International Journal of Engineering, Technology and Natural Sciences
E-ISSN: 2685-3191 | P-ISSN: 2775-7706
Vol 3 No 1 (2021)

Designing Query Optimization for Scheduling the Lecture Across Faculties and Study Programs

Joko Aryanto [*, a,1], Ahmad Tri Hidayat [b,2]
[a,b] Universitas Teknologi Yogyakarta, Yogyakarta, Indonesia
[1] joko.aryanto@uty.ac.id, [2] ahmad.tri.h@uty.ac.id[*]

Abstract

University Course Timetabling Problem (UCTP) is a combinatorial optimization with a high complexity model. UCTP optimization aims at producing an optimal arrangement of lecture schedules by eliminating problems during the lectures and minimizing violations of the constraints. Previous studies apply UCTP problems with fixed or static scheduling time slots to handle the lecture time of each lecture session. However, not all universities use a static scheduling model to collect research data. This study aims to design an optimization model to solve UCTP with dynamic scheduling slots. Data were collected from all existing study programs at the University of Technology Yogyakarta. Data consisted of (1) the distribution of theoretical classrooms data, (2) the distribution of practicum classrooms data, (3) the distribution of credit load and course time slots data, (4) the distribution of lecturers' teaching schedules data, (5) the distribution of odd and even courses data, (6) the distribution of sessions on each lesson data. The software architecture, including database design, system workflow, and interface design, was conducted after collecting and analyzing data. The results of this study are expected to produce an appropriate lecture schedule to use utilizing Query Language optimization to implement the Spread Insertion Algorithm.

Keywords: Query Optimization, component, formatting, styling

I. INTRODUCTION

The preparation of the lecture schedule in the learning-teaching process is a tangible thing in an educational institution. To produce effective learning-teaching activities, the processing of the lecture schedule must be arranged as well as possible. Preparing the lecture schedule at the university level, which has several faculties and many study programs, is not easy. The university applies a system of using space and resources for all study programs and faculties. The preparation of the lecture schedule will be a big problem if it is done conventionally with a spreadsheet application.

Lecture schedules must avoid scheduling conflicts between lecturers, rooms, and lecture participants. It means that one lecturer may not teach more than one lecture simultaneously, one room may not be used by more than one lecture simultaneously, and lecture participants (students) may not attend more than one lecture at the same time. In addition, the preparation of this schedule must also pay attention to certain limitations set by the university; for example, each lecturer may not teach more than 6 credits in one day to obtain maximum quality lectures. Another factor that underlies this research is how to take advantage of the limited-lecture rooms available and used by many study programs.

University Course Timetabling Problem (UCTP) is one of the widespread problems in combinatorial optimization, which has a high level of complexity. UCTP aims to arrange lecture schedules at the university level by allocating each lecture consisting of lecturers, students, and courses into a certain space and time [1] lectures into a certain space and time [1]. UCTP defines *constraints* as *Hard Constraints* (HC) and *Soft Constraints* (SC). HC is a limitation that is absolute and cannot be violated. Violation of the HC results in an invalid resulting solution. While SC is a compromising constraint, meaning that the limit is not violated as much as possible, if there is a violation of the SC, then the resulting solution is still considered valid.

Experts have widely studied UCTP with various methods applied, which generally focus on swarm-based or evolutionary techniques. In [2], it is stated that UCTP is a combinatorial problem with a very large solution space, which of course, cannot be solved efficiently by deterministic algorithms. The search space is very large, while the deterministic search method tends to make observations one by one on the possible solutions in that space. In addition, the optimal solution of the problem is not only one, but there are many possible optimal solutions. Therefore, in [2], it is stated that an evolutionarily more realistic approach is applied to solve UCTP. *Swarm intelligence*-based metaheuristic approaches, such as *Particle Swarm Optimization* (PSO), can also be applied to solve case studies such as UCTP [3, 4]. However, PSO is an optimization method based on decimal values, while UCTP requires a discrete-based solution, so PSO requires a discretization mechanism for each particle to convert decimal values to discrete. In addition, PSO uses a fairly complex mathematical model to update the position of each particle in each iteration/generation.

Research in distributed database systems has popularized the architecture in which the DBMS (*Database Management System*) provides an interface for processing heterogeneous data, and the interface provides access to data sources [5]. In this context, data processing using a *Query* consists of sending a sub- *Query* to the data source and then integrating the results of that sub- *Query* to produce the final result. One of the main reasons for the success of relational database technology is the use of declarative languages and optimization of the *Query* process. The user side determines what data needs to be retrieved, and the database takes over the task of finding the most efficient method of retrieving that data. The process is a *Query* optimization process to evaluate alternative methods of running a *Query* and choose the best alternative.

Several techniques have been proposed to enhance traditional *Query* optimization. These techniques include better statistics [6], new algorithms for optimization [7-9], and adaptive architecture for execution [10]. A particularly promising technique in this direction is re-optimization, in which the optimization and execution stages of the *Query* processing are inserted, up to several times, during the run time of the *Query* [11-14]. Re-optimizers currently use a reactive approach to re-optimization; they use traditional optimizers to create plans, then do statistical tracking, respond to forecast errors, and generate sub-optimality detected in plans during execution. Reactive re-optimization is limited by an optimizer that does not include issues affecting re-optimization and suffers from several flaws. A proactive re-optimization approach is used during optimization to generate a robust plan; it is switchable and used as a random sample processing for each *Query* execution. As a result, it suffers some negative points [15].

This research presents *Query* optimization based on a time scheduling approach to reduce *Query* re-optimization. To this end, we added a timer object with an adaptive *Query* processing system to address issues with effect re-optimization. A timer object was created to execute a predefined set of simple *Queries in* a predefined schedule to collect statistical estimates about the database quickly, accurately, and efficiently at runtime. When the timer object fires (multiple times), it executes a specified set of simple *Query* and the specified amount of time between executions, the timer object performs the actual *query* execution. This pattern of *Query* work prompted us to conduct simulation experiments by demonstrating a new approach to produce significant improvements in actual *Query* performance.

II. METHOD

This research uses the query method to optimize the scheduling process. Optimization refers to the organization's needs for which a system requirements analysis and design has been carried out.

A. System Requirements Analysis

In general, the system developed aims to assist the administration in compiling the lecture schedule, which has been prepared manually using a spreadsheet that takes a long time, along with the limited supporting resources available. The application must also accommodate special requests from lecturers, especially external lecturers, who want class schedules at a certain time. In addition, the application must also be able to generate a schedule that can be changed manually by the user/university administration but still does not cause problems that interfere with the lecture process.

The amount of lecture data that must be compiled requires application developers to use models and resources as optimally as possible. Therefore, the Spread Insert Algorithm, which is used as an optimization method, must be adjusted to the conditions during the research to produce better performance. In addition, the computing process must be designed to run in parallel through a multithreading mechanism so that the use of computer resources is more optimal to facilitate the process of preparing lecture schedules.

Based on data collection, the following requirements are needed in this design, including Data 1). Name and number of Faculties, 2). Name and Number of Study Programs., 3). Name and number of theoretical rooms., 4) Name and the number of practicum rooms., 5). Course data and the number of classes for each course.

1) Faculty name and number

The number of faculties used was taken from the Yogyakarta Technological University; there are 4 types.

K_FAK	N_FAK
1	Bisnis & Humaniora
2	Sains & Teknologi
3	Program Diploma
4	Program Pascasarjana

Figure 1. Faculty

2) Name and Number of Study Programs.

As for the study programs from these faculties, there are 8. So to schedule it, the concept of generating is used automatically.

K_PRODI	PRODI	FAK
1	Akuntansi	1
2	Teknik Elektro	2
3	Akuntansi	3
4	Manajemen	1
5	Teknik Komputer	2
6	Bahasa Jepang	3
7	Sastra Inggris	1
8	Sistem Informasi	2
9	Bahasa Inggris	3
10	Psikologi	1

Figure 2. Study Program

3) Name and Number of Rooms

The scheduling process uses space with a capacity of 40 for each room. The number of rooms is 43 classrooms for theory lectures and 11 laboratory rooms.

idruang	kdruang	ruang	kpst
1	R1	RUANG1	40
2	R2	RUANG2	40
3	R3	RUANG3	40
4	R4	RUANG4	40
5	R5	RUANG5	40
6	R6	RUANG6	40
7	R7	RUANG7	40
8	R8	RUANG8	40
9	R9	RUANG9	40
10	R10	RUANG10	40

Figure 3. Lecture room

B. System Planning

The system design is carried out using the concept of optimization to get the best schedule. The schedule will be automatically plotted based on the built Query.

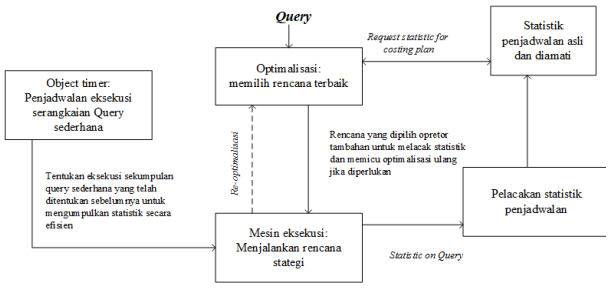

Figure 4. System Planning

The system will generally receive static data input as master data and dynamic data input as transaction variables (as in Figure 1). In most cases, distributed database environments run in unpredictable and unstable environments. So it is not easy to produce efficient database *query* optimization based on the information available at compile time. The solution to this problem is to exploit the information available at the *Query* runtime and adapt the *Query* plan to change environmental conditions during execution. This section presents an adaptive *Query* plan re-optimization technique for dynamically changing the *Query* plan at data processing time.

Query optimization based on the scheduling time approach aims to reduce *Query* re-optimization. This approach runs a set of simple predefined *Queries within* a predefined scheduling time to collect approximate statistics about the database quickly, accurately and efficiently at data processing time.

III. RESULT AND DISCUSSION

This research produces a scheduling model to plot the courses into the available rooms so that this research can automatically make a schedule.

A. Making Room

Creating a room is carried out to determine how many classes are available from the room prepared as a whole. Physically, the available rooms will be divided into hours of user sessions to form the rooms' names. The regular time for one theory room is used for 12 credits or 6 sessions, divided before recess and after resting hours every day. The process carried out is giving a code to each room consisting of days, types of courses, rooms, and sessions. An example of coding for each room can be seen in the modelling below from code 11196.

Figure 5. Space coding sample

1. Days are represented by the numbers 1 = Monday, 2 = Tuesday, etc.
2. Type symbolizes room 1= Theory, 2= Practicum.
3. Space is the number of available physical space codes.
4. The session describes how many sessions the room is in a day.

The code for the new room is written in the Pseudocode below.

```
@x=max(roomCode);
@y=1;
While @y<=@x
Begin
  @day=1
  While @day<=6
  begin
  @sesi=1
  While @sesi<=6
  Begin
     @newcode=@day+@jen=(select    jns
from room where roomcode=@x)+@roomcode+
                                  @sesi;
     @sesi=@sesi+1;
  End;
  @day=@day+1;
  End
 @y=@y+1
end
```

The results of the pseudocode execution will be stored in a table which will then be changed to be filled with courses that already have teaching and learning time, as shown in the following table.

ID	ROOM_CODE	MK	SKS	START	FINISH
1	11011	-	0	00:00.0	00:00.0
2	11012	-	0	00:00.0	00:00.0
3	11013	-	0	00:00.0	00:00.0
4	11014	-	0	00:00.0	00:00.0
5	11015	-	0	00:00.0	00:00.0
6	11016	-	0	00:00.0	00:00.0
7	11021	-	0	00:00.0	00:00.0

The number of rooms obtained is based on the following formula calculation:

$$R_t = (j_t * s_t) * h \quad \ldots 1)$$

r_t = Theoretical space
j_t = Number of theoretical rooms
s_t = Number of theoretical Sessions
h = number of days

If there are 50 theoretical rooms provided, the number of theoretical rooms will be: (50*6)*6 → 1800 rooms can be used every week.

B. Determine Each Course Session Hours

Determining PBM hours is needed to get a room and avoid overlapping rooms. The first process that is carried out is to create a new code in each course which aims to make it easier to determine the start and finish hours based on the number of credits for each course. This new coding process will help speed up the process of searching for course data based on:

1. This type of subject is theory or practical.
2. It is a subject of the study program and its faculties.

This process is carried out and makes it easier to determine study hours; it also determines the number of courses to be held because one course consists of several class groups. The new code from the course will be stored in the following table.

URT	ID	ARTICLE	SKS	JKLS	JNS	NEW_CODE	SMT
1	201	BAHASA INGGRIS II	2	8	1	101245	2
2	202	SISTEM BASIS DATA	3	8	1	102245	2
3	203	SISTEM BASIS DATA PRAKT	2	12	2	203245	2
4	204	TEKNOLOGI BERNASIS OBJEK	3	8	1	104245	2
5	205	APLIKASI TEKNOLOGI INF.II	2	11	2	105245	2

There is a column "new_code", the coding used as a liaison between the courses and the available rooms. Here is an explanation of the new code that was formed

The following is the Pseudocode for creating a new course code.

```
@x=max(ID);
@urt=1;
While @urt<=@x
Begin
   update            MK            set
NEW_CODE=CONVERT(varchar(2),jns)+
CONVERT(varchar(2),(right(kode,2)))+
    CONVERT(varchar(2),
(left(kode,1)))+ @prodi where ID=@urt
    @urt=@urt+1;
End
```

C. Connecting Courses to the Room

The stage of merging courses and rooms is the last stage to get a schedule that does not overlap, both the room and the lecturer in charge of each course.:

```
   @sesi=1;
   while @sesi<=6
   begin
     insert into @kls select ROW_NUMBER()
over(order by mk) as  nom,mk, msk,sls,
   ke,0 from jadual where ke=@sesi and
ruang='0'
    set @urt=1;
    while @urt<=(select COUNT(nom) from
@kls)
    begin
     set @mk=(select mk from @kls where
nom=@urt);
     set @kelas=right(@mk,4);
     set    @adakelas=(select    top    1
RIGHT(mk,4) from jadual where ke=@sesi
and
   RIGHT(mk,4)=@kelas);
     set @mulai=(select msk from @kls
where nom=@urt)
     set @sls=(select sls from @kls where
nom=@urt)
     set @ada=(select distinct case when
RIGHT(mk,4) = null then '0' else
   RIGHT(mk,4) end as v  from jadual
                where     ke=@sesi     and
RIGHT(mk,4)=@kelas);
     set @ruang=(select top 1 ruang from
detruang where left(ruang,4) in
                (select     left(ruang,4)
from jadual where left(ruang,4) in
          (select    left(ruang,4)    from
jadual where ruang<>0 group by
   left(ruang,4),sls
having((MAX(RIGHT(ruang,1))=@sesi-1) and
   (sls<@mulai)))   and   sls<@mulai   and
SUBSTRING(ruang,2,1)=LEFT(@mk,1)) and
   LEFT(ruang,1)>(select
max(LEFT(ruang,1))from jadual
                where
RIGHT(mk,4)=@kelas and ke=@sesi)
                and ruang not in (select
ruang from jadual) order by ruang asc)
     if @ruang is null
     set @ruang=(select top 1 ruang from
detRUANG where RIGHT(ruang,1)=@sesi and
   SUBSTRING(ruang,2,1)=LEFT(@mk,1)
                and left(ruang,4) not in
(select left(ruang,4) from jadual)
                and
LEFT(ruang,1)>(select
max(LEFT(ruang,1))from jadual where
   RIGHT(mk,4)=@kelas and ke=@sesi)
                and ruang not in (select
ruang from jadual) order by ruang asc)
     if @ruang is null set @ruang=0 else
set @ruang=@ruang;
     update jadual set ruang=@ruang where
mk=@mk
      set @urt=@urt+1;
    end;
     set @sesi=@sesi+1;
    end;
```

The above process will produce data that is entered into a table as follows:

ID	MK	SKS	MSK	SLS	SESI	ROOM
1	101241A	2	07:00:00.000	08:40:00.000	1	11011
2	102241A	2	08:50:00.000	10:30:00.000	2	11132
3	203241A	4	10:40:00.000	14:00:00.000	3	12533
4	104241A	2	14:10:00.000	15:50:00.000	4	11274
5	205241A	2	16:00:00.000	17:40:00.000	5	12585
6	106241A	2	07:00:00.000	08:40:00.000	1	21021
7	107241A	2	08:50:00.000	10:30:00.000	2	31032
8	108241A	3	10:40:00.000	13:10:00.000	3	21276
9	109241A	2	13:20:00.000	15:00:00.000	4	31114
10	210241A	2	15:10:00.000	16:50:00.000	5	62522

IV. CONCLUSION

The experiments show that the scheduling results can plot all the lecturers according to the rules. The application of query optimization can make scheduling optimally, where the plotted schedule does not conflict. In addition, this query optimization can provide unique coding for each schedule so that the schedule is more effective and efficient in its design.

V. REFERENCES

[1] H. Babaei, J. Karimpour, A. Hadidi, “A Survey of Approaches for University Course Timetabling Problems”, in Computers and Industrial Engineering, vol. 86, pp. 43 – 59, 2015.

[2] J. Pandey, AK Sharma, “Survey on University Timetabling Problem”, in Proceedings of 3rd Int'l Conf. on Computing for Sustainable Global Development, New Delhi, India, 2016, pp. 160 – 164.

[3] SFH Irene, S. Deris, SZ Mohd. Hasyim, “A Combination of PSO and Local Search in University Timetabling Problem”, 2009 Int'l Conference on Computer Engineering and Technology, Singapore, Singapore, pp. 492 – 495.

[4] SI Hossain, MAH Akhand, MIR Shuvo, N. Siddique, H. Adeli, "Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search", Expert Systems with Applications, vol. 127, pp. 9 – 24, 2019.

[5] T. Ozsu and P. Valduriez. "Principles of Distributed Database Systems". 2nd Edition. Prentice-Hall, 1999.

[6] V. Poosala, E. Ioannidis, J. Haas, and J. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". In Proc. of the 1996 ACM SIGMOD Intl. Conf. on Management of Data, pages 294–305 June 1996.

[7] F. Chu, J. Halpern, and P. Seshadri. "Least Expected Cost Query Optimization: An Exercise in Utility". In Proc. of the 1999 ACM Symp. on the Principles of Database Systems, pages 138–147, May 1999.

[8] A. Hulgeri and S. Sudarshan. " AniPQO: Almost nonintrusive parametric Query optimization for nonlinear cost functions ". In Proc. of the 2003 Intl. Conf. on Very Large Data Bases, pages 766–777, Aug. 2003.

[9] Y. Ioannidis, R. Ng, K. Shim, and T. Sellis. "Parametric query optimization". In Proc. of the 1992 Intl. Conf. on Very Large Data Bases, pages 103–114, Aug. 1992.

[10] R. Avnur and J. Hellerstein. "Eddies: Continuously Adaptive Query Processing". In Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, pages 261– 272, May 2000.

[11] Z. Ives, A. Halevy, and D. Weld. " Adapting to source properties in processing data integration queries ". In Proc. of

the 2004 ACM SIGMOD Intl. Conf. on Management of Data, pages 395 – 406, June 2004.

[12] N. Kabra and D. DeWitt. "Efficient Mid-Query ReOptimization of Sub-Optimal Query Execution Plans". In Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data, pages 106–117, June 1998.

[13] V. Markl, V. Raman, D. Simmen, G. Lohman, and H. Pirahesh. "Robust Query processing through progressive optimization". In Proc. of the 2004 ACM SIGMOD Intl. Conf. on Management of Data, pages 659–670, June 2004.

[14] T. Urhan, M. Franklin, and L. Amsaleg. "Cost-Based Query Scrambling for Initial Delays". In Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data, pages 130–141, June 1998.

[15] S. Babu, P. Bizarro, D. DeWitt. "Proactive Re-Optimization". SIGMOD 2005, Baltimore, Maryland, USA, June 14-16, 2005