Academic Planner

Delgado Ruiz, David Mauricio dm.delgado10@uniandes.edu.co
Hernandez Cantero, Laura Sofía ls.hernandez10@uniandes.edu.co
Mendoza Arrieta, Camilo Andres ca.mendoza968@uniandes.edu.co
Murillo Castillo, Juan Guillermo jg.murillo10@uniandes.edu.co
Remolina-Gutiérrez, Maria Camila mc.remolina197@uniandes.edu.co

***Abstract*—This article presents an optimization application that helps undergraduate students of Universidad de los Andes (Colombia) that are enrolled in a double major or minor, to minimize the number of semesters required for their graduation. The application offers a detailed plan, semester by semester, with all the courses suggested to sign up for. It is also an iterative tool that takes into account the classes that the student has already approved. This article consists in the final project for the class *Modeling, Simulation & Optimization* (2017-10).**

***Keywords*—optimization, semestral planning, graphs**

I. Introduction

The problem of academic planning by semester or quarters is something each undergraduate student has to approach when enrolled in college. However, in the majority of the cases, each major offers a suggested flow of courses per period so that the degree can be obtained in the time the university claims.

Nonetheless, most universities offer the possibility to perform a double major or minor during the undergraduate studies. Then, the task of planning each academic period becomes non trivial and deploys a big number of combinations between majors and minors. For this reason, and taking into account that most of the students and parents want to minimize the time and money spent for college tuition, we want to present an optimization tool that solves this problem specially for major/minor merges.

All of us, the authors, are undergraduate students at Universidad de los Andes (Uniandes) in Bogotá, Colombia. In this university, by 2015 the 30.2% of the students were enrolled in a double major and 21% in a minor [1]. This means that a great portion of our school has to deal with the problem stated above. For this reason we chose our university to base the prototype of our application and to test the results of our solution.

II. Problem solution

In order to approach this problem, we started with a mathematical formulation followed by an implementation in the software GAMS. Then we created an interactive frontend website that allows to set the restrictions. Finally we run GAMS and other technologies in the backend to join all the application together. The details for each part are displayed in the next subsections.

A. Mathematical formulation

1) Parameters: In the following table we describe the conventions we will use during the article.

TABLE I. Table of conventions

Symbol	Meaning
S	Number of semesters
C	Number of courses
M_{ik}	Adjacency matrix between courses

The matrix M contains the information of the merge of the all majors and minors that the student has. It is defined as:

$$M_{ik} = \begin{cases} 0 & \text{if course } i \text{ has no relation with course } k \\ 1 & \text{if course } i \text{ is prerequisite of course } k \\ 2 & \text{if course } i \text{ is corequisite of course } k \end{cases}$$

Where $i, k \in [1, C]$.

2) Decision Variable: The decision variable we will use is:

$$x_{ij} = \begin{cases} 1 & \text{if I take course } i \text{ in semester } j \\ 0 & \text{the rest of the cases} \end{cases}$$

Where $i \in [1, C]$ and $j \in [1, S]$.

3) Objective function: The variable that we want to minimize is the number of semesters n:

$$n = \sum_i \sum_j j^5 x_{ij} \tag{1}$$

4) Constraints: The mathematical formulation takes into account 4 basic constrains to simplify the problem:

- A course is approved only once.

$$\forall i, \quad \sum_{j=1}^{S} x_{ij} = 1 \tag{2}$$

- The student sets a maximum of credits (c_{max}) by semester, that cannot be exceeded.

$$\forall j, \quad \sum_{i=1}^{C} x_{ij} * \text{credits}(i) \leq c_{max} \tag{3}$$

- If course A is a prerequisite of another course B, A has to be in a semester previous to B's.

$$\text{if} \quad x_{ij} = 1 \quad \Rightarrow \quad \forall a | M_{ai} = 1; \sum_{k=1}^{j-1} x_{ak} = 1 \tag{4}$$

- If course A is a corequisite of another course B, A has to be in a semester previous or equal to B's.

$$\text{if} \quad x_{ij} = 1 \quad \Rightarrow \quad \forall a | M_{ai} = 2; x_{aj} = 1 \tag{5}$$

B. *Computational implementation*

For the computational implementation of the optimization, we used the software **GAMS** [2]. We treated the solution as a mixed integer linear programming problem (MIP). We selected the solver CBC. The results were finally exported as a text file. The code that shows the implementation of the optimizer with a basic example is available publicly in our GitHub repository at the URL https://github.com/projectsuniandes/planeadorsemestral_optimizador/blob/gams/optimizador.gms.

After the optimizer implementation, we created a website in which the user could insert the basic restrictions and major/minor combinations of his/her choice. For this, we created an **AngularJS** frontend that connects to a backend in **NodeJS**. In order to call the GAMS implementation, we executed it by command line and read the text file it exported. The code that shows this is also available publicly in our GitHub repository at the URL https://github.com/projectsuniandes/planeadorsemestral_back.

Finally, as for the data used in the application, we created a scrapper that goes through the courses offer website of Uniandes. We cleaned it and created a database in **PostgreSQL** that the backend calls in order to create the matrix M of the merged courses. However the information that was obtained automatically contained the information about prerequisites, corequisites and course details, but not the courses associated to each program. For this reason we had to fill the information by hand only for 6 of the majors offered at Uniandes.

III. Simulation

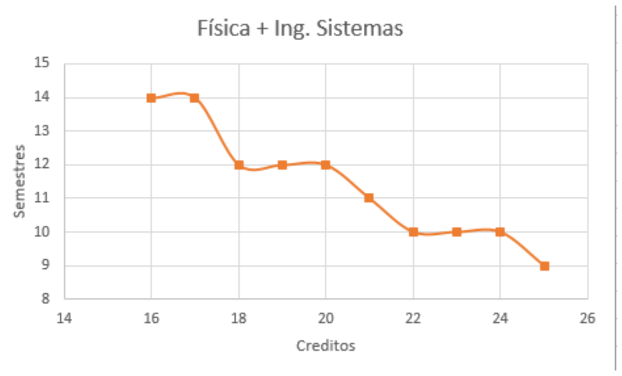
In order to simulate the performance of our optimizer we decided to implement a real API reachable through HTTP REST requests. Using a simulator that varies the parameters such as programs and maximum number of credits we simulate load on our optimizer component.

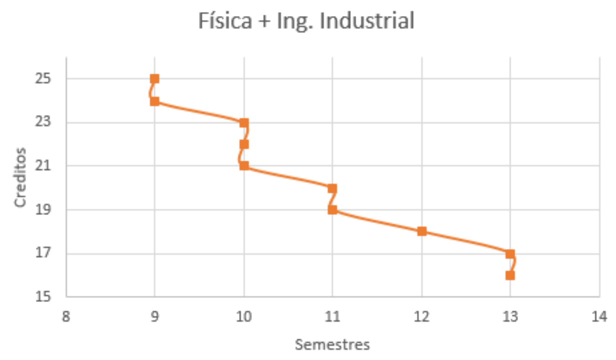
The format of the request includes the first program, the second program and the maximum number of credits the student want to take each semester. Our optimizer will receive the request, access the database of courses, find all that belong to the programs specified and then make a merged list of them. After that, it will again access the database for every course in the list and find all its requisites and corequisites, in order to build the matrix representing the graph that the optimizer will use as input data.

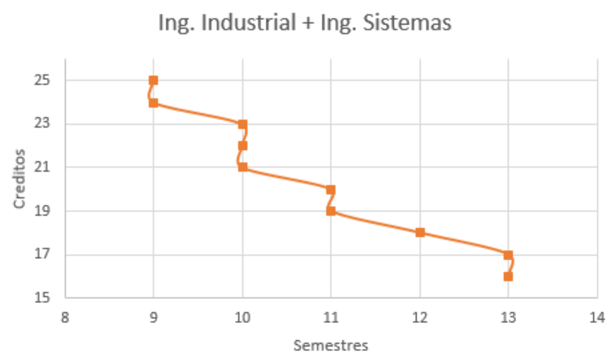
When the optimizer finishes the optimization, it will return the optimized result to the client via an HTTP response. Our Javascript simulator send requests to the API 100 times for each possible combination, each changing the parameters and measuring response time of the systems, minimum number of semesters possible and credits seen last semester for the selected optimization preferences.

On each iteration, our simulator collect all the results of the request and finally iterations are finished, it will average the response time for every combination and output data in CSV format for easy access and visualization in other programs.

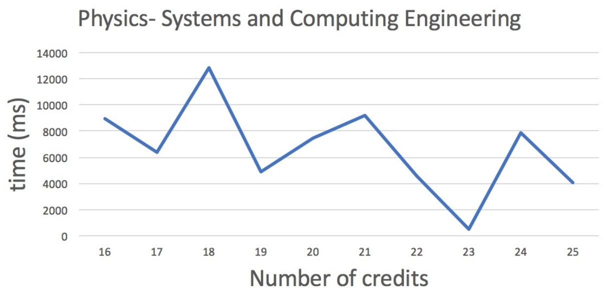
We obtained the following results for semesters vs credits:

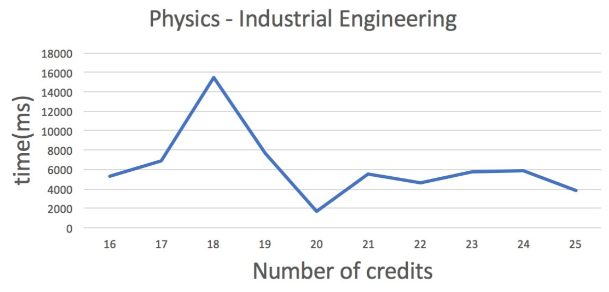


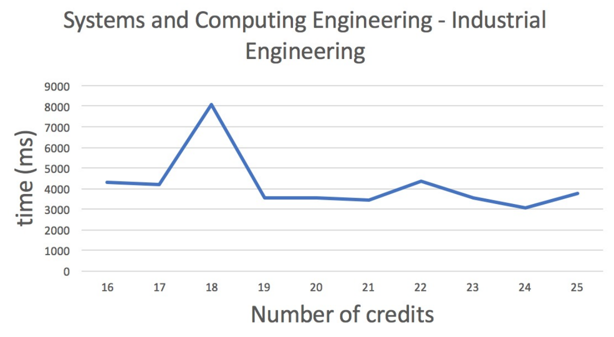




We obtained the following results for semesters vs average time of response:



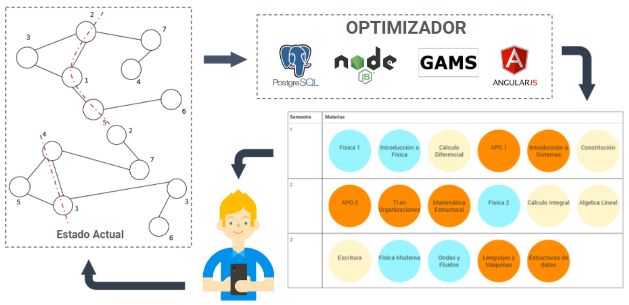




IV. Results

A fully functional web application was developed as part of this project. A public launch of this site was not possible under the course because of licensing problems with **GAMS**. It was released as part of the Product Design and Innovation Fair on May the 8th, at Uniandes. A short demo can be seen at the URL https://www.youtube.com/watch?v=V6zJbEuhjT4. The web app uses input from the user to calculate the recommended distribution of courses over the remaining semesters. Users can select their two majors and maximum number of credits by semester. Then they must select their taken courses, so the app can calculate the optimized remaining semesters.

This process is iterative and can be performed at any stage of the student. It doesn't have to happen at first semester necessarily. In summary, we take some directed graphs (that represent the flow of a major) at a certain stage. We merge them and passed them through an optimizer enhanced with the technologies mentioned previously. Finally we present the academic plan. This process is represented in Fig. 1, where the table displays the results as seen in the web app:



Semestre	Materias					
1	Física 1	Introducción a Física	Cálculo Diferencial	APO 1	Introducción a Sistemas	Constitución
2	APO 2	TI en Organizaciones	Matemática Estructural	Física 2	Cálculo Integral	Algebra Lineal
3	Escritura	Física Moderna	Ondas y Fluidos	Lenguajes y Máquinas	Estructuras de datos	

Fig. 1. Cycle

V. Conclusions and Future Work

A. Conclusions

This project leaves the following conclusions:

- We created an application that minimizes the number of semester a student needs to finish any combination of majors and minors and offer a detailed plan semester by semester of which courses to enroll in each period.
- Using the capabilities of GAMS and the theory learned about MIP problems optimization, it was possible to make a boring and time-consuming task in a matter of seconds.
- The course offer for each semester is not an easy task for the university because they have to estimate based on previous years what classes the students want to take. It often happens that the university need more or less courses, which translates into money loss. With

this solution they could get a better estimate of the classes that students want to take.

B. Future Work

As for future work we have several ideas to improve our implementation of the solution. These are:

- Fill the database with all the majors and minors offered by Uniandes. Specially, try to find an automatic way to do it by talking to the university.
- Implement user accounts so that each student preferences are saved.
- Implement program coordinators' accounts, so that for each program the its coordinator can update information and add specific restrictions that don't apply to the whole university.
- Integration with next semester schedule, so users can find courses with their preferred hours. Specially, so that the application doesn't suggest combinations that cannot happen due to courses that overlap in their schedules.
- Automatic integration of new courses, to make the application easily scalable.

As for the final future work idea, we would like to implement this software in other universities, after finishing the user tests at Uniandes.

Acknowledgments

The authors would like to thank the professor Yezid Donoso for his counseling in the objective function definition and the license to test and use non-demo solvers. Also to the teaching assistant German Montoya for his help with the GAMS implementation.

References

[1] D. de Planeación y Evaluación Universidad de los Andes, "Boletín estadístico uniandes 2015." [Online]. Available: https://planeacion.uniandes.edu.co/dmdocuments/Boletin_estadistico_2015.pdf

[2] G. D. Corporation, "General Algebraic Modeling System (GAMS). Release 24.0.2," *ApJ*, vol. 812, p. 123, Oct. 2015.