University Timetabling Algorithm Considering Lecturer's Workload

Lintang Yuniar Banowosari[1], Vega Valentine[2]
[1]Department of Computer Science and Technology
[2]Department of Industrial Technology
University of Gunadarma
Jakarta, Indonesia
[1]lintang@staff.gunadarma.ac.id, [2]valentvga@gmail.com

Abstract—**University Timetabling Problem is an allocation or subject to constraints, of given resources being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable university schedule requirements. In this paper, university timetabling algorithm is implemented, considering lecturer's workload in order to have a balance between lecturer's workload as a teaching staff of the university and to actualize the obligation of *Tridharma Perguruan Tinggi,* regulation issued by Indonesia's Ministry of Education. The implementation of faculty timetabling, the workloads summation and the lecture-class timetabling has successfully built in Java Netbeans Swing GUI.**

Keywords - University Timetabling Problem; lecturer's workload; university schedule requirements; university timetabling algorithm; **Tridharma Perguruan Tinggi;** ***Java Netbeans Swing GUI.***

I. Introduction

Scheduling is a process or a way of organize time according to arrangement of work order plan. It also means a list or activity table or activity plan with a detailed execution time [1]. In university terminology, this scheduling problem is known as University Timetabling Problem.

Every university has their own studying activities organized in such a way to satisfy any requirements they need. In schedule arrangement, universities tend to have a system which can schedule all courses optimally. To have such optimal condition of the course, a well-organized of all scheduling components is needed.

A scheduling system is also the core of university activities because it involves many elements in affiliations to the university, that is human resources (lecturers and students), time slot availability (length of lecture), type of the activity (theory or lab practice), and the facility to support those activities (classroom or laboratory) [6].

Bardadym (2006) classified the university timetabling into five groups [4], they are:

- *Faculty timetabling,* assigns qualified teachers to courses
- *Class-Teacher timetabling*, assigns courses with the smallest timetabling unit being a class of students
- *Course Scheduling*, assigns courses with the smallest scheduling unit being an individual student
- *Examination Scheduling*, assigns examination to students such that students do not have two examinations at the moment
- *Classroom Assignment*, assigns class-teacher couples to classrooms

In fact, there are many algorithms used to organize schedule in university timetabling. Algorithms such as Genetic Algorithm, Simulated Annealing and Tabu Search, are commonly used in university timetabling research.

But, which one is the best algorithm to do university timetabling? This question cannot be generally answered, because the problem is highly institution-specific. Every university has its own way in manage scheduling, with different requirements and regulations. In other words, managing timetable will be dependent on what regulation they hold and what requirements they need.

That is why no specific answer for the question. The best solution will be an algorithm that violate the least constraint or satisfy the most requirements or preferences for a certain university regulation.

One of the constraints in doing timetabling is nonetheless the activity of the lecturer itself because teaching is not always their only activity. Some regulations, such as the one issued from the government, obligate lecturers to do other things in order to dedicate and contribute more in education. In this paper, *Tridharma Perguruan Tinggi*, issued by Education Ministry of Indonesia, is taken as reference in defining activities of lecturer which will lead to some calculations to obtain optimal university timetable.

The outline of this paper is: Section II explains theory of timetabling, Section III describes methodology of workload calculation, and Section IV shows design and implementation of the algorithm.

II. THEORETICAL FRAMEWORK

A. *University Timetabling Problem (UTP)*

University Timetabling Problem is an allocation or subject to constraints, of given resources that is human resources (lecturers and students), time slot availability (i.e. length of lecture), type of the activity (theory or lab practice), and the facility to support those activities (classroom or laboratory) being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable university schedule requirements.

Edmund Burke in his article titled '*Applications to Timetabling*' [6] specified the timetabling problem as a problem with four parameters, *T* (a finite set of times), *R* (a finite set of resources), *M* (a finite set of meeting) and *C* (a finite set of constraints):

1. *Times*
   A time *t* is an element of the set of times *T* of an instance of the timetabling problem. A time slot is a variable constrained to contain one time.
2. *Resources*
   A resource *r* is an element of the set of resources *R* of an instance of the timetabling problem. A resource slot is a variable constrained to contain one resource. What we called resources are teachers, rooms, items of special equipment, students or group of students that supports a meeting.
3. *Meeting*
   A meeting *m* is a named collection of time slots and resource slot. Assigning values to those slots means that all of the assigned resources attend this meeting at all of the assigned times.
4. *Constraints*
   Constraints divided into two, hard constraint and soft constraint. Hard constraint must be satisfied while soft constraint is desirable, but not necessary, to satisfy—more to optimization objective.
   In university course timetabling, no-clashes constraint would typically be a hard constraint for lecturers but a soft constraint for student as far as optional courses are concerned since it usually impossible to satisfy every student.

B. *Algorithms to Solve University Timetabling Problem*

Algorithms had been developed and implemented in building a timetable for universities. Literatures about university course timetabling teach us that researchers applied different approaches to tackle the problem [4]. Above many algorithms, there are three most applicable and most widely used meta-heuristic algorithm to make an optimal university timetabling:

1. *Simulated Annealing*

Simulated annealing is a probabilistic method proposed in Kirkpatrick, Gellat, and Vecchi (1983) and Cerny (1985) for finding the global minimum of a cost function that may possess several local minima. It works by emulating the physical process whereby a solid is slowly cooled so that when eventually its structure is 'frozen', this happens at a minimum energy configuration [8].

Simulated annealing started with making the mathematical formulation of the problem that is the hard and soft constraints. After that, define properties of the constraints such as teaching duration, available class, etc. Then a lecture initially placed onto available timeslot.

Energy function, cooling and acceptance probability function also applied. The energy function is derived from the main timetabling objective (considering times, meeting, resource, and constraints), while cooling schedule and acceptance probability function controls accepting new solution with certain energy value. These two functions used to reach the objective of building optimized university timetable.

2. *Genetic Algorithm*

Genetic Algorithm was founded by John Holland in Michigan University, United State (1975) through some researches and David Goldberg introduced [9].

Three main aspects in genetic algorithm are definitions of fitness function, implementation of genetic representation and genetic operation. If the three aspects are defined, then the algorithm will be well-performed.

The algorithm started with a set of randomly selected state called population. Each state defined as a string. It combines two main parent populations. Through some crossover, mutation and fitness function, new children population will be defined as the solution.

3. *Tabu Search*

The basic concept of Tabu Search as described by Glover (1986) is 'a meta-heuristic superimposed on another heuristic' or a higher-level meta-heuristic procedure for solving discrete and continuous optimization problems.

The overall approach is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited. The solution space that has been visited therefore listed as 'tabu' [10].

Three main strategies of tabu search are [11]:

- Forbidding strategy, control what enters the tabu list
- Freeing strategy, control what exits the tabu list and when
- Short-term strategy, manage interplay between the forbidding strategy and freeing strategy to select trial solutions

Between those three algorithms, Tunçhan CURA, Istanbul University research group, compare the performance by modify those three algorithms [5] into a similar structure design to be proper with the IUFBA requirements. The proposed algorithm has been tested with the 2006-2007 academic year, first term course timetabling data of IUFBA.

From the comparison, he found that simulated annealing has been the best algorithm to solve university timetabling problem. He concludes this based on his experiments in having the three algorithms to do the same case.

Thus for the case of lecturer's workload, simulated annealing algorithm will be used and the equation performed by [5] will be modified to satisfy requirements as explained in the next section.

C. *Lecturer's Workload*

Lecturers stated as a professional educator and a scientist whose prime objective is to transform, develop, and publish knowledge, technology, and art through education, research, and dedication to public [7].

In Indonesia, lecturer's performance of education always obeying the rule of *Tridharma Perguruan Tinggi* which consists of three dharma. They are Dharma of Education and Teaching, Dharma of Research and Dharma of Public Dedication. Detailed description explained on the calculation part (section three).

III. METHODOLOGY

A. *Formulation of the Problem*

The following subsection explains the process of listing obligatory rules and constraints and the mathematical formulation of rules and constraints defined.

1. *Defining Obligatory Rules, Hard and Soft Constraints*

The obligatory rules that generally overdue in universities are:

- No. 1: Each lecture must be assigned to only one class of student at one day and to a single time slot
- No. 2: The lengths of the lectures hours must be taken into consideration while assigning the lectures. For example if the lecture hours are from 9 am to 5 pm and the length of the lecture is 2 hours, this lecture cannot be assigned to 4 pm since it would have exceeded the official lecture hours
- No. 3: More than one lecture cannot be assigned to a given class at the same time interval
- No. 4: A lecturer cannot have more than one lecture assigned in a given time interval

The hard and soft constraints defined as seen in table I.

TABLE I. HARD AND SOFT CONSTRAINTS

Hard Constraints	**Soft Constraints**
No resources (lecturer and a class of students) may be assigned to different events at the same time	Every lecturer has his/her own availability schedule or submits a plan with desirable time periods that suits him/her best
There is a maximum number of time periods per day, that may not be exceeded	Every lecturer has a minimum and a maximum limit of weekly work-hours
More than one lecture can not be assigned to a given class at the same time slot	Minimize the time gaps within the schedule of each lecturer
Each lecture may be assigned to a lecturer that belongs to a specific set of lecturers that can deliver the lecture	Minimize the time gaps within the schedule of each given class

2. *Mathematical Formulation of the Problem*

Meeting of lectures, lecturers and rooms that available, denoted by *J, I* and *L* respectively. Lectures can be assigned to any lecture day from Monday to Saturday. Each day consists of 10 hours. Thus, $D = 6$, $H = 10$, denote the number of days and hours of timetable. Thus, the rules will be denoted as follow:

- The general mathematical model for satisfying the lecturer desires represented as:

$$\max \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{d=1}^{D} \sum_{h=1}^{H} \sum_{l=1}^{L} \sum_{h^*=h}^{\min\{(Y_j+h),H\}} X_{ji} \times P_{idh} \times C_i \times S_{jldh} \quad (1)$$

- Obligatory rule 1 is imposed by:

$$\sum_{l=1}^{L} \sum_{d=1}^{D} \sum_{h=1}^{H} S_{jldh} = 1, j = 1,..., J \quad (2)$$

- Obligatory rule 2 is imposed by:

$$\beta_{jldh} \le H$$
$$j = 1,...J; h = 1,..., H; l = 1,..., L; d = 1,..., D \quad (3)$$

- Obligatory rule 3 is imposed by:

$$\beta_{jldh} \times S_{jldh} \le \beta_{j^* ldh^*} \quad (4)$$

- Obligatory rule 4 is imposed by:

$$\beta_{jldh} \times S_{j^* l^* dh^*} \times X_{j^* i} \times X_{ji} \le \beta_{j^* ldh^*} \quad (5)$$

Y_j denotes the length of lecture j (j = 1,..., J). X_{ji} is a class of students with defined lecture j and lecturer i (i = 1,..., I). P_{idh} denotes the desire time slot (a higher value indicating a higher preference) of lecturer i for day d (d = 1,..., D) and hour h (h = 1,..., H). C_i denotes the lecturer's workload of lecturer i (i = 1,..., I). S_{jldh} is space for lecture in the timetable. β_{jldh} is a lecture with defined length in hour (duration).

B. *Lecturer's Workload Calculation*

Based on *Lampiran II Surat Dirjen Dikti No. 3298/D/T/99* issued on 29 Desember 1999 [7] about lecturer's workload evaluation, the details of workload calculation is described on table II.

TABLE II. LECTURER'S WORKLOAD IN DETAIL

No	Activity	Hour/week	Multiplier Notation
A	***Education***		
1.	Give a lecture 'X' (*y* credits)	*y*	$\sum class$
2.	Assess final examination	0.5	$\sum exam$
3.	Assess thesis defense for 3 students	0.5	$(\sum student) \div 3$
4.	Thesis consultation to a student	2	$\sum student$
5.	Student academic adviser for 20 students	1	$(\sum student) \div 20$
B.	***Research***		
1.	Make one research topic per year (as main researcher)	10	$\sum research$
2.	Writing papers to accredited journal, a title per 2 year (as main author)	1	$\sum paper$
C.	***Public Dedication***		
	Giving a workshop for 1 topic per semester	1	$\sum workshop$
D.	***Supporting Activities***		
1.	Active in a committee during a year	1	$\sum committee$
2.	Attend campus event (seminars, meetings, etc)	0.5	$\sum event$

Table III describes the maximum workload can be hold by a lecturer according to ministry's regulation.

TABLE III. WORKLOAD CALCULATION

No	Activity (appropriate to ideal lecturer's workload)	hour/week
A	***Education***	
1.	Teaching a lecture 'X' (3 credits)	9
2.	Teaching a lecture 'Y' (3 credits)	9
3.	Giving consultation to students for (undergraduate) thesis, 3 student per semester	6
4.	Student advisor for 20 students per semester	1
5.	Assessing final examination or (undergraduate) thesis defense , 3 students per semester	0.5
6.	Making one course dictate per year	2
	Total of A	**27.5**
B.	***Research***	
1.	One research topic per year, as the main researcher	10
2.	Writing papers to accredited journal, a title per 2 year as main author	1
	Total of B	**11**
C.	***Public Dedication***	
	Giving a workshop for 1 topic per semester	1
D.	***Supporting Activities***	
	Active in a committee during a year	1
	Sum of Total	**40.5**

C. *Timetabling Solver*

1. *Defining the Number of b Vectors*

Let X be the number of different lecture lengths. Thus, each different length, the number of b_k where $hour_k$ equals this length are denoted by λ_x, δ_x, and μ_x ($x = 1,\ldots, X$) respectively.

For example, if there are 3 lectures and their lengths are 2 hours, 2 hours and 3 hours respectively, then the number of different lengths (X) will be 2 ($\lambda_1 =$ 2 hours and $\lambda_2 =$ 3 hours), and δ_1 will be 2 and δ_2 will be 1. The number, K, of b vectors imposed by equation (6).

$$\sum_{x=1}^{X} \mu_x \tag{6}$$

For this study, the sample data was taken from Gunadarma University's Faculty of Psychology for 4th grade class in ATA 2008/2009. For this sample, we got $X = 3$ with $\lambda_1 = 1$, $\lambda_2 = 2$ and $\lambda_3 = 3$. Thus by equation above, we got $K = 7$ with $\delta_1 = 1$, $\delta_2 = 5$ and $\delta_3 = 1$.

2. *Filling the b Vectors with Lectures*

In this study, the process of assigning defined lecture to *b* vectors using indirect representation. In such representation, the encoded solution usually represents an ordered list of events, which are placed into the timetable according to some predefined method, or so called timetable builder. The timetable builder can use any combination of heuristics and local search to place events into the timetable, while observing the problem's constraints.

For this work, the indirect representation encodes 3 fields for each event:

- Day and hour (time slot) to allocate the event
- Teacher (1 or more) to be assigned to the event
- Class of students that supposed to take the event

All fields are first encoded as integers and then converted into appropriate variable type for further process in the program. In generating the solution, the solver first decodes it to gain these four fields for every event in the schedule. Then it invokes the timetable builder to works as in Figure 1.

IV. Design & Implementation

A. *Lecturer's Workload Implementation in Timetabling Algorithm*

In this study, lecturer's workload divided into two different workloads. First is teaching workload (`workload_teach`) and the second is administration workload (`workload_adm`). Teaching workload is the total workload of assigned course calculated by the amount of SKS per course for each class. Administration workload is the total workload of activity but teaching, which is defined in the *Tridharma*.

Pseudocode of Lecturer's Workload implementation to timetable is:

```
get lecturer's workload_adm
get lecturer's workload_teach
if workload_adm + workload_teaching < 40.5
    then put lecturer into S(d,h) matrix
    Do
       insert lecture into S(d,h)
       if any constraint violated
           then search subsequent S(d,h) until no
           violation
       else continue inserting to S(d,h)
    Until workload_teach = 0
else exceed lecturer's max workload
```

For describing how the algorithm works in such a real data, table IV is sample input case of lecturer's activity in a semester:

TABLE IV. Lecturer's Activity as Input Test Case

Administration Workload	Teaching Workload
Assess 3 final examination (A2)	Teach a lecture 'M' (2 credits) @ 3 classes (A1)
Give workshop for 3 topics this semester (C1)	Teach a lecture 'N' (1 credit) @ 6 classes (A1)
Write a paper to accredited journal (B2)	Teach a lecture 'P' (1 credit) @ 2 classes (A1)
Thesis consultation for 9 students (A3)	

From Table IV, we can calculate weights for administration and teaching workload as follow:

Administration Workload

1. Assess 3 final examination
   $0.5 \times \sum exam = 0.5 \times 3 = 1.5$
2. Give workshops for 3 topics
   $1 \times \sum workshop = 1 \times 3 = 3$
3. Write a paper to accredited journal
   $1 \times \sum paper = 1 \times 1 = 1$
4. Thesis consultation for 9 students
   $0.5 \times (\sum students) \div 3 = 0.5 \times (9 \div 3) = 1.5$

Total administration workload = 7 hours per week

Teaching Workload

1. Teach a lecture 'M' (2 credits) @ 3 classes
   $y \times \sum class = 2 \times 3 = 6$
2. Teach a lecture 'N' (1 credit) @ 6 classes
   $y \times \sum class = 1 \times 6 = 6$
3. Teach a lecture 'P' (1 credit) @ 2 classes
   $y \times \sum class = 1 \times 2 = 2$

Total teaching workload = 14 hours per week

From calculations above, we get the total result of 21 hours workload from administration workload plus teaching workload (7+14). The value is below the maximum workload of 40.5 hours per week. Thus, the lecturer can still be assigned to another teaching assignment through the timetable process (Figure 1) or other administration work. While for some other that reach the total workload of 40.5, they will have the opposite treatment such as workload reduction either from administrational or teaching assignment.

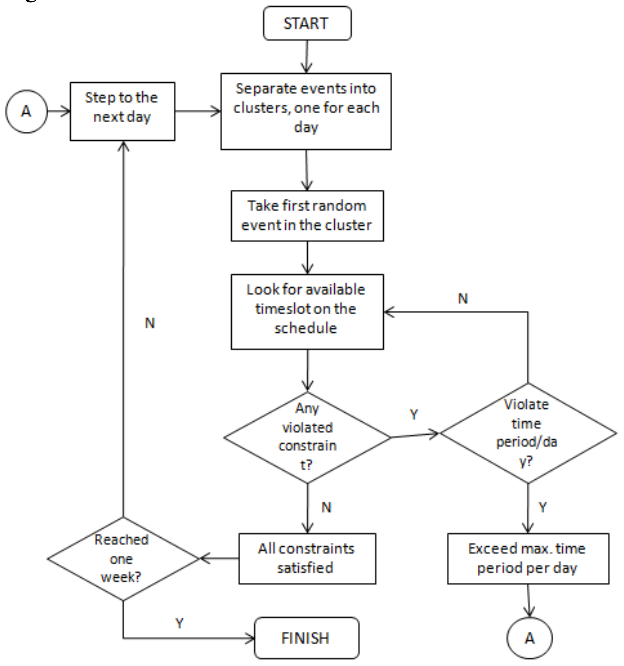

START
Separate events into clusters, one for each day
Take first random event in the cluster
Look for available timeslot on the schedule
Any violated constraint?
Violate time period/day?
Exceed max. time period per day
All constraints satisfied
Reached one week?
Step to the next day
FINISH
A
Y
N

Figure 1. Timetable Builder Algorithm

The constraints involved so far are the hard and soft constraints as listed in Table I. According to the pseudocode, lecturers with maximum workload cannot be assigned to another event anymore. This condition verifies the soft constraint. The time slot is set to be a unique *S(d,h)* matrix including the unique day and hours per week. Therefore, an event-clash for related resources (lecturer and class of student) can be automatically avoided. The treatment for any other constraint will be the same, i.e. search for the next available *S(d,h)* slot.

B. *Implementation on Java, Netbeans Swing GUI*

University Timetabling Application is a desktop application which is developed to facilitate computerization in solving the university timetabling problem. It is built in Java programming language using Netbeans Swing GUI for designing the graphical user interface and database, handled by PostgreSQL.

This application built based on previous object-oriented analysis through the system which also developed by the algorithm already explained. The analysis then visualized using Unified Modeling Language (UML), i.e. use case diagram, class and activity diagram (Figure 2)

From the analysis, this application would contain five modules. They are functioned to store subject's data, lecturer data, lecturer's activity, to assign lecturers to subjects and the timetable module.

The main parameter in this application is the credit of a subject, number of class which should get the related subject and the total credit taken by the lecturer. The total credit is calculated by number of credits and class (as explained in IV A) which provides total hours that should be taken by the lecturers. The total hours considered as the lecturer's workload and determine whether the lecturer can still be assigned to another event or not (Figure 1).

In Figure 3, the initial workload calculation of administrative work is calculated by module 'lecturer's activity' (Figure 3a), while the assignment to teach a subject organized by module 'lecture assignment' which shows the detailed parameter of subjects (subject's name, credit) and also the initial administrative workload (Figure 3b).

The first four modules are already set and work properly, while so far, the timetable module is still on progress.

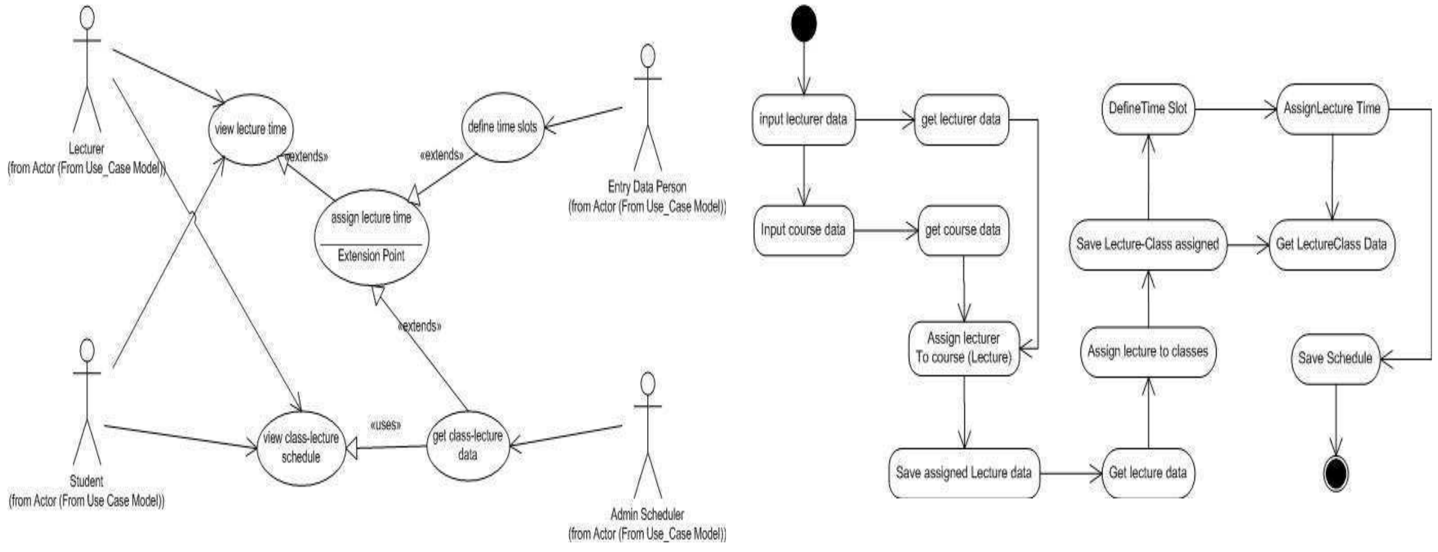
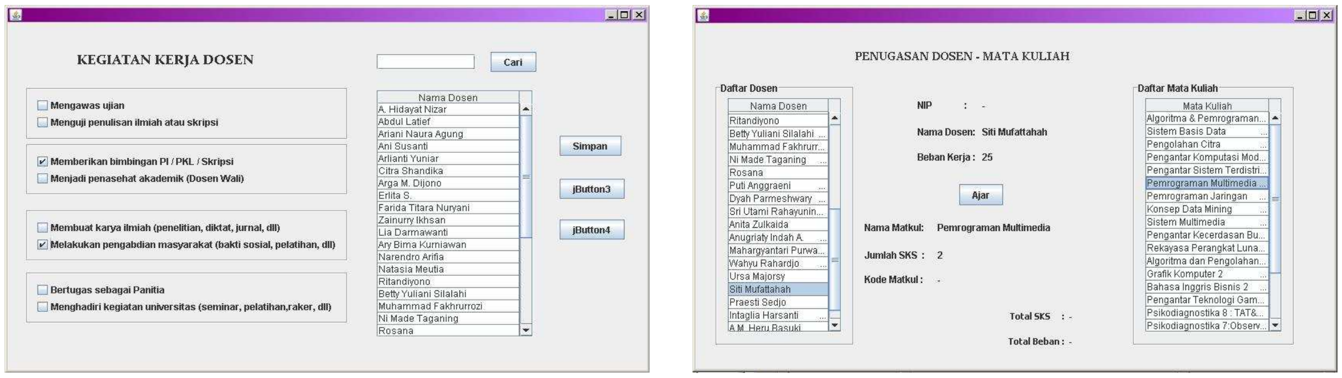


Figure 2. Unified Modeling Language for University Timetabling (a) Use Case for Assigning Lecture (b) Activity Diagram



KEGIATAN KERJA DOSEN

Cari

- [ ] Mengawas ujian
- [ ] Menguji penulisan ilmiah atau skripsi
- [✓] Memberikan bimbingan PI / PKL / Skripsi
- [ ] Menjadi penasehat akademik (Dosen Wali)
- [ ] Membuat karya ilmiah (penelitian, diktat, jurnal, dll)
- [✓] Melakukan pengabdian masyarakat (bakti sosial, pelatihan, dll)
- [ ] Bertugas sebagai Panitia
- [ ] Menghadiri kegiatan universitas (seminar, pelatihan,raker, dll)

Nama Dosen
A. Hidayat Nizar
Abdul Latief
Ariani Naura Agung
Ani Susanti
Arlianti Yuniar
Citra Shandika
Arga M. Dijono
Erlita S.
Farida Titara Nuryani
Zainurry Ikhsan
Lia Darmawanti
Ary Bima Kurniawan
Narendro Arifia
Natasia Meutia
Ritandiyono
Betty Yuliani Silalahi
Muhammad Fakhrurrozi
Ni Made Taganing
Rosana

Simpan

jButton3

jButton4

PENUGASAN DOSEN - MATA KULIAH

Daftar Dosen

Nama Dosen
Ritandiyono
Betty Yuliani Silalahi ...
Muhammad Fakhrurr...
Ni Made Taganing ...
Rosana
Puti Anggraeni ...
Dyah Parmeshwary ...
Sri Utami Rahayunin...
Anita Zulkaida
Anugriaty Indah A. ...
Mahargyantari Purwa...
Wahyu Rahardjo ...
Ursa Majorsy
Siti Mufattahah
Praesti Sedjo
Intaglia Harsanti ...
A.M. Heru Basuki

NIP : -

Nama Dosen: Siti Mufattahah

Beban Kerja : 25

Ajar

Nama Matkul: Pemrograman Multimedia

Jumlah SKS : 2

Kode Matkul : -

Total SKS : -

Total Beban : -

Daftar Mata Kuliah

Mata Kuliah
Algoritma & Pemrograman...
Sistem Basis Data ...
Pengolahan Citra ...
Pengantar Komputasi Mod...
Pengantar Sistem Terdistri...
Pemrograman Multimedia ...
Pemrograman Jaringan ...
Konsep Data Mining ...
Sistem Multimedia ...
Pengantar Kecerdasan Bu...
Rekayasa Perangkat Luna...
Algoritma dan Pengolahan...
Grafik Komputer 2 ...
Bahasa Inggris Bisnis 2 ...
Pengantar Teknologi Gam...
Psikodiagnostika 8 : TAT&...
Psikodiagnostika 7:Observ...

Figure 3. Screenshot of University Timetabling Application's GUI (a) Lecturer's Activity Module (b) Lecture Assignment Module

V. CONCLUSION

In solving university timetabling problem, three algorithms, Simulated Annealing, Genetic Algorithm, and Tabu Search had been theoretically studied. Simulated Annealing supports solving university timetabling problem with consideration of additional variable, such as lecturer's workload, therefore selected for this case.

Lecturer's activity had been categorized and being weighted. It applied to an input test case, simulating calculation of the lecturer's workloads, together results the output of calculation. This output will determine placement of schedule onto the timetable, obeying the obligatory rule, hard, and soft constraints.

However, the implementation of the algorithm using GUI Swing Netbeans has only reached the process of faculty timetabling, the workloads summation and the lecture-class timetabling. Further refinement needed to be done to get the optimal University Timetabling Application. Cooling function for this application is to be considered for the whole timetable because so far it only considers individual lecturer's workload.

REFERENCES

[1] D. Sugono, "Kamus Besar Bahasa Indonesia", Pusat Pembinaan dan Pengembangan Bahasa Indonesia, Departemen Pendidikan dan Kebudayaan, Balai Pustaka PN, 1993.

[2] V. Bardadym, "Computer-Aided School and University Timetabling: the New Wave", Selected and Revised Papers of the 1st International Conference on Practice and Theory of Automated Timetabling, (PATAT 1995), Edinburgh, Springer LNCS 1153, 22-45, 1996.

[3] B. Oestereich, "Developing Software with UML: Object-Oriented Analysis and Design in Practice", Second Edition, 2002, Edinburgh Gate, Harlow CM20 2JE, Pearson Education.

[4] A. Mieke, P. Causmaecker, P. Demeester, and G. Berghe, "Tackling the University Course Timetabling Problem With an Aggregation Approach", KaHo Sint-Lieven Information Technology & Katholieke Universiteit Leuven Campus Kortrijk, Belgium, 2005.

[5] T. Cura, "Timetabling of Faculty Lectures Using Simulated Annealing Algorithm", İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, Turkey, 2007.

[6] E. Burke, D. Werra, and J. Kingston, "Applications to Timetabling", University of Nottingham (UK), École Polytechnique Federale de Lausanne (Switzerland), University of Sidney (Australia), 2003.

[7] Lampiran II Surat Dirjen Dikti No. 3298/D/T/99 issued on 29 Desember 1999.

[8] D. Bertsimas and J. Tsitsiklis, "Simulted Annealing", Statistical Science Vol. 8 No. 1 pp 10-15, Sloan School of Management & Electrical Engineering and Computer Science Management, Massachusetts Institute of Technology, Cambridge, 1993.

[9] S. Kazarlis, V. Petridis and P. Fragkou, "Solving University Timetabling Problem Using Advanced Genetic Algorithms", Technological Educational Institute of Serres & Aristotle University of Thessaloniki, Greece, 2002.

[10] R. Battiti, P. Gray, W. Hart, "Tabu Search", 1997, Sandia National Laboratories, Albuquerque, NM 87185. (http://www.cs.sandia.gov/opt/survey/ts.html) [accessed 19 August 2010]

[11] H. Zhang, "Artificial Intelligent: Tabu Search", 22C:145 – Artificial Intelligence, 24 October 2008, The University of Iowa, Fall 2008. (http://www.cs.uiowa.edu/~hzhang/c145/notes/04ts-search-6p.pdf) [accessed 19 August 2010]