1	3015	3130	4%	821
Instance 2	1954	2355	17%	724
Instance 3	1688	2005	16%	764

6. CONCLUSIONS

In this paper, we introduced two systems for solving concrete curriculum based course and high school timetabling problems in Vietnam. Problem models are able to be extended to problems of other universities and high schools due to their popular concepts. The advantage of these two systems is that it can help lecturers and staff managers work together throughout the Internet. Moreover, there are also two off-line applications which have the same main functions as the online systems. Input data and the result timetable of an instance are contained in just one XML file for flexibility. Therefore, users could switch between online and off-line systems without any confusion. Future work of this paper will focus on two problems: improving the convenience in putting input data and modifying result timetables by hand, and improving algorithms of the solvers. The authors hope that UTS and HTS could be helpful to educational institutions for solving timetabling problems - which are often hard to solve by hand and for managing information, distributing timetables to lecturers by taking the advantages of the Internet. Any comment on these two systems sent to the authors is highly appreciated. Addresses of UTS and HTS are www.fit.hcmuns.edu.vn/~nmkhang/uts and www.fit.hcmuns.edu.vn/nmkhang/~hts

ACKNOWLEDGEMENTS

The authors would like to express their thanks to Tran Duc Khoa, Tran Thi Ngoc Trinh, Dinh Quang Thuan, and Vo Xuan Vinh, Dang Thi Thanh Nguyen, Trieu Trang Khon for their major contribution to these systems.

REFERENCES

Costa., D. (1994) A Tabu Search algorithm for computing an operational timetable, *the European Journal of Operation Research*, vol. 76, pp. 98-110

Glover, F. (1989) Tabu Search - part I, *ORSA Journal on Computing*m vol. 1, pp. 190--206

McCollum , B., McMullan, P., Paechter, B., Lewis, R., Schaerf, A., Gaspero, L.D., Parkes, A. Qu, R., Burke, E. (2007) Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition, Dipartimento di Informatica e Sistemistica, Technical Report, the Internaltional Timetabling Competition 2007-2008

Jeffrey, H.K. (2006) The KTS High school Timetabling System, *in the Proceedings of the Practice and Theory of Automated Timetabling '06 conference*, Czech Republic, pp. 181—195

Schaerf., A. (1999a) Tabu search techniques for large high-school timetabling problems, *in the Proceeding of Association for the Advancement of*

Artificial Intelligence '96, pp. 210-218

Ramon A. V., Crespo E., Tamarit J. M. (2002) Design and implementation of a course scheduling system using Tabu Search, the European Journal of Operation Research, vol.137, pp. 517--525

Schaerf, A. (1999b) A Survey of Automated Timetabling, Dipartimento di Informatica e Sistemistica

Khang, N., Nguyen, D., Khon, T., Nuong, T. (2010a) Using Tabu Search for Solving a High School Timetabling Problem, Advances in Intelligent Information and Database Systems, vol. 283, pp. 305-313

Khang, N. (2010b) Automating a Real-world University Timetabling Problem with Tabu Search algorithm, in the Proceedings of the 2010 International Conference on Computing and Communication Technologies (RIVF'10), pp. 1-6