Graph Theory and NoSQL Database Applied to School Scheduling Problem

Jocivan Suassone Alves, Luídne da Silva Mota, Carlos Henrique Correa Tolentino
IFTO - Federal Institute of Education,
Science and Technology of Tocantins
Palmas, Tocantins, Brazil
Email: {suassone, luidne}@gmail.com, chtolentino@ifto.edu.br

***Abstract*—This paper presents a graph-based model for the school scheduling problem. A Web system was developed using the Neo4J graph-oriented non-relational database management system. The results show that investing in the modeling and use of a fully compatible database management system is worth the cost and effort, since the execution time for the algorithms was highly satisfactory. The modeling is extensible and supports representation of other aspects of the problem, while the architecture of the solution allows each of its components to be optimized without impacting the others, favoring the development of future work.**

Keywords*–*Graph; Scheduling Problem; Neo4j.

I. Introduction

School scheduling is a problem that affects educational institutions around the world. That is the case with the IFTO - Federal Institute of Education, Science and Technology of Tocantins, a public institution for secondary, technical and college education. The campus in the state capital Palmas alone, has 29 courses at different educational levels and more than 250 teachers.

This problem has already been used as a case study to validate several modeling and optimization techniques and there are many software products that help solve it. On the other hand, the cost of this software makes it difficult for public institutions with scarce resources to acquire it. In this way, the task is usually dealt with by a single person or team, who are usually overwhelmed. This fact increases the chances of obtaining inconsistent class scheduling as a result.

According to Sousa et al. [1], the problem of generation of schedules is NP-Complete, which means that it can-not be solved with polynomial runtime algorithms and the exact methods, meaning that, algorithms that always return the optimal solution normally run for prohibitive computational times. However, according to Saviniec et al. [2], for that reason, the approach to the problem must be made by heuristic methods which, although not guaranteeing an optimal solution, are able to generate satisfactory solutions in an acceptable runtime.

Due to nature of the problem, which requires robust tools for solving it, the human cost involved, the constant changes in the courses offered by the institutions, as well as the inconstancy of relationship between teachers-courses-students and other peculiarities of the public sector, it is appropriate to propose a solution that fits the IFTO profile using free tools and to explore its positive aspects.

Because school resources are better used in didactic and pedagogical than in administrative activities, this work presents the development stages of software for performing the automatic generation of class schedules for the IFTO - Campus Palmas. The main objectives are to provide:

- Graph-based modeling, which allows the use of low complexity algorithms;
- Storage in a graph-oriented NoSQL database, which makes object-relational mapping unnecessary, since it is common in this type of software, allows more efficient queries and runs part of the algorithm on the server itself;
- Use of an architecture of independent modules that allows the improvement of each one without impact on the others;
- Use of the Iterated Local Search (ILS) meta-heuristic to explore the solution search space.

Once the objectives are reached, a reduction of the effort, especially human, is expected in the assembling of the course timetable.

This work is organized as follows: After the Introduction, Session II presents related works, Session III presents the proposed modeling and formulation of the problem, Session IV presents the methodology and Session V presents the results. Finally, Section VI presents some considerations and future work.

II. Related Work

The main difference between the works that approach this topic using meta-heuristics is the modeling and the heuristic basis for solving each particular instance of the problem. In Freitas et al. [3], the class scheduling problem is solved using Genetic Algorithms and binary arrays for storing each solution. They developed a software called Kayrós, using Java programming language and the data is stored in an object-oriented database supported by DB4 6.0. The solution proposed in Viera et al. [4], also uses Genetic Algorithms but the modelling is based on an array where each element is a four-field structure (course, teacher, schedule[], vacancy[]). The last two are also arrays.

In Cassemiro et al. [5], a hybrid solution based on genetic algorithms and Tabu search was proposed. The model supported in a three-dimensional array was developed to allow the optimization of some aspects of the problem using a target function that considers, among other values, the number of collisions in each individual of the population, treating them as penalties.

A similar approach to this work was performed by Saviniec et al. [2]. Such an approach involves three algorithms based on ILS that are implemented separately and in combination and do not present any mechanism for storing the results. Despite the good results, modeling the problem involves complex mathematical elements that are not easy to reproduce.

In Catarino et al. [6], the performance of a relational database (MySQL) is compared to the performance of a NoSQL (Neo4J) [7]. The results show an advantage for the MySQL database for a small amount of data. However, as the amount of data increases the performance of Neo4J becomes higher. Finally, in Côrrea et al. [8], the performance of two databases was compared, considering insertion, update and query operations. The NoSQL database ran the inserts at 38% of the time in the relational database. In the update and consultation operations, the percentages were 6.46% and 2.69%, respectively.

III. Graph-Based Modelling

Generating schedules is a problem associated with school activity. Each teaching institution has particularities that need to be represented in the model. Such a model should also be able to store a solution to the exposed problem. In addition, the aspects to be optimized must be present.

The amount of constraints involved influences the complexity of the algorithm that solves the problem. In this way, generically, [2][9] one may classify the constraints into two main groups:

- Strong constraints (essential for solution consistency):
  - a teacher cannot teach two courses at the same time;
  - one class can not be in two courses at the same time;
  - classes of the same course must be held on the same day;
  - courses must be scheduled in the course in which the course is offered.
- Weak restrictions (non-essential):
  - avoid windows in the teachers' timetable;
  - minimize the number of days each teacher will be in the classroom;
  - If classes of the same course are not all on the same day, they should not be on consecutive days.

In this work, only the strong constraints were observed.

This work represents the elements of the scheduling problem using a graph in the form $G = (V, E)$ form, in which V is formed by the following subsets, which represent the different types of elements described as vertices:

- $P = \{p_1, ..., p_k\}$ represents the teachers;
- $D = \{d_1, ..., d_l\}$ represents the courses;
- $T = \{t_1, ..., t_m\}$ represents the classes;
- $H = \{h_1, ..., h_n\}$ represents possible schedules (week day and time interval).

The set of edges is also composed by typified edges, representing constraints and associations between the different types of vertices:

- $RH = \{rh_1, ..., rh_l\}$ represents the constraints between H elements connected to other P subset elements. For example, where a $rh \in RH$ connects a $p \in P$ to a $h \in H$ it means that p is unavailable for scheduling in h interval;
- $PA = \{pa_1, ..., pa_m\}$ associates teachers and courses, it defines that a $p \in P$ teacher will be the chair of a $d \in D$ course;
- $TD = \{td_1, ..., td_n\}$, which represents the courses and their respective classes.

Figure 1 shows a graphical representation of $G = \{V = \{P, D, T, H\}, E = \{RH, PA, TD\}\}$.

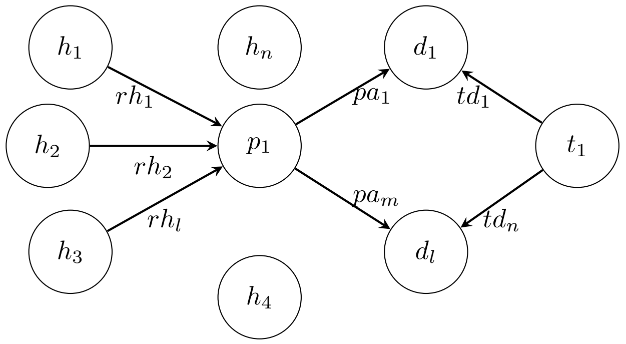


Figure 1. Graph-based model.

To represent the solution of the problem by indicating at which time each class will be performed, the following types of edges are added to the set $E = \{RH, PA, TD\}$:

- $HAT = \{ht_1, ..., ht_n\}$ associates a class with a schedule;
- $HAP = \{hp_1, ..., hp_n\}$ indicates that a teacher is in class at a given time;
- $HAD = \{hd_1, ...hd_n\}$ defines that the classes of a given course are performed at a certain time.

Considering the extension of the model with the inclusion of HAT, HAP and HAD edges, it becomes complete. Figure 2 illustrates a simplified scheduling where the highlighted edges hp_1, hd_1 and ht_1 says that the teacher p_1 teaches the d_1 course for t_1 class. In the same way, the highlighted edges hp_2, hd_2 and ht_2 says that teacher p_1 teaches the d_2 course for t_1 class.

IV. Materials and Methods

In order to achieve the objective of offering a low-cost tool for the IFTO, only free distribution software was used in the development of this work. To store the problem data to be captured and inserted into the model, the NoSQL Neo4J DBMS was used. This graph-oriented database allows the model to be stored in its original format, requiring no object-relational mapping that would increase the computational cost of the system. In addition, the Cypher language [10], used for data manipulation, allows the algorithm to determine the class scheduling to run on the server itself.

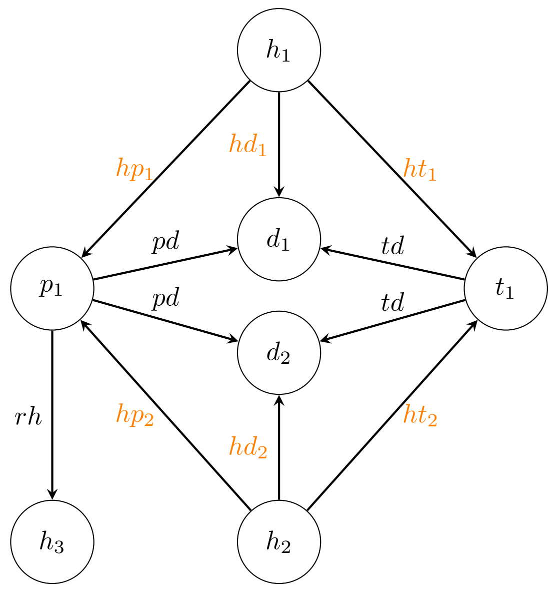


Figure 2. Schedule represented.

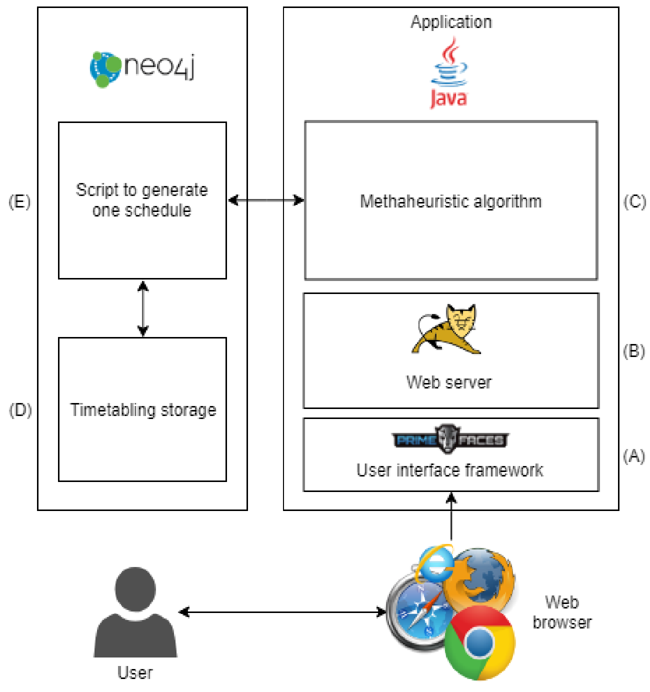


Figure 3. System architecture.

To explore the solutions space the meta-heuristic ILS was used. This stage of scheduling is performed by an application written in Java. The user interface is based on the PrimeFaces framework [11], running on a Tomcat 9.0 Web server. Finally, the Web system was developed using NetBeans IDE 8.2 [12]. Figure 3 illustrates the system architecture.

Looking at Figure 3, the bottom block represents the user view. On the right, the three internal blocks represent: (A) a framework to assist the user interface (which is optional), (B) the Web server and (C) the metaheuristic coded in a programming language. Note that these elements are independent and can be optimized and replaced without affecting the others, preserving compatibility.

The left block represents the DBMS, which includes storage in (D) and scheduling in (E). The main interaction occurs between blocks (C) and (E) where (C) receives and evaluates the schedules generated in (E).

V. Results

The algorithm developed to generate the schedules of a single course $d \in D$ related to a class $t \in T$ and a teacher $p \in P$, written in the Cypher language, is shown in Listing 1.

Listing 1. Internal Algorithm on Cypher Language.

```
1  match(t:Class)-[td:ClassCourse]->(d:Course
       {code: id })<-[pd:TeacherCourse]-(p:
       Teacher)
2  match(h:Schedule)
3  where not (h)-[:ClassSchedule]->(p) and
       not (h)-[:ClassSchedule]->(t) and not
       (p)-[:RTeacherSchedule]->(h)
4  with d, t, h, p limit 1
5  optional match(d)<-[ha:ClassSchedule]-(:
       Schedule)
6  with d, t, h, p, d.ClassPerWeek as nClass
       , count(ha) as nHad
7  where nHad < nClass
8   create(h)-[:ClassSchedule]->(p)
9   create(h)-[:ClassSchedule]->(t)
10  create(h)-[:ClassSchedule]->(d)
```

Having d (course) as input, line 1 finds the class t and the teacher p. The available schedules (line 2) are filtered so that only those free of any constraints remain (line 3). Next, line 5 checks if the course has already been scheduled, if the selected time interval is sufficient for the classes of the course (line 7), and in this case, if the connection between p, t, and d is made with the chosen time interval h (lines 8, 9 and 10), inserting the appropriate edges. Adapting structured logical reasoning to the Cypher language paradigm was a challenge encountered at this stage of the work.

Figure 4 shows the execution flow of the ILS. The algorithm presented in Listing 1 composes block 1, indicated. Note that the scheduling is not complete until all the subgraph $G = \{D\}$ is visited and each $d \in D$ has been associated with a interval $h \in H$ by an edge.

Steps 1, 2, and 3 of Block 1 in the Figure 4 are executed by DBMS, whereas the external steps to that block are executed by the application. Thus, different scheduling solutions can be generated from several courses sorting in array D [].

The tests were carried out considering that the association between teachers and courses, classes and courses and the constraints were previously defined, for example, by the school managers. Therefore, the problem is summarized in staggering the classes, since the possible conflicts were previously solved. To perform the tests, two scenarios were created: scenario 1, simpler, with 15 courses, 5 teachers, 3 classes and 20 schedules and scenario 2, more complex, containing 50 courses, 19 teachers, 10 classes and 20 schedules. The chart in Figure 5 illustrates the comparison of runtime in seconds with the PowerCubus software [13].

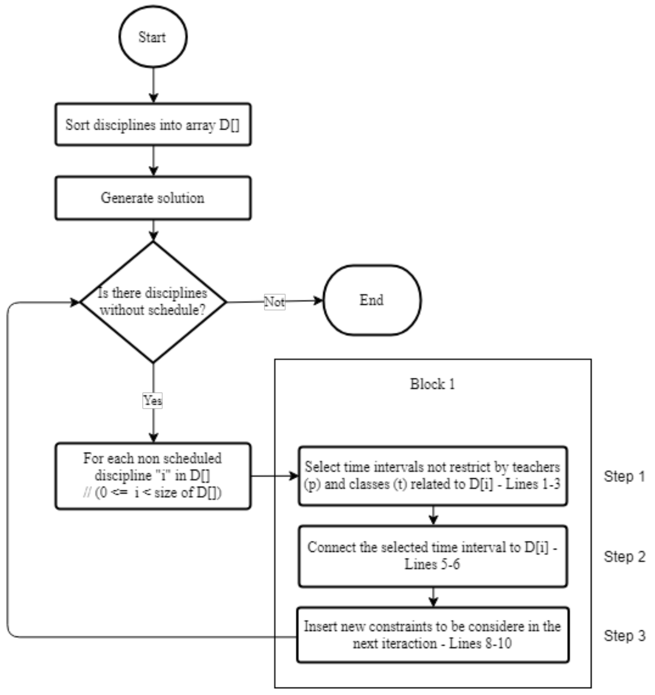

Start
Sort disciplines into array D[]
Generate solution
Is there disciplines without schedule?
Not
End
Yes
For each non scheduled discipline "i" in D[] // (0 <= i < size of D[])
Block 1
Select time intervals not restrict by teachers (p) and classes (t) related to D[i] - Lines 1-3
Step 1
Connect the selected time interval to D[i] - Lines 5-6
Step 2
Insert new constraints to be considere in the next iteraction - Lines 8-10
Step 3

Figure 4. Algorithm flowchart.

As one can see, the runtime of the solution proposed in this article is only 12.5% of the time spent by the PowerCubus software in the scenario 1. Scenario 2 could not be tested in the PowerCubus software because it is limited to a free version. However, the solution proposed in this article in the second scenario was executed using only 30% of the time spent by the PowerCubus software compared to the first scenario. Thus, even for a more complex case, our solution obtained a significantly more satisfactory result.

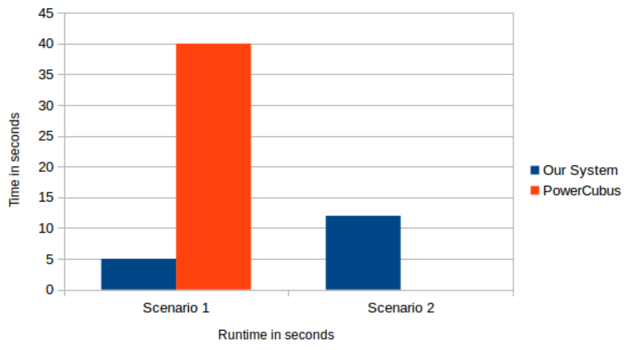


Figure 5. Our System x PowerCubus.

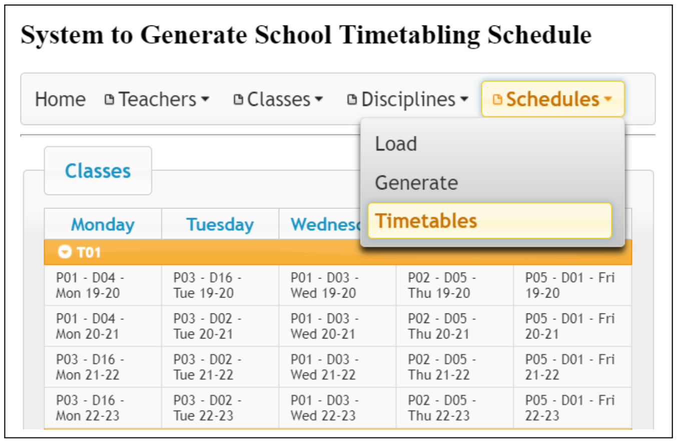
Finally, Figure 6 illustrates the interface developed and presents the result of scheduling for a single class.

VI. Conclusion and Future Work

The proposed modeling was efficient in representing the aspects of the scheduling problem. The model is general enough to allow inserting of new data, such as other types of constraints. The ILS presented satisfactory solutions and the algorithm executed directly in the graph-oriented DBMS was consistent and quick to execute. Note that the graph-oriented database allows queries to be performed without the need to cross-out data in cross and joins the are common in relational databases, being more direct and much faster. Furthermore, since Neo4J does not limit the storage of nodes, the solution proposed in this article allows us to deal with instances of any size for the proposed problem, meaning that it, it can be used in the long term by the IFTO.

Continuation of this work will involves the inclusion of the weak restrictions previously mentioned and also environment allocation for holding classes. Since the IFTO offers courses in different areas and requires the shared use of facilities such as classrooms, swimming pool, sports gym, auditoriums, computer labs, among others, this functionality will be very useful.

In addition, the user interface must be upgraded to improve usability, because, despite automated scheduling generation, human intervention will still be necessary in the process, e.g. for inserting data into the system.



System to Generate School Timetabling Schedule

Monday	Tuesday	Wednes		
T01				
P01 - D04 - Mon 19-20	P03 - D16 - Tue 19-20	P01 - D03 - Wed 19-20	P02 - D05 - Thu 19-20	P05 - D01 - Fri 19-20
P01 - D04 - Mon 20-21	P03 - D02 - Tue 20-21	P01 - D03 - Wed 20-21	P02 - D05 - Thu 20-21	P05 - D01 - Fri 20-21
P03 - D16 - Mon 21-22	P03 - D02 - Tue 21-22	P01 - D03 - Wed 21-22	P02 - D05 - Thu 21-22	P05 - D01 - Fri 21-22
P03 - D16 - Mon 22-23	P03 - D02 - Tue 22-23	P01 - D03 - Wed 22-23	P02 - D05 - Thu 22-23	P05 - D01 - Fri 22-23

Figure 6. Web page: Screenshot schedule for single class.

References

[1] V. N. d. Sousa, A. C. Moretti, and V. A. d. Podestá, "Programação da grade de horário em escolas de ensino fundamental e médio [Schedule the timetable in elementary and middle schools] ," Pesquisa Operacional [Operational Research], vol. 28, 2008, pp. 399–421.

[2] L. Saviniec, A. A. Constantino, W. Romão, and H. G. Santos, " Solving the High Timetabling Problem to Optimality by Using ILS Algorithms," Simpósio Brasileiro de Pesquisa Operacional, September 2013.

[3] C. C. Freitas, P. R. B. Guimarães, M. C. M. Neto, and F. J. R. Barboza, "Uma Ferramenta Baseada em Algoritmos Genéticos para a Geração de Tabela de Horário Escolar [A Tool Based on Genetic Algorithms for School Timetable Generation]," Faculdade Ruy Barbosa (FRB) – Salvador – BA - Brasil, September 2014.

[4] F. Vieira and H. Macedo, "Sistema de Alocação de Horários de Cursos Universitários [System of Time Allocation of University Courses]," Um Estudo de Caso no Departamento de Computação da Universidade Federal de Sergipe, São Cristovão, Brasil [A Case Study in the Computer Department of the Federal University of Sergipe, São Cristovão, Brasil], vol. 7, no. 3, March 2011.

[5] M. V. d. S. Cassemiro, "Desenvolvimento de um Modelo Híbrido Baseado em Algoritmo Genético e Busca Tabu para Resolução do Problema de Quadro de Horários Escolar [Development of a Hybrid Model Based on Genetic Algorithm and Tabu Search for Problem Resolution of School Schedules]," Simpósio Brasileiro de Pesquisa Operacional,

Salvador – BA - Brasil [Brazilian Symposium on Operational Research, Salvador – BA - Brazil], September 2014.

[6] M. H. Catarino, "Integrando banco de dados relacional e orientado a grafos para otimizar consultas com alto grau de indireção [Integrating relational and graphical database to optimize queries with high degree of indirection]," Instituto de Matemática e Estátistica da Universidade Federal de São Paulo, São Paulo - Brasil [Institute of Mathematics and Statistics of the Federal University of São Paulo, São Paulo - Brasil], November 2017.

[7] "The Native Path to Graph Performance," URL: https://neo4j.com/business-edge/native-path-to-graph-performance/ [accessed: 2018-10-14].

[8] T. d. S. Côrrea, D. E. C. d. Almeida, and A. F. G. Neto, "Comparação entre banco de dados relacional e não relacional em arquitetura distribuída [Comparison between relational and non-relational database in distributed architecture]," III Seminário de desenvolvimento em soa com cloud computing e conectividade, Instituto Nacional de Telecomunicações – INATEL [III Seminar of development in sounds with cloud computing and connectivity, National Institute of Telecommunications - INATEL], September 2017, ISSN: 2447-2352.

[9] E. A. Abdelhalim and G. A. El Khayat, "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)," Alexandria Engineering Journal, vol. 55, March 2016, pp. 1395–1409.

[10] "Intro to Cypher," URL: https://neo4j.com/news/explorando-o-cypher-a-linguagem-de-pesquisas-em-grafo-do-neo4j/ [accessed: 2018-10-14].

[11] "Why PrimeFaces," URL: https://www.primefaces.org/whyprimefaces/ [accessed: 2018-11-11].

[12] "NetBeans IDE - The Smarter and Faster Way to Code," URL: https://netbeans.org/features/ [accessed: 2018-11-11].

[13] "PowerCubus," 2016, URL: http://www.powercubus.com.br/ [accessed: 2018-10-04].