A WEB-BASED SYSTEM FOR STUDENT-PROJECT ALLOCATION

Jean-Pierre Gerval, Yann Le Ru

Institut Supérieur de l'Electronique et du Numérique – Brest
20 rue Cuirassé Bretagne – CS 42807 - 29228 BREST cedex 2 – FRANCE
Tel: +33 (0)2 98 03 84 00, Fax: +33 (0)2 98 03 84 10
E-mail: jean-pierre.gerval@isen.fr; yann.le-ru@isen.fr

ABSTRACT

This paper presents a Web-Based System that aims at helping both academic staff and students in the task of Student-Project Allocation (SPA). For this purpose, we use a Constraint Programming Solver and we describe the model that has been implemented. The server implements a LAMP (Linux, Apache, MySQL and PHP) architecture. This system also provides functionalities that concern all the life cycle of a student's project from the writing of subjects by the teachers in the early beginning to the evaluation of the students' work at the end of their projects. This System implements various types of technologies: Databases, Web Technologies, Object Oriented Programming, Constraint Programming...

KEY WORDS

Web-Based System, Student-Project Allocation (SPA), Constraint Programming Solver

1. Introduction

In this paper, we propose a Web-Based System that aims at helping both academic staff and students in the task of Student-Project Allocation (SPA). This system also offers functionalities that concern all the life cycle of students' project from the description of the work to be performed at the early beginning to the evaluation of students' work by the end of their projects.

Such a system is of interest for all educational organizations that are willing to give projects to students. This is the case in many universities, where projects play a great role in the students' academic programme. One of the interests in proposing an automated solution for project allocation is to avoid long discussions and conflicts between students, especially concerning the decision of knowing who will do what! A gain of time is

also another advantage for the academic staff in charge of the follow up of the students' projects.

Student-Project Allocation Problem is a two-sided matching problem [1] and could be seen as a generalization of the Stable Marriage Problem [2] or the Hospitals-Residents (HR) problem [3].

One of the most famous centralized matching systems may be the National Resident Matching Program (NRMP) in the US [4] which is a variant of the HR problem. For many years, a lot of work has been achieved in the field of SPA [2]. For example [5] points out that to automate the allocation of students to projects: "*an optimal linear-time algorithm for this setting cannot be obtained by simply reducing an instance of SPA to an instance of HR*" and the authors propose a linear-time algorithm to solve the SPA in a student-oriented approach.

In our case, we have chosen a Constraint Programming approach. Thus, we do not have to focus on how the problem is solved but, on the contrary, we have to formalize the description of the problem by means of constants, variables and constraints. According to this solution, it is easy to change the rules of our SPA. For example, adding a list of reserved projects for some students or changing the way groups of students are allocated to projects. In the next chapters, we will give an overview of our Web-Based System and details about our SPA implementation.

2. System overview

2.1 Introduction

An overview of the system (Figure 1) is presented hereafter. It points out the main components of this system: solver, server database and the different roles of the users: administrator, student and teacher.

Figure 1. System overview

Server and database implement Linux, Apache, MySQL and PHP. This architecture has come to be known simply as LAMP. The solver implements Choco [6], which is a java library for Constraint Satisfaction Problems (CSP) and Constraint Programming (CP).

2.2 Functionalities

We have three roles in our system and a five step workflow (Figure 2). Available functionalities depend on which step is currently active.

Figure 2. Five step workflow

Step one - Projects proposal by academic staff

The administrator can register teachers in the database. From this step, he can edit projects that have been saved in the database.

Teachers (i.e. Professors or lecturers) can write, edit and save their projects into the database. Teachers may submit as many projects as they want. However, they are generally assigned three to six projects according to other work charges.

Students can view past projects. They have to wait for the next step in order to discover the new projects.

Step two - Projects overview by students

From this step, the teachers are neither allowed to modify their projects nor to add any more projects.

The students can read the new proposals that were submitted by teachers.

Step three - Choice by students

As the students will work in pairs, groups of two students are expected to make three choices among the list of available projects. These three choices must be different and are intended to be equivalent. That means there is no preferred choice. This is how we have managed the students' choices until today.

The Teachers can view the students' choices.

Step four - Student Project Allocation

During this step, the teachers and the students cannot view anything.

According to previous discussions with colleagues (i.e. teachers), the administrator is allowed to reserve a project (Figure 3) for a group of students or to exclude a group of students from a project they have already chosen. The administrator can run Student-Project Allocation as many times as he wants, as long as he has not saved an instance of allocation in the database. When the result of Student-Project Allocation has been saved, the system starts in step five.

Figure 3. User Interface - Example

Step five - Work evaluation

The functionalities that are offered in this step are intended to be used at the end of the project. Three different evaluations are performed:

- The assessment by the teacher who supervised the students' work;
- The assessment of the students' report; this evaluation is achieved by another teacher than the one who was in charge of the student supervising;
- The assessment of the project presentation performed by students in front of an academic jury.

All these assessments are stored in the database so that they can be easily retrieved from one year to another.

3. Solving Student-Problem allocation

3.1 Introduction

Student-Project Allocation is solved by means of a Constant Programming Solver. To perform this task we use Choco [6] because:

- it is an open system distributed under a BSD license;
- it is a java library for Constraint Satisfaction Problems (CSP) and Constraint Programming (CP) and it provides various domain types and various algorithms for constraint propagation;
- it can either be used in satisfaction mode (computing one solution, all solutions or iterating them) or in optimization mode (maximization and minimization);
- It is user-oriented constraint and it provides a clear separation between model and solver (Figure 4).

Figure 4. Choco architecture

3.1 Model

The model is set up by means of constants, variables and their domains of values and, finally, constraints.

3.1.1 Constants

Our model constants are defined as follows:

A set of groups (G) that contains all the identifiers (g_i) of students group:

$$G = \{ g_i \mid i \in [1, n] \}$$

in which n is the total number of groups.

A set of projects (P) that contains all the identifiers (p_k) of projects that were proposed by the teachers:

$$P = \{ p_k \mid k \in [1, m] \}$$

in which m is the total number of projects and must be equal or greater than the total number of groups.

The set of choices (C) is a set of triplets where each triplet is a random permutation of the three projects that were chosen by a pair of students.

C and G must have the same cardinality (i.e. the same number of elements):

$$C = \{ (p_{i,1}, p_{i,2}, p_{i,3}) \mid i \in [1, n], (p_{i,1}, p_{i,2}, p_{i,3}) \in P^3, |C| = |G| \}$$

A set of reservations (R) that contains couples group-project when a project must be allocated to the corresponding group:

$$R = \{ (g_i, p_r) \mid i \in [1, n], r \in [1, m] \}$$

A set of exclusions (E) that contains couples group-project when a group must be excluded from the corresponding project:

$$E = \{ (g_i, p_e) \mid i \in [1, n], e \in [1, m] \}$$

3.1.2 Variables and their domains of values

Variables and their domains of values are defined as follows:

A set of integers (X) in which each element (X_i) represents the project that is allocated to a group of students:

$$X = \{ X_i \mid i \in [1, n], |X| = |G| \}$$

Each element of X must take its value in a Domain $D(X_i)$, which is the triplet that represents the three projects chosen by a pair of students:

$$D(X_i) = \{ p_{i,1}, p_{i,2}, p_{i,3} \} \text{ where } (p_{i,1}, p_{i,2}, p_{i,3}) \in C$$

3.1.3 Constraints

We have three different constraints according to allocations, reservations and exclusions. The first constraint C_{Values} states that all elements of X must be different. That means that all groups must be allocated a different project.

$$C_{\text{Values}} = \{ X_i \neq X_j \mid i, j \in [1, n], i \neq j \}$$

The other two constraints $C_{Reservations}$ and $C_{Exclusions}$ respectively implement projects reservations and exclusions:

$$C_{Reservations} = \{ X_i = p_r \mid i \in [1, n], r \in [1, m], \exists (g_i, p_r) \in R \}$$

$$C_{Exclusions} = \{ X_i \neq p_e \mid i \in [1, n], e \in [1, m], \exists (g_i, p_e) \in E \}$$

3.2 Model implementation

The model is implemented in java language using Choco Application Programming Interface (API). The code sample (Figure 5) points out Choco API requests. It shows how easy it is to implement a model when this model has been correctly defined. One of the most important advantages in using Choco is that modelling and solving are really two separate tasks where the most important is modelling.

```
public static void main(String[] args) {
    // Get all required data from database
    Request db = new Request ();
    // Constants
    G = db.getGroups();
    C = db.getChoices();
    R = db.getReservations();
    E = db.getExclusions();
    // Create an instance of Choco Model
    Model MyChocoModel = new CPModel();
    // Define variables...
    IntegerVariable[] X = new IntegerVariable[db.NumberOfGroups];
    // ... and their domains
    for (int i=0; i< db.NumberOfGroups; i++) {
        X[i] = Choco.makeIntVar("allocationForGroup"+ G[i], C[i]);
    }
    // Define constraints
    // 1) All values must be different
    MyChocoModel.addConstraint(Choco.allDifferent(X));
    // 2) Reservations
    for (int j=0; j< db.NumberOfReservations; j++) {
        // Search for group index in matrix X[]
        for (int i=0; i< db.NumberOfGroups; i++) {
            if (G[i] == R[j][0]) {
                MyChocoModel.addConstraint(Choco.eq(R[j][1], X[i]));
            }
        }
    }
    // 3) Exclusions
    for (int j=0; j< db.NumberOfExclusions; j++) {
        // Search for group index in matrix X[]
        for (int i=0; i< db.NumberOfGroups; i++) {
            if (G[i] == E[j][0]) {
                MyChocoModel.addConstraint(Choco.neq(E[j][1], X[i]));
            }
        }
    }
    // Create an instance of Choco Solver
    Solver MyChocoSolver = new CPSolver();
    // Read the model and try to solve the problem
    MyChocoSolver.read(MyChocoModel);
    MyChocoSolver.solve();
    if (MyChocoSolver.isFeasible()) {
        // GREAT! There is a solution! Results are saved in X[]
    }
    else {
        // Find another method in order to provide some results...
    }
}
```

Figure 5. Sample of java code using Choco API

Choco may not always find a solution. For example, there may be fewer projects than groups or it may occur that all the choices of a group were banned. In that case, we provide another method in order to compute a partial solution. We know that this solution will not be complete. That is to say that some pairs of students will not be allocated any project.

Firstly, we store choices in an $n*m$ matrix where lines represent students pair and columns projects. If a pair of students has chosen a project then the value in the matrix is set to one, zero everywhere else.

Secondly, we order this matrix (Figure 6) so that the sum of columns ranges from the least wanted project to the most wanted one. This way we are sure to allocate to the right pair of students projects that have been chosen only one time.

Figure 6. Ordering choices

Thirdly, we proceed to the allocation according to the following algorithm:

```
For each group (i.e. line)
    If there is a choice
        Allocate this project to the current group
        Delete this project for all other groups
        (i.e. Sum of the current column = zero)
    Else
        This group does not have any project allocated
    End if
End for
```

In order to improve the quality of this method, we repeat $n*m$ times these three steps with a random permutation of lines. The number of repeats has been chosen empirically. Finally, we choose as an acceptable solution the result that gives the greatest number of allocations.

At the end of this process, we provide three lists:

1. a list of groups with an allocated project;
2. a list of groups without any allocation;
3. a list of projects that have not been allocated.

If there are some groups without any project then these groups are given some time, generally a week, so that they can discuss together and with teachers in order to make their choice among unallocated projects.

4. Conclusions and future work

This Web-Based System has been successfully experimented this year with our students. Fifty-five projects were proposed and forty-two pairs of students had to make a choice. We didn't have any feedback from unsatisfied students because the rule was clear: each choice is of equal importance.

The approach based on Constraint Programming that has been described in this paper, enables to set up easily new rules, new models for the Student-Project Allocation Problem.

We are thinking about a new model for next year. For the time being, the project allocation is performed for a pair of students. Most educators know that, sometimes, when students decide to work together it is not a matter of chance. One of them may be better than the other one thus it is a good way to increase marks without too many efforts.

In our new model, a student will still have to choose some projects he wants to work on. However, he will also have to set priorities on his choices and to select the colleagues he wants to work with. Therefore, the challenge would be to generate the pairs of students and to allocate projects to these pairs.

Acknowledgements

The authors would like to thank all colleagues and students who contributed to this study. We are grateful to Dr Jean-Yves Mulot, academic dean at ISEN-Brest, for his collaboration and helpful discussions during preliminary investigations.

We would also like to thank Ms Dominique Maratray, English teacher at ISEN-Brest, for her helpful cooperation.

References

- [1] A.E. Roth, M.A.O. Sotomayor, Two-sided matching: a study in game-theoretic modelling and analysis, volume 18 of Econometric Society Monographs, Cambridge University Press, 1990.
- [2] Kazuo Iwama, Shuichi Miyazaki, A Survey of the Stable Marriage Problem and Its Variants, International Conference on Informatics Education and Research for Knowledge-Circulating Society, Kyoto, Japan, 2008
- [3] D. Gale and L.S. Shapley, College admissions and the stability of marriage, American Mathematical Monthly, 69:9–15, 1962.
- [4] National Resident Matching Program, <http://www.nrmp.org/> (Retrieved on January 2012)
- [5] David J. Abraham, Robert W. Irving, and David F. Manlove, The student-project allocation problem. In, Proceedings of ISAAC 2003: the 14th Annual International Symposium on Algorithms and Computation, 15-17 December, 2003, Lecture Notes in Computer Science, Vol 2906, pages 474-484, Kyoto, Japan.
- [6] Choco: An open source Java CP library <http://www.emn.fr/z-info/choco-solver/> (Retrieved on January 2012)