Application of scheduling techniques to Teacher-Class assignments

Pedro Ferreira (70467)
Instituto Superior Técnico - Universidade de Lisboa

October 2015

Abstract

Nowadays, it is more important than ever to be as efficient as possible. At Instituto Superior Técnico, Teaching Service Distribution is one of the steps that must be accomplished to get the next academic year ready. We created a system that helps carrying out this task. It is divided into 3 components: Web-platform, database and TeachOpt Solver. TeachOpt Solver is responsible of the automatically assignment the professors to courses. Literature pointed two main areas and techniques that are used to perform this task: Operations Research by solving Integer programming model instances and Artificial Intelligence based on Local Searches. We tested both techniques. From the tests we performed, we concluded that Operational Research techniques performed the best. They perform quickly to generate high quality solutions. Even though the solutions generated may need same manual adjustment, the system dramatically reduced the time it takes to have all professors assigned to courses.

1 Introduction

Schools, universities, and other educational institutions, in preparation of each academic period, have a complex and sometimes difficult task in hand: the allocation of the faculty team and rooms to the courses to be taught. Planning each academic period involves several tasks such as forecasting students' enrollments, predicting students' choices for optional courses, scheduling courses, assigning the faculty team to each of the courses also known as Teaching Service Distribution (TSD), and scheduling exams. Each school has its own way of planning the academic period. This thesis focus on the TSD of Department of Computer Science and Engineering (DEI), Instituto Superior Técnico (IST).

1.1 Teaching Service Distribution Problem

As we concentrate our efforts on TSD problem, there are some fundamental concepts that we need to have in mind. This problem consists on assigning a set of professors to a set of classes. Although it looks like a straightforward process, we must take into consideration that each professor has qualifications and also has a number of credits or "working hours", that indicate which/how many classes he/she will taught, respectively.

We must also consider that in each course, there are, typically, two types of classes, recitations and lectures and each type has a limit on the number of students that may attend to one session. As, every student must attend both types of classes, there will be multiple session of the same type of class.

1.2 Actual Method

Today, at DEI in IST, the process of generating the TSD is made by hand, using an excel file. This file aggregates all the information about the available professors and courses.

The person in charge for this task, by hand, tries to satisfy all the professors' preferences and constraints.

It is possible to identify several steps in this process: *1*) All the information about the forthcoming courses is reviewed, as well as the estimation of enrolled students in each course. *2*) Professors are asked about how many hours they have to teach and how many credits they must have in that year. *3*) The person in charge of the TSD gathers all the information and, by hand, tries to assign all classes to professors in order to fulfill all the requirements. *4*) In the last step, after students have registered for courses, the number of enrolled students are checked and the previous forecast numbers are now changed to the real ones. Finally, the calculation of the professors' credits in that year can be made.

2 Related Work

The problem that we address in this thesis is classified as a resource scheduling problem consisting in finding out combinations of tasks or activities, subjected to a certain number of constraints, and assigning them to resources

according to certain rules and constraints. Both the combinations of tasks and the resources assignments aims at optimizing some goals.

2.1 Techniques

Operational Research (OR) and Artificial Intelligence (AI) are the fields of research that most actively contribute to the solutions of resource scheduling problems.

2.1.1 Operational Research

Integer Linear Programming (ILP) is the most used contribution from OR in the solution of resource scheduling problems by describing a wide range of problems using only mathematical expressions and solved with problem-independent techniques that can be applied to any specified problem. For most problems, those techniques produce optimal solutions. However, under certain conditions, the problems must be relaxed and those techniques adapted in order to decrease the time needed to produce an optimal-enough solution. A common approach is to specify the model is based on a set covering problem.

Branch and bound is one of the algorithms that are used to obtain an optimal solution for ILPs. This algorithm relaxes the integer constraints and then, based on the obtained solution (without the integer constraints), introduces new constraints in order to limit the range of variables. This repeats until it reaches the integer solution (Lawler and Wood, 1966). One factor that decreases the performance of such algorithm is the number of variables.

2.1.2 Artificial Intelligence

AI offers different methods to solve resource scheduling problems. It tries to adapt generic ideas in order to build/search a solution. Local searches and genetic algorithms are the main methods that are focused in the AI literature related to resource scheduling problems because they can produce good results.

Local Searches Local searches are widely used in this resource scheduling of problems. Local searches start from a partially built solution and little changes are made at each step, also known as moves, knowing that a better solution may be produced after a certain number of steps. With these techniques, we may have more control over the overall process (Van den Bergh et al., 2013).

They can even produce better results than OR Maenhout and Vanhoucke (2010). The following local searches are the most common and widely used: *i*) Hill Climbing; *ii*) Simulated Annealing (Kirkpatrick et al., 1983); *iii*) Tabu Search (Glover, 1989); *iv*) Variable Neighborhood Search.

(Lü and Hao, 2012) proposed a local search that can change its behavior during the search process according to its perception of the success. At the beginning of the solution process, it uses a constructive heuristic to obtain an initial solution that will be improved. Afterwards, it uses three different strategies to improve the quality of the solution: *i*) Intensive Search; *ii*) Intermediate Search; *iii*) Diversification Search. Each strategy is chosen based on the improvements of the objective function at each step.

As shown in (Lü and Hao, 2012), the correct selection of multiple possible changes (moves) of a solution can improve results obtained as it allows to cover a much larger search space.

Genetic and Evolutionary Algorithms Another techniques that is being used to solve resource scheduling problems are Genetic algorithms. These techniques try to incorporate ideas from biology and how combinations are made in the natural world to optimize a solution. The main idea is that good characteristics of a solution are kept, and bad characteristics are dropped.

A hybrid scatter search was proposed in (Maenhout and Vanhoucke, 2010) based on the basic concepts of the canonical genetic algorithm (Whitley, 1994). This is an evolutionary approach that showed to produce better results than variable neighborhood search and the a Branch-and-Price procedure (branch-and-bound with column generation (Barnhart et al., 1998)). It could produce solutions 3% better than variable neighborhood search and up to 30% better than de Branch-and-Price.

2.2 Application Areas

To the best of our knowledge, there are a few published works on TSD alone (section 2.2.2). Other similar problems in the literature like *crew scheduling/rostering* applied to airlines and railways companies, as well as nurse roster scheduling contribute with helpful idea that can be used to solve TSD problems.

2.2.1 Educational Timetabling

Preparing an academic year at an university is a complex task. It is possible to identify five steps (I and Laporte, 1998). Teaching service distribution or Teacher assignment problem is one of those five steps of course scheduling. Sometimes the allocation of time, space and people is done at the same time, in a single step. This is called timetabling (Lewis, 2007). Examples of such solution are (Lü and Hao, 2010; de Grano et al., 2009; Schimmelpfeng and Helber, 2007).

Most of the work is based on modeling the timetabling problem as an assignment problem. An ILP model is defined and its main component is a decision variable X_{ijk}

that take the value 1 when teacher i teaches class j on time k and 0 otherwise. Then, several constraints over variable X are also defined to address real restriction. (Schimmelpfeng and Helber, 2007; Al-Yakoob and Sherali, 2006)

Meta-heuristic methods are widely studied on literature. Typically these methods are used in a two stage approach. The first stage is used to construct a feasible solution, i.e. a solution that at least do not violate any hard constraint. Then, in the second stage, the solution obtained is improved in order to avoid soft constraints violations. Genetic algorithms are also used to tackle this kind of problems.

A recent work on a similar problem (Phillips et al., 2015) used an ILP model to solve a class assignment problem (instead of assigning teacher, they assigned classrooms to class). They proposed an exact method to achieve the best solution considering several measures in a sequentially process. At each step one criteria is optimized and then, in the next iteration the criteria from the previous iteration is added as an upper bound constraint.

2.2.2 Teacher Assignment Problem

Gunawan and Ng used an Indian University to propose an algorithm based on two local searches (Simulated Annealing and Tabu Search) and they showed that their proposed algorithm perform better than manual allocation and even genetic algorithms.

A very similar problem to ours was described in (Gunawan et al., 2008). The approach is based on a Genetic algorithm and it assumes that the faculty team is complete by the time the TSD is done. Their proposal was compared with the hand-made allocations and it is clear that the automatic one is better.

2.2.3 Staff Scheduling

In the transportation domain, the main goal, that is relevant to our work, is to have for each employee a monthly schedule that covers all the transportation work required, minimizes the cost for the company, and, at the same time maximizes the satisfaction of the workforce.

Crew rostering is very similar to our problem although our restrictions are slightly different. One of the major similarities is that each employee must achieve a certain number of flight credit (Gamache et al., 1998). In the health sector, scheduling nurses is also a similar problem.

Most of the approaches used for staff scheduling are based on the models presented in section 2.1.1. An ILP model is formulated and different techniques applied.

Different goals and constraints lead to different approaches. A complete model that addresses the crew scheduling problem for large assembly lines was proposed in (Sabar et al., 2008). In this model, the objective function is presented as a minimization of costs. Those costs are not only the normal ones, like cost for assigning an activity to a certain employee but also penalty costs for deviation from preferred characteristics for each employee.

A similar objective function was also defined in (Eiselt and Marianov, 2008). Here, the objective function is defined as a sum of different costs in order to minimize several measures.

The local search proposed by (Lü and Hao, 2012) was applied to nursing scheduling

A hybrid scatter search was proposed in (Maenhout and Vanhoucke, 2010) to solve a crew rostering problem. Another genetic algorithm was proposed to in order to solve the problem of a subway system. It was compared with other methods (tabu search and greedy algorithm) and it performed better for most of the instance of the problem (Elizondo et al., 2010).

3 Solution

Before we describe our system, we need to explain in detail what is the problem we address, which parameters we have to consider, and, which restrictions/constraints we have to take into account.

3.1 Problem Formulation

Our problem consists in assigning a *schedule*, that is a group of classes, to each professor according to some restrictions and guided by certain goals. In our approach, the classes that must be lectured are the set of tasks that must be accomplished and all the professors (employees) are the available resources. At IST, each professor is associated with two scientific areas (skills), each one representing a group of courses. Each course, according to the number of enrolled students must have a certain number of classes (course sections) that must be offered in each week due to the fact that a classroom has limited seats.

As an effort metric, each professor indicates a number of *teaching credits* that he/she needs to achieve that academic year. Each class, based on its duration, semester length (number of weeks that it lasts), enrolled students, course's year are valued in terms of teaching credits. So, the idea is that each teacher will be assigned to as many classes as his teaching credits allow.

The equilibrium must be achieved considering our secondary goals: *a*) Assign professors to classes within their scientific areas; *b*) Avoid part-time professors; *c*) Satisfy, as much as possible, the preferences of each professor.

There are restrictions that must be satisfied as well in order to consider a TSD as valid: *a*) All classes must be

covered; *b*) The sum of the credits of the classes taught by each professor must not be less than the expected credits (plus/minus a threshold); *c*) All professors must have classes assigned.

For each academic year, professors must state their preferences but they can do it in the most varied way. In order to have a uniform preference's system, each preference represent a characteristic that must taken into account on the TSD. A professor can state their preferences about: *i*) Semester; *ii*) Campus; *iii*) Course; *iv*) Type of Class; *v*) Type of Class of a specific Course;

The professor must also rate each choice with a preference from -10 (not desired) to $+10$ (highly desired).

This model for preferences assumes that anything that is not stated has a neutral desire, as if it were stated with preference 0.

3.2 Data Models

In order to automate the problem described in Section 3.1, we must gather all the information that will be relevant in our solution.

We must distinguish two parts of the model: one is responsible for storing the information about professors and courses details (Generic Model); the other store only specific data about the classes and professors that is dependent of the academic year (Specific Model). Figure 1 shows a UML representation of both data model.

3.2.1 Generic Model

A **professor** has some identification data, a rank and two scientific areas that indicate the courses that he/she is able to teach. In some cases, it is possible to teach courses outside those areas. We will not explore the attribute Rank nor the ability to teach courses from other scientific areas beyond their two scientific areas.

A **course** belongs to a scientific area, it has a location (IST is divided into three campi) and a semester. It is also associated with a certain course year (degrees are divided into years, and each course year has a set of courses that students have to attend), as well as a teaching workload.

3.2.2 Specific Model

Concerning the professors' information that varies each academic year (**Prof Details** in Figure 1), it is only necessary to store their expected credits.

The model also needs to represent estimates of the enrolled students and, consequently, how many credits a course will value. It is important to have a forecast of enrolled students. Dividing the forecast between first attendees and repeaters allows to better estimate the number of course sections as repeaters tend to go less to classes (**Course Details** in Figure 1).

Each **class** has a type (lecture, recitation, seminars), an hour duration, and a duration in terms of weeks in the semester.

A **preference** is associated with a professor. It has a type, an attribute that indicates the real desire like which campus (A or T) or semester (1 or 2) or even both, as well as the rate.

3.3 Techniques Explored

In this Section, we provide a description of the techniques used in our work. We explored techniques from: Operational Research and Artificial Intelligence.

3.3.1 Operational Research

From the literature review, it was clear that there are two different models that can be applied. One is based on resource assignment, where we directly assign a resource to a group of tasks. The other is based on schedule assignment, where we assign resources to schedules.

Resource Assignment Let us assume that we have a decision variable X_{ctp} which contains a positive integer representing the number of classes of type t of course c that the professor p will teach.

Let us also assume a set of auxiliary parameters:

- P_{ctp} - Preferences value of classes of type t of course c to the professor p;
- V_{ct} - Teaching credits value of classes of type t of course c;
- M_p - Teaching credits of professor p;
- Bl and Bu - Lower and upper bounds for the teaching credits;
- LC_{ct} and UC_{ct} - Minimum and maximum number of times that the class type t of course c has to be taught, respectively;
- H_{ctp} - Qualification of professor p to teach classes of type t of course c;
- ω_t - Set of types of classes;

If we denote by Tp the total number of professors and Tc the total number of courses, expressions 1 to 4 represent all the constraints we need to have.

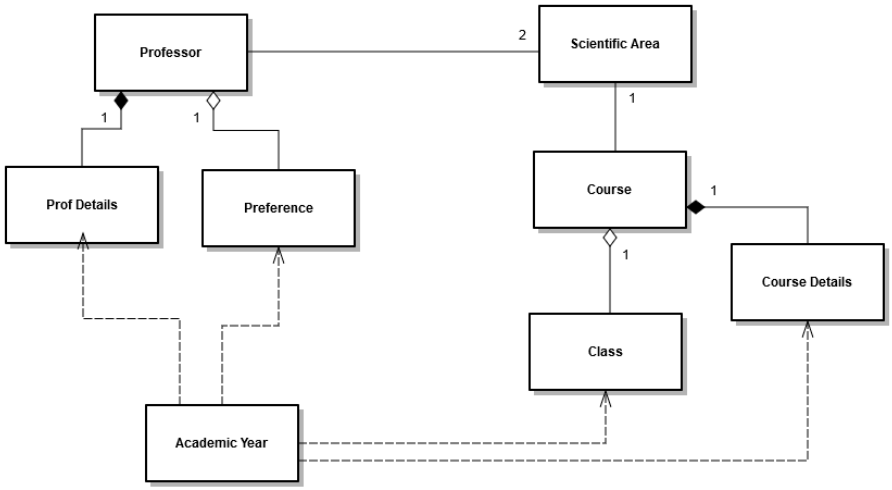

Figure 1: UML representing the both data models

$$M_p - Bl \leq \sum_{t \in \omega_t} \sum_{c}^{Tc} V_{ct} X_{ctp} \leq M_p + Bu \qquad (1)$$

$$p = 0, 1, .., Tp$$

$$\sum_{p}^{Tp} X_{ctp} \in [LC_{ct}, UC_{ct}] \qquad (2)$$

$$X_{ctp} \geq 1 \implies H_{ctp} = 1 \qquad \forall_{ctp} \qquad (3)$$

(4)

The only piece missing is the objective function. In this case we need to maximize the preferences for each professor:

$$\sum_{p}^{Tp} \sum_{t \in \omega_t} \sum_{c}^{Tc} P_{ctp} X_{ctp} \qquad (5)$$

This model has some limitations: a) It is not possible to have restrictions over the number of courses a professor will teach; b) It is not possible to impose bounds for the number of courses a professor may teach. To solve this problem, we introduced a binary decision variable C_{cp} that has the value 1 if professor p teaches at least one class of course c, and a constraint responsible for updating this decision variable:

$$\sum_{t \in \omega_t} X_{ctp} > 1 \implies C_{cp} = 1 \quad p = 0, 1, ..., Tp \quad c = 0, 1, ..., Tc \qquad (6)$$

With this variable, we can constraint the number of courses a single professor can teach by adding the restriction:

$$\sum_{c}^{Tc} C_{cp} \leq k \qquad p = 0, 1, 2, 3, .., Tp \qquad (7)$$

This ensures that no professors teaches more than k different courses, even tough they can teach more than one class in each course.

The model presented earlier represents the basic model for our problem. In order to improve and enrich our model we had another objective: minimize the sum of all differences between the real value of the assigned classes and the desired teaching credits (Equation 8). Another equivalent objective is to minimize the average differences between the real value of the assigned classes and the desired teaching credits (Equation 9).

$$\sum_{p}^{T_p} \left| M_p - \sum_{t \in \omega_t} \sum_{c}^{Tc} V_{ct} X_{ctp} \right| \qquad (8)$$

$$\frac{\sum_{p}^{T_p} \left| M_p - \sum_{t \in \omega_t} \sum_{c}^{Tc} V_{ct} X_{ctp} \right|}{T_p} \qquad (9)$$

Even tough, ILP do not allow multiple objectives in a straightforward way, there are at least two methods that allow to have multiple objectives. The first method is the composition of multiple objectives in a single expression such as: $Obj(X) = \alpha_1 O_1 + \alpha_2 O_2$ where O_1 and O_2 are the individual objectives and α is a coefficient that represents the relative importance of the objectives.

The second method consists on sequentially optimizing with a single objective at each time. We start with the basic model, solve it, and store the objective value (S_i) of the solution that was found. Then, we add a constraints to the basic model: $O_i = S_i$ and solve it again. If there are more objectives to be optimized in the next iteration, we repeat the process, but considering the model with all constraints previously added.

These two methods for optimizing multiple objectives originated two different models where the two objectives expresses in Expression 8 and 9 are used as a weighted objective function or as a multi-objective sequential optimization problem.

(Pre-generated) schedules assignment The schedules assignment model assumes that we first generate all possible schedules (combination of classes) for each professor or at least generate a very large number of them.

In order to generate the schedules, we consider all the combinations between all the classes a professor can teach, that are inside the interval of expected credits. We could use all possible schedules, however we applied heuristics to decrease the number of schedules per professor, such as eliminate all schedules that have negative or zero rating assuming that the professor has stated at least one preference.

Then, for the model definition, we have a decision variable X_{sp} that takes the value 1 if the schedule s for professor p is chosen.

Some additional parameters need to be set to correctly define the constraints:

- A_{scp} that indicates how many classes c are present on schedule s for professor p.
- LC_c and UC_c - Minimum and maximum number of times that class c has to be taught, respectively.
- P_{sp} - Preferences value of schedule s to the professor p

Let denote by Tp the total number of professors, Tcc the total number of classes and Ts the total number of schedules.

In this model, we have to ensure that the selected schedules comply with the minimum and maximum number of times professors teach a certain class.

$$\sum_{p}^{Tp}\sum_{s}^{Ts} A_{scp}X_{sp} \in [LC_c, UC_c] \qquad c = 0, 1, ...Tcc \quad (10)$$

Another important aspect is to ensure that only one schedule is selected by each professor.

$$\sum_{s}^{Ts} X_{sp} = 1 \qquad p = 0, 1, 2, .., Tp \quad (11)$$

The objective is still to maximize the preferences values of the selected schedules.

$$\sum_{p}^{Tp}\sum_{s}^{Ts} P_{sp}X_{sp} \quad (12)$$

Constraints such as the maximum number of courses per professor and others are incorporated in the schedule generation process. We only generate valid schedules.

3.4 Artificial Intelligence - Local Search

AI also gives contributions to our problem. We have to define three different aspects concerning local searches: *i*) how to get the initial solution. *ii*) what moves we consider *iii*) which search strategies to use

Local search only works for complete solutions even though the starting point may be an invalid solution. We built a constructive heuristic in order to obtain a reasonable solution and then the local search tries improve it. This heuristic sequentially tries to assign each class to the professor who have the highest sum of preferences rating that involve that class.

In order to develop a local search, we had to define the moves that we will use. Based on the literature review, we consider a move to be: a *swap* of classes between 2 professors, and a *direct move* of a class from one professor to another.

Concerning the strategies, we decided to use Hill Climbing and Tabu Search. These strategies combined with our moves can produce a large number of changes in a short period of time. This fact can help on achieving a better solution, i. e. improving the initial solution. In both strategies the objective function is the weighted sum of the preferences value and the total deviation of credits. The Tabu list used in the Tabu Search can have single moves or whole states and the size will determine the memory aspect of this search strategy.

4 Architecture

In this chapter, we present the details about the architecture of our solution. Our solution was conceived to be modular, having in perspective future uses of it. One fundamental aspect of our system is versatility as it allows to test multiple and equivalent components that help to improve our solution and build a more valuable system.

4.1 General Overview

Our system is divided into three main independent components as in shown Figure 2:

- Web Platform - used by professors and department's staff to introduce data;
- Database - to store all data in a persistent way;
- TeachOpt Solver - responsible for the automatic step of generating a TSD.

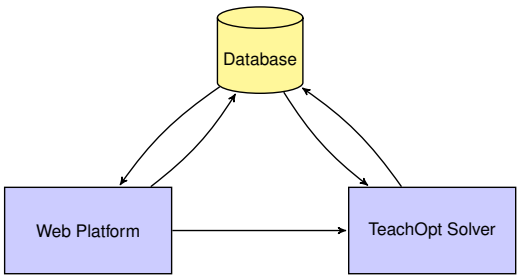

Figure 2: TeachOpt architecture schema

The arrows in Figure 2 represent data flows between the components. The interaction with the database are read/write actions, but the interaction between the web platform and the software to generate the solution is just an activation interaction. In this case, the platform starts the software, however the result is written in the database and not directly returned to the web platform.

4.2 Web Platform

In order to automate the process of generating a TSD, a web-based platform was built. We decided to use a web interface because it allows users with a Internet connection to be autonomous and insert his/her preferences and view the generated TSD.

In the web-platform, we separated the features into areas according to the user that accesses it. Users are divided into two different user profiles:

1. Professor - each professor has an account with his/her user profile.

2. Department - every person in the department access and modify the information as well as plan the TSD. This is an administration profile.

4.3 Database

The database component does not have any special feature or function. It is used to store data in a persistent and consistent way. Both the web platform and the TeachOpt solver connect to it to read information and store other information.

4.4 TeachOpt Solver

Concerning the automatic step of this process, we created a software that can generate the TSD. This software makes a bridge between the database and the solver that is capable of solving OR problems or the AI solvers that use local searches.

In order to generate a TSD, it starts by reading the database and create an intermediate structure. This intermediate structure helps to avoid dependency of the solver with the specificity of the database. Then, the solver is initiated and the model is configured based on the formulas specified in Section 3.3.1. Finally, the variables/parameters are filled according to the intermediate structure.

The interface of the software also allows to configure some parameters about the solver and to view the generated TSD before it is saved into the database.

It is important to note that if no feasible solution is obtained, i.e. if no solutions that respect all the constraints is found, one of the following error messages will appear: - *UpperBound - Credits*; - *LowerBound - Credits*; - *LowerBound - Classes*; - *UpperBound - Classes*; The first two messages appear when, for at least one professor, the credits constraints defined in Equation 1 are violated. The other two appear when the constraint about the number of classes of a course is not right (Eq. 3).

We do not have any specific mechanism on the OR models to address the problem of no feasible solution being found.

5 Implementation

To implement the features discussed in the Section 4, we made some decisions about the implementation and tools used. The usage of well-know tools allowed us to avoid simplifications of the architecture. As we do not explore any special feature from the adopted tools, all parameters of all tools were set to the default values.

5.1 Simplifications

The simplification that was made to our proposed solution was the ability of the web platform to start TeachOpt Solver. We did not implemented this feature due to issues with the infrastructure where the solution run. So, the TeachOpt Solver must be started by running its executable. However this simplification does not affect the purpose of our solution.

Every other feature described in Section 3 was implemented and is fully functional in the web platform.

5.2 Web Plaform and Database Technologies

To develop the web platform, we used PHP. Concerning the look & feel and to avoid having a plain HTML site, we used Bootstrap [1] to give a nicer look to the web pages.

The database where all the data is stored is the one of the key pieces of the system. Note that we just need to use the basic operation of databases to access and manipulate the data: Select, Insert, Update and Delete. So, we chose to have a relational model, rather than other possibilities, because it is widely used as it allows

[1] http://getbootstrap.com/

to have the model presented in Section 3.2 with minor changes. As we are just interested in those base features, our choice of DBMS was MySQL[2].

5.3 TeachOpt Solver

5.3.1 Generic Behavior

TeachOpt Solver is the core software of our system. In order to easily test different techniques, we implemented a two step process: *1*) read the DB data into the intermediate structure; *2*) start the solver using the intermediate structure. The first step is done by a single DB reader, as we do not have multiple sources of information. The second step is where the TSD solution is created (solved). As stated in Section 3.3, we needed to test several techniques, so we created multiple solvers. These solvers have the same structure and interface but implement different techniques.

5.3.2 Solvers

Solvers represent one of the major pieces of the system. They are responsible to generate the TSD solution.

There are several software products that already have implemented those algorithms to avoid implementing the algorithms that solve OR models. In most of them, it is only necessary to specify the models and data and they can produce the solution without further work. So, we can just pick one of solver from those that allows to solve ILPs. We selected CPLEX[3] from IBM, via IBM Academic Initiative[4].

The AI solvers were completely implemented by us. We did not used any external library in order to have more control over the solution generation process. Every aspects of the AI solvers were implemented using the basic data structures and built-in features of C#.

6 Evaluation

Our problem, by itself, is not very explored in the literature, so it is not easy to compare results. Another aspect that makes a comparison difficult is that each problem has its own specification.

In order to perform the tests we created several small datasets as well as datasets based on real TSDs. The small datasets consist on 3 professors and 5 courses, but with different teaching credits as well as enrolled students. This datasets were used to manually check if the solution produced by our system respected all constraints. We did not find any constraint violation. These results ensured that our model is correct that it does what is expected.

[2]https://www.mysql.com/

[3]IBM software property

[4]http://www-304.ibm.com/ibm/university/academic

We also used datasets based on real TSDs detailed in Table 1. In these datasets, some teachers had generic preferences, and others had preferences that are more specific.

All tests were run in a computer with an Intel Core i7-4702MQ cpu and 16gb of RAM. The IBM CPLEX version used was v12.6.1.

6.1 Solution quality

OR techniques (Resource Assignment - Section 3.3.1) were exhaustively tested to assess the real quality of the produced solution. We first had to ensure that the created model respect all the problem constraints, then compared the generated solution with the manually generated solution, and, for last, checked with an experienced person if the solution could be applied.

6.1.1 Comparison with real solutions

In Table 2, we have a description of which dataset and technique were used to produce each solution. We have one solution of each combination of dataset with resource schedule model as presented in Section 3.3.1. The first two columns indicate which of the datasets described in Table 1 were used in that solution. The following column indicates which objectives were used. The last two columns indicate which of the methods described in Section 3.3.1 were used.

Comparing the solution obtained by our approach with real TSD allows to determine if our solution is conceptually correct and produce satisfactory results. The comparison between the some key indicators of the solutions are shown in Table 3.

Solutions 5 and 10 represent the solutions produced by humans. In the first column of Table 3, we have the solution value based on the used preferences calculated by summing the value of all preferences that were satisfied in the solution. Having Solution 5 as the baseline for the first four solutions, we can see that we have a 10% decrease of the solution value from the manually generated solution. For the other four solutions, which have the solution 10 as baseline, we have a decrease of 5% on solution 6 but we also have an increase of about 2% on solutions 7-9.

Considering the other two measures presented in Table 3, we must consider them to assess the real quality of the solutions. According to these two measures, Solution 1 and Solution 6 are better than their corresponding baseline solutions. This means that, in total, there are fewer credits over or under the expected. Also, note that the difference between solution 6 and 10 is very significant.

	Number of professors	Number of courses	Average desired credits	Minimum desired credits	Maximum desired credits	Total desired credits
Dataset 1	17	22	10.25	4.5	16	174.25
Dataset 2	18	22	10.19	4.5	18	183.5

Table 1: Description of the dataset based on real TSDs

	Datasets		Objectives			Optimization Methods	
	Dataset 1	Dataset 2	Maximize preferences	Minimize total deviation of credits	Minimize average deviation of credits	Weighted ($\alpha_1 = 1, \alpha_2 = -10$)	Sequential
Solution 1	X		X	X		X	
Solution 2	X		X		X	X	
Solution 3	X		X	X			X
Solution 4	X		X		X		X
Solution 5	X		Manual				
Solution 6		X	X	X		X	
Solution 7		X	X		X	X	
Solution 8		X	X	X			X
Solution 9		X	X		X		X
Solution 10		X	Manual				

Table 2: Solution Details

6.1.2 Manual Validation

We asked the person who performs the task of elaborating the TSD for several years, to informally evaluate each of the 10 solutions.

The main defects of the solutions were about some mismatches between professors and courses even if it were within his scientific areas. Other comments were about having some professors assigned to just one single lecture of a course. Both the comments have the some cause. As one of the objectives is to minimize the difference between expected teaching credits and real teaching credits, the system tries to fill all gaps with other classes that do not affect the solution value.

6.2 Why some techniques fail in our problem?

Local searches showed several problems. The first problem is the initial solution because it must be valid and be the best possible. Constructing such solution is not an easy task and most of the time, our greedy algorithm could not do it.

Another problem with this technique is the amount of successors that can be generated at each step of the search. It generates one successor for each professor and for each course section of a course. For instance, for dataset 1, we have 17 professors and $22 * 2$ course sections (assuming that each course has two course sections), so it generates $17 * 22 * 2 = 748$. Only just a small percentage of the successors correspond to a valid solution.

Another technique that revealed to be inefficient is the Schedule Assignment model (Section 3.3.1). To use this model, we first need to generate a large number of schedules. This task is very complex and must be done for each professor individually. The number of schedules that were generated is unpractical as well as the time it takes to generate them. Having a huge number of schedules affects the OR models as it has to deal with each single one. This fact also affects performance.

Local searches and the Schedule Assignment model take considerable more time than the Resource Assignment model evaluated in this chapter to obtain the some solution. The Resource Assignment model retrieves a solution almost instantaneously with guaranteed optimal solution for our model.

6.3 Extensibility of the solution

Another important point of our solution, web platform and software generator of TSD, is the fact that this is generic enough to allow to be extend.

If we want to add a new type of preference to the system, we need to add a table in the database; then add the form to the web platform in order to be able for the professor the specify this new type of preference; finally, we need to add the computation of that type of preference to the internal representation of the software to automatic generate the TSD.

If we need to change the method of how the course sections is calculated, we can do it directly on the web platform as it allows to change same parameters that affect this calculation, or if it is a more extreme change, we must do it in the implementation of the software to automatic generate the TSD.

7 Conclusions

7.1 Summary

Assigning professors to courses is not an easy task and done under very different approaches in the schools that do it. We analyzed how DEI performs such task in order to conceive a system that automate it.

	Preferences Value	Total Deviation from Expected Teaching Credits	Average Deviation from Expected Teaching Credits
Solution 1	1180	23,75	1,39
Solution 2	1200	26,75	1,57
Solution 3	1200	26,75	1,57
Solution 4	1200	26,75	1,57
Solution 5	1300	24,5	0,9
Solution 6	2550	14,5	0,81
Solution 7	2720	28	1,56
Solution 8	2780	34	1,89
Solution 9	2720	28	1,55
Solution 10	2680	40,8	1,9

Table 3: Preferences and Expected Credits measures

Our literature review showed that there are two main areas that contribute to solve this problem: OR and AI. We developed models for both approaches. Two models for OR were created. One based on assigning directly courses to professor and the second based on assigning pre-generated schedules to professors. We also used two search strategies from AI: Hill Climbing and Tabu Search. Both strategies use as basic moves: changing a class from one professor to another and a swap of classes from two professors, one class from each.

From our tests done using datasets that are based on real TSD, the OR model that makes the direct association between courses and professors outperforms any other technique.

7.2 Achievements

Our work allowed to develop a solution that can be used by DEI in order to start automating the process of assigning professors to courses and class. Our system can generate good enough solutions that are valid, that can satisfy the same number of preferences of a manual solution and decrease the deviation of the difference between expected and effective credits. Nevertheless, sometimes the solutions may need some small modifications or improvements that must be done manually.

7.3 Future Work

Our work can have improvements to get better results. One of the problems that raised is how the number of courses sections is calculated. We used a basic rule to do it. However sometimes it does not get it right. Using the knowledge from previous year, as well as using data from students attendance to classes, can be very useful to have better predictions of number of classes needed.

Another interesting point to improve is the objective function of the OR models. Adding other objectives that make sense or even giving the possibility to the user to choose which of them he wants to use may help to avoid the problems pointed in the manual validation.

List of Acronyms

DEI Department of Computer Science and Engineering

IST Instituto Superior Técnico

TSD Teaching Service Distribution

ILP Integer Linear Programming

OR Operational Research

AI Artificial Intelligence

References

Al-Yakoob, S. M. and Sherali, H. D. (2006). Mathematical programming models and algorithms for a class-faculty assignment problem. *Eur. J. Oper. Res.*, 173:488–507.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-Price: Column Generation for solving huge integer programs. *Oper. Res.*, 65(3):316–329.

de Grano, M. L., Medeiros, D. J., and Eitel, D. (2009). Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Manag. Sci.*, 12:228–242.

Eiselt, H. a. and Marianov, V. (2008). Employee positioning and workload allocation. *Comput. Oper. Res.*, 35(2):513–524.

Elizondo, R., Parada, V., Pradenas, L., and Artigues, C. (2010). An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport. *J. Heuristics*, 16(4):575–591.

Gamache, M., Soumis, F., Villeneuve, D., Desrosiers, J., and Gelinas, E. (1998). The preferential bidding system at Air Canada. *Transp. Sci.*, 32(3):246–255.

Glover, F. (1989). Tabu Search - Part I. *ORSA J. Comput.*, 1(3):190–206.

Gunawan, A. and Ng, K. M. (2011). Solving the teacher assignment problem by two metaheuristics. *Int. J. Inf. Manag. Sci.*, 22:73–86.

Gunawan, A., Ng, K. M., and Ong, H. L. (2008). A genetic algorithm for the teacher assignment problem for a University in Indonesia. *Int. J. Inf. Manag. Sci.*, 19(1):1–16.

I, M. W. C. and Laporte, G. (1998). Recent Developments in Practical Course Timetabling. *Pract. Theory Autom. Timetabling II*, 1408:3–19.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science (80-.).*, 220(4598):pp. 671–680.

Lawler, E. L. and Wood, D. E. (1966). Branch-And-Bound Methods: A Survey. *Oper. Res.*, 14:699–719.

Lewis, R. (2007). A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectr.*, 30(1):167–190.

Lü, Z. and Hao, J. K. (2010). Adaptive Tabu Search for course timetabling. *Eur. J. Oper. Res.*, 200(1):235–244.

Lü, Z. and Hao, J.-K. (2012). Adaptive neighborhood search for nurse rostering. *Eur. J. Oper. Res.*, 218(3):865–876.

Maenhout, B. and Vanhoucke, M. (2010). A hybrid scatter search heuristic for personalized crew rostering in the airline industry. *Eur. J. Oper. Res.*, 206(1):155–167.

Phillips, A. E., Waterer, H., Ehrgott, M., and Ryan, D. M. (2015). Integer programming methods for large-scale practical classroom assignment problems. *Comput. Oper. Res.*, 53:42–53.

Sabar, M., Montreuil, B., and Frayret, J.-M. (2008). Competency and preference based personnel scheduling in large assembly lines. *Int. J. Comput. Integr. Manuf.*, 21(4):468–479.

Schimmelpfeng, K. and Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *OR Spectr.*, 29:783–803.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *Eur. J. Oper. Res.*, 226(3):367–385.

Whitley, D. (1994). *A Genetic Algorithm Tutorial*, volume 4. Kluwer Academic Publishers.