From a MILP Model to Solve the Weighted MAX-CSP to a MILP Model to Generate University Timetables with Teacher and Student Preferences

J. Barahona da Fonseca
Department of Electrical Engineering and Computer Science
Faculty of Sciences and Technology
New University of Lisbon
jbfo@fct.unl.pt

Abstract

The maximal constraint satisfaction problem (MAX-CSP) is an over constrained optimization problem where we want to maximize the number of constraints satisfied or equivalently minimize the number of unsatisfied constraints and where all constraints can be unsatisfied or relaxed. In the weighted MAX-CSP we also associate a weight to each constraint and we minimize the sum of weights associated to unsatisfied or relaxed constraints. We present a general MILP model to solve this class of problems and then apply it to an over constrained problem constituted by four linear equations of two variables. We showed that the MILP model obtains the solution of the system of the two equations to which we attributed a higher weight and so they were not relaxed or removed from the MILP model. To our knowledge our work is the first proposal of solution of the weighted MAX-CSP using a MILP model and the Cplex solver. Previous proposals were based in dedicated solvers and logic programming. Then we adapt this MILP model to solve university timetabling problems that are a variant of the weighted MAX-CSP where there are a set of constraints that cannot be unsatisfied or relaxed, the *hard constraints*. The remaining constraints that can be unsatisfied or relaxed are called *soft constraints*. To each *soft constraint set* is associated a weight that reflect the different priorities to satisfy them. We associate to each set of *soft constraints* an indexed binary variable that will control the relaxation of *soft constraints*. Finally we apply this MILP model to a small instance of the university timetabling problem with four different preferences of teachers and students. Due to the big runtimes, in the near future we plan to implement our MILP Model with the Prolog language.

Keywords: Artificial Intelligence, Operations Research, Optimization Problems, Mathematical Programming, Linear Programming, MILP Model to Generate University Timetables.

1. Introduction

It seems that the first attempt of solving the university timetabling problem was in [1], but this solution did not use constraints relaxation. It seems that constraint relaxation was introduced in [2] but without the use of indexed binary variables associated to *Soft Constraints*. In this work we introduce *Soft Constraints* that can be relaxed with the help of a *indexed binary relaxation variable*, each element associated to each relaxed constraint, with the techniques developed in our previous work [3]. Finally we developed a MILP model and solve it with the Cplex solver with four different values of student and teacher preferences.

2. A MILP Model to solve the weighted MAX-CSP Problem

Imagine we have three sets of constraints, two of them inequality constraints and the third constituted by equality constraints and we want to maximize the weighted sumo of satisfied constraints or equivalently minimize the weighted sum of unsatisfied constraints. To *control* the relaxation of each set of constraints we use three indexed binary variables, *Relax_Set*1(*i*), *Relax_Set*2(*j*) and *Relax_Set*3(*k*).
When any of these binary variables assumes the value 1, the correspondent constraint is always true for any variable assignment, which is *equivalent* to remove the constraint from the mathematical program. So, minimizing the total number of 1s over the three indexed binary variables is *equivalent* to *maximize* the number of satisfied constraints, being the objective variable defined by (1).

$$obj = \sum_{i=1}^{N_1} Relax\_Set1(i) + \sum_{j=1}^{N_2} Relax\_Set2(j) + \sum_{k=1}^{N_3} Relax\_Set3(k) \quad (1)$$

This is the solution of the MAX-CSP problem. To solve the weighted MAX-CSP problem we just multiply each sum by a positive number, and the distribution of these weights will define a hierarchy of constraints using an objective variable defined by (2).

$$obj = W_1 \sum_{i=1}^{N_1} Relax\_Set1(i) + W_2 \sum_{j=1}^{N_2} Relax\_Set2(j) + W_3 \sum_{k=1}^{N_3} Relax\_Set3(k) \quad (2)$$

Next we show how to modify the inequality and equality constraints to allow them to be relaxed by the respective indexed binary variables, a technique developed in a previous work [3]. In (3) and (4) we show the general form of inequality constraints that are always true for any variables instantiation when the relaxation binary variables assume the value 1, which is *equivalent* to remove them from the mathematical program.

$$a_{11}x_1 + ... + a_{n_1,n_1}x_{n_1} + k_1 Relax\_Set1('1') \geq b_1 \quad (3)$$

$$a_{11}x_1 + ... + a_{n_2,n_2}x_{n_2} + k_2 Relax\_Set2('1') \leq b_2 \quad (4)$$

In (3) and (4) the parameters k_1 and k_2 are fixed such that they guarantee that the respective inequalities are always true for all the search space when the respective indexed binary variables assume the value 1, which is *equivalent* to remove the inequalities from the mathematical program.

Having into account that ($A \geq B$) AND ($A \leq B$) $\equiv$ A=*B*, and that the solution of a mathematical program can be seen as an instantiation of model variables that satisfies the logical AND of its constraints, the relaxation of an *equality* constraint can be implemented by two inequality constraints relaxed by the same indexed binary variable, as we show in (5) and (6).

$$a_{11}x_1 + ... + a_{n_3,n_3}x_{n_3} + k_3 Relax\_Set3('1') \geq b_3 \quad (5)$$

$$a_{11}x_1 + ... + a_{n_3,n_3}x_{n_3} \leq b_3 + k_3 Relax\_Set3('1') \quad (6)$$

When *Relax_Set*3('1')=0, the *combination* of (5) and (6) is *equivalent* to (7).

$$a_{11}x_1 + ... + a_{n_3,n_3}x_{n_3} = b_3 \quad (7)$$

When *Relax_Set*3('1')=1, (5) and (6) are always true for all possible variable instantiations, which is equivalent to remove (5) and (6) from the mathematical program and the constant k_3 is adjusted to guarantee that this happens for all possible variable assignments.

3. Illustrative Example of a System of 2 Equations with more 2 Incompatible Equations that must Be Relaxed

To illustrate this technique of *Automatic Constraint Relaxation* described in previous section we will describe a very simple and small MILP model with 2 compatible linear equations and 2 incompatible linear equations that must be relaxed for the solver obtain the desired solution. Since any two linear equations of two variables have solution we must relax two equations from the mathematical program. We will do this using a binary indexed variable and attributing a higher weight to the equations we want to solve.

To define each equation we used two inequality constraints where appear the same relaxation binary variable, in the right side in the *less or equal* (represented by =l=) *constraint* and in the left side in *greater or equal* (represented by =g=) *constraint*. The system of 2 equations is described by (8)

$$\begin{cases} 3x_1 - 2x_2 = 8 \\ 4x_1 - 3x_2 = 10 \end{cases} \quad (8)$$

which solution is x_1=4, x_2=2, that in the MILP model are represented by the following 4 inequalities

$$\forall x_1, x_2 : 3x_1 - 2x_2 \le 8 + 10000 Relax\_eq('eq1') \quad (9)$$

$$\forall x_1, x_2 : 3x_1 - 2x_2 + 10000 Relax\_eq('eq1') \ge 8 \quad (10)$$

$$\forall x_1, x_2 : 4x_1 - 3x_2 \le 10 + 10000 Relax\_eq('eq2') \quad (11)$$

$$\forall x_1, x_2 : 4x_1 - 3x_2 + 10000 Relax\_eq('eq2') \ge 10 \quad (12)$$

which are implemented by the following lines of GAMS code:

```
eq1.. 3*x1-2*x2=l=8+10000*Relax_eq('eq1');
eq11.. 10000*Relax_eq('eq1')+3*x1-2*x2=g=8;

eq2.. 4*x1-3*x2=l=10+10000*Relax_eq('eq2');
eq21.. 10000*Relax_eq('eq2')+4*x1-3*x2=g=10;
```

The other two equations that are not compatible with (1) neither with each other are defined by the following four inequalities

$$\forall x_1, x_2 : 7x_1 - 8x_2 \le 1112 + 10000 Relax\_eq('eq3') \quad (13)$$

$$\forall x_1, x_2 : 7x_1 - 8x_2 + 10000 Relax\_eq('eq3') \ge 1112 \quad (14)$$

$$\forall x_1, x_2 : 8x_1 - 7x_2 \le 1555 + 10000 Relax\_eq('eq4') \quad (15)$$

$$\forall x_1, x_2 : 8x_1 - 7x_2 + 10000 Relax\_eq('eq4') \ge 1555 \quad (16)$$

which are implemented by the following lines of GAMS code:

```
eq3.. 7*x1-8*x2=l=1112+10000*Relax_eq('eq3');
eq31.. 10000*Relax_eq('eq3')+7*x1-8*x2=g=1112;

eq4.. 8*x1-7*x2=l=1555+10000*Relax_eq('eq4');
eq41.. 10000*Relax_eq('eq4')+8*x1-7*x2=g=1555;
```

The objective variable will be the weighted sum of the binary variable *Relax_eq*. To guarantee that the third and fourth equations are relaxed we attributed them a lower weight $w(i)$:

$$obj = \sum_i w(i)\, Relax\_eq(i) \qquad (17)$$

this is implemented by the following line of GAMS code:

```
calc_obj.. obj=e=sum(i, w(i)*Relax_eq(i));
```

And then we minimize the objective variable which prevents the trivial solution of all values of *Relax_eq* being 1 and x_1 and x_2 being any possible value:

```
Model Sist2Eqs /all/;
Solve Sist2Eqs using MILP minimizing obj;
```

When the Cplex solver minimizes *obj* it finds that the optimal solution corresponds to *Relax_eq*('eq1')=*Relax_eq*('eq2')=0 since the correspondent weights are higher and sets *Relax_eq*('eq3')=*Relax_eq*('eq4')=1, removing the correspondent equations and obtaining the solution x_1, x_2 of the system of equations constituted by the first and second equations.

Next we show the output of the run of this MILP mode that took few seconds.

```
----     61 VARIABLE x1.L                =        4.000
           VARIABLE x2.L                =        2.000

----     61 VARIABLE Relax_Eq.L
eq3  1.000,    eq4 1 .000

----     61 VARIABLE obj.L               =       10.000
```

If by the contrary we attribute to equations 3 and 4 a greater weight we got another solution:

```
----     41 VARIABLE x1.L                =        0.256
           VARIABLE x2.L                =        0.231

----     41 VARIABLE Relax_eq.L

eq1 1.000,    eq2 1.000
```

This is the solution of the system of equations constituted by the third and fourth equations. In appendix 1 we show the complete GAMS code and all possible solutions correspondent to all possible pairs of equations associated to different weight distributions.

4. Solution of University Timetabling using Automatic Relaxation

In university timetabling there are *hard* constraints, constraints that must be satisfied and cannot be relaxed, and a hierarchy of *soft* constraints that can be unsatisfied and we want to maximize the number of satisfied soft constraints. In the implementation of the MILP model to solve the Timetabling problem with *soft constraints* we used the same philosophy of creating a binary variable for each set of *soft constraints* developed in our previous work [3]. Then we create an auxiliary variable with the number of 1's of the relaxation binary variable. Maximizing the number of satisfied soft constraints is equivalent to minimize the number of 1's of the respective binary relaxation variable, which we solve minimizing the sum of the correspondent binary indexed relaxation variable.
To define a hierarchy of soft constraints we multiply each of these auxiliary variables by a weight. The sets of soft constraints with higher weights will have higher priority than soft constraints with lower weights and the Cplex solver will try first to satisfy the soft constraints with higher weights.

The definition of the *Timetable* is implemented by only one binary variable, *ocupation*(*s*,*d*,*h*,*c*,*t*,*sg*), with indexes are the discipline *c*, the room *s*, the day *d,* the hour *h*, the *student group sg* and the teacher *t*. Next we show 2 *hard* restrictions that cannot be relaxed *and* 2 *soft* constraints that may be relaxed and the definition of the objective variable *obj* that will be minimized. Note that each *aux* variable is the number of relaxed soft constraints and the constants that multiply them define a hierarchy of the different sets of soft constraints.

** HARD CONSTR- in one room, s, d and h it can be only 1 discipline*
hard_constr_ocup_s(s,d,h).. **sum**((c,t,sg), ocupation(s,d,h,c,t,sg))**=l=**1 ;

** 1 student Cannot Have More than 1 Class at the Same Time*
constr1_student(sg,d,h).. **sum**((s,c,t), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c))**=l=**1;

** Soft Constraint: IF POSSIBLE Prevent Classes in the End of the Day, d*
constr2_student(d)..
sum((s,t,h,c,sg),
ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c)*(ord(h)=(h_max)))**=l=Relax_constr2_st(d**)*100;

* Soft Constraint: *IF POSSIBLE 1 Teacher Must Not Give Consecutive Classes in 2 different days d and d1*; Note that Teacher_hours(t,c) is a Parameter
constr7_cad(c,s,s1,d,d1,h,h1,t,sg,sg1)$(ord(d)>ord(d1))..
50*Relax_dist_c(t,c,sg,sg1)+ocupation(s,d,h,c,t,sg)*ord(d)-
ocupation(s1,d1,h1,c,t,sg1)*ord(d1)+ 50*(1-ocupation(s,d,h,c,t,sg)) +
50*(1-ocupation(s1,d1,h1,c,t,sg1))+ 50*(Teacher_hours(t,c)=0) **=g=2** ;

calc_obj.. obj**=e=**Flag_eq_h*100*aux_eq + **sum**((c,sg),(n(c,sg)-Nhours_c(c)))+5 * **sum**((t,c,p,sg),Categ(t)*Weights_pref(p)*Relax_pref(t,c,p,sg)) + 5*aux3 + 5 * aux4 + 50*aux5 + 1400*aux6 + 1350*aux61+ 1800* aux62 + 50*aux7 + 150*aux8 + 5000*aux9 + aux10 + aux11 + 1200*aux12 + 500*aux13 + 5* aux2_sg;

In appendix 2 we show the obtained solutions with our MILP model for a very simple case with 5 disciplines, c1..c5, 4 teachers, t1..t4, 3 courses, sg1..sg3, 8 hours of classes, h1..h8 and 2 rooms, s1-s2, with 4 different combination of *relevance coefficients*. The runtimes were 1.7 days in average and a maximum of 2.5 days for a margin less than 5% from the optimal solution with a Pentium IV with a clock of 3.6GHz and the Cplex algorithm. In appendix 3 we show the GAMS code that implements our MILP model to solve university timetables. Due to the big runtimes, in the near future we plan to implement our MILP Model with the Prolog language.

References

[1] S. Daskalaki, T. Birbas and E. Housos (2004), "An integer programming formulation for a case study in university timetabling", *European Journal of Operational Research* vol. 153, pp. 117–135.

[2] M. Guignard(1989), "On solving structured integer programming problems with Lagrangian relaxation and/or decomposition", in *Proceedings of the 28th IEEE Conference on Decision and Control,* 13-15 Dec, 1989, vol.2, pp. 1136 – 1141.

[3] J. Barahona da Fonseca(2009), "Solving any Nonlinear Problem with a Linear MILP Model", in *Proceedings of Escape-19*, pp. 647-652, Bucharest, Romania.

Appendix 1. Solution of a System of Two Equations and Two Variables with Four Equations Relaxing Two of them

```
sets i /eq1*eq4/;

binary variable Relax_eq(i);

variable x1, x2;

variable obj;

parameter w(i);

w('eq3')=1000;
w('eq4')=1000;

w('eq1')=1;
w('eq2')=1;


equations eq1, eq11, eq2, eq21, eq3, eq31, eq4, eq41, calc_obj;


eq1.. 3*x1-2*x2=l=8+10000*Relax_eq('eq1');
eq11.. 10000*Relax_eq('eq1')+3*x1-2*x2=g=8;

eq2.. 4*x1-3*x2=l=10+10000*Relax_eq('eq2');
eq21.. 10000*Relax_eq('eq2')+4*x1-3*x2=g=10;


eq3.. 15*x1-8*x2=l=2+10000*Relax_eq('eq3');
eq31.. 10000*Relax_eq('eq3')+15*x1-8*x2=g=2;

eq4.. 18*x1-7*x2=l=3+10000*Relax_eq('eq4');
eq41.. 10000*Relax_eq('eq4')+18*x1-7*x2=g=3;


calc_obj.. obj=e=sum(i, w(i)*Relax_eq(i));

Model SistEqs /all/;

Solve SistEqs using MIP minimizing obj;

display x1.l, x2.l, Relax_eq.l, w;
```

Solutions of this MILP model with various weight distributions $w(i)$:

```
GAMS Rev 233  WIN-VIS 23.3.3 x86/MS Windows          06/18/13 09:28:27 Page 6
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
E x e c u t i o n


----     41 VARIABLE x1.L                =        4.000
           VARIABLE x2.L                =        2.000

----     41 VARIABLE Relax_eq.L
```

eq3 1.000,   eq4 1.000
* This means that equations 3 and 4 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 1 and 2

---- 41 PARAMETER w

eq1 1000.000,   eq2 1000.000,   eq3   1.000,   eq4   1.000

---- 41 VARIABLE x1.L   =   0.256
     VARIABLE x2.L   =   0.231

---- 41 VARIABLE Relax_eq.L

eq1 1.000,   eq2 1.000
* This means that equations 1 and 2 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 3 and 4

---- 41 PARAMETER w

eq1   1.000,   eq2   1.000,   eq3 1000.000,   eq4 1000.000

---- 41 VARIABLE x1.L   =   -10.000
     VARIABLE x2.L   =   -19.000

---- 41 VARIABLE Relax_eq.L

eq2 1.000,   eq4 1.000
* This means that equations 2 and 4 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 1 and 3

---- 41 PARAMETER w

eq1 1000.000,   eq2   1.000,   eq3 1000.000,   eq4   1.000

---- 41 VARIABLE x1.L   =   -3.333
     VARIABLE x2.L   =   -9.000

---- 41 VARIABLE Relax_eq.L

eq2 1.000,   eq3 1.000
* This means that equations 2 and 3 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 1 and 4

---- 41 PARAMETER w

eq1 1000.000, eq2 1.000, eq3 1.000, eq4 1000.000

GAMS Rev 233 WIN-VIS 23.3.3 x86/MS Windows 06/18/13 09:35:39 Page 6
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
E x e c u t i o n

---- 41 VARIABLE x1.L = -5.692
VARIABLE x2.L = -10.923

---- 41 VARIABLE Relax_eq.L

eq1 1.000, eq4 1.000
* This means that equations 1 and 4 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 2 and 3

---- 41 PARAMETER w

eq1 1.000, eq2 1000.000, eq3 1000.000, eq4 1.000

GAMS Rev 233 WIN-VIS 23.3.3 x86/MS Windows 06/18/13 09:36:49 Page 6
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
E x e c u t i o n

---- 41 VARIABLE x1.L = -2.346
VARIABLE x2.L = -6.462

---- 41 VARIABLE Relax_eq.L

eq1 1.000, eq3 1.000
* This means that equations 1 and 3 are relaxed, so the solution *x*1, *x*2 results from the conjunction of equations 2 and 4

---- 41 PARAMETER w

eq1 1.000, eq2 1000.000, eq3 1.000, eq4 1000.000

Appendix 2. Four University Timetables obtained with our MILP Model with Different Relaxation Weights

	d1	d2	d3	d4	d5
h1	(c5,sg1,t1,r1)-(c5,sg2,t3,r2)	(c2,sg3,t2,r1)-(c4,sg2,t4,r2)	(c4,sg1,t3,r1)-(c4,sg3,t4,r2)	(c4,sg1,t3,r1)-(c2,sg3,t2,r2)	(c2,sg1,t1,r1)-(c2,sg2,t2,r2)
h2	(c2,sg3,t2,r1)	(c3,sg1,t3,r1)-	(c3,sg2,t4,r2)	(c2,sg2,t2.r1)-	(c1,sg3,t2,r1)-
h3	(c3,sg1,t3,r1)-(c3,sg2,t4,r2)	(c5,sg3,t4,r1)-(c1,sg2,t1,r2)	(c3,sg3,t3,r1)-(c2,sg1,t1,r2)	(c5,sg1,t1,r1)-(c3,sg3,t3,r2)	(c1,sg1,t1,r1)-
h4	(c1,sg3,t2,r1)	(c2,sg1,t1,r1)	(c4,sg2,t4,r1)-	(c1,sg2,t1,r1)	(c1,sg2,t1,r1)-
h5	(c1,sg1,t1,r1)	(c2,sg2,t2,r1)-(c3,sg3,t3,r2)	(c1,sg3,t2,r1)-(c3,sg1,t3,r2)	(c1,sg1,t1,r1)-(c4,sg3,t4,r2)	(c5,sg3,t4,r1)-
h6	`	(c4,sg1,t3,r1)-	(c5,sg2,t3,r2)	(c4,sg2,t4,r2)	(c3,sg2,t4,r1)-

	d1	d2	d3	d4	d5
h1	(c3,sg2,t4,r1)-(c5,sg1,t1,r2)	(c2,sg2,t2,r1)-(c2,sg1,t1,r2)	(c3,sg3,t3,r1)-(c2,sg2,t2,r2)	(c2,sg3,t2,r1)	(c4,sg2,t4,r1)-(c5,sg1,t1,r2)
h2	(c4,sg3,t4,r2)	(c4,sg3,t4,r1)	(c4,sg1,t3,r1)	(c2,sg1,t1,r1)-(c2,sg2,t2,r2)	(c5,sg3,t4,r1)
h3	(c4,sg1,t3,r1)	(c1,sg2,t1,r1)-(c3,sg1,t3,r2)	(c2,sg3,t2,r1)-(c1,sg2,t1,r2)	(c1,sg3,t2,r1)	(c1,sg2,t1,r1)-(c4,sg1,t3,r2)
h4	(c1,sg3,t2,r1)-(c5,sg2,t3,r2)	(c5,sg3,t4,r2)	(c1,sg1,t1,r1)	(c3,sg1,t3,r1)-(c3,sg2,t4,r2)	(c1,sg3,t2,r1)
h5	(c3,sg1,t3,r1)	(c1,sg1,t1,r1)-(c4,sg2,t4,r2)	(c4,sg3,t4,r1)-(c5,sg2,t3,r2)	(c3,sg3,t3,r1)	(c3,sg2,t4,r1)
h6	(c2,sg3,t2,r1)	(c3,sg3,t3,r1)	(c2,sg1,t1,r1)	(c4,sg2,t4,r1)	(c1,sg1,t1,r1)

h1	(c1,sg2,t1,r1)-	(c4,sg3,t4,r1)-	(c1,sg1,t1,r1)-(c2,sg3,t2,r2)	(c5,sg1,t1,r1)-(c5,sg3,t4,r2)	(c2,sg1,t1,r1)-(c5,sg2,t3,r2)
h2	(c3,sg3,t3,r1)-(c1,sg1,t1,r2)	(c4,sg1,t3,r1)-(c3,sg2,t4,r2)	(c4,sg2,t4,r1)-	(c1,sg2,t1,r1)-	(c4,sg3,t4,r1)-
h3	(c4,sg2,t4,r1)-	(c5,sg3,t4,r1)-	(c1,sg3,t2,r1)-(c3,sg1,t3,r2)	(c1,sg3,t2,r1)-(c4,sg1,t3,r2)	(c3,sg1,t3,r1)-(c1,sg2,t1,r2)
h4	(c4,sg1,t3,r1)-	(c4,sg2,t4,r1)-	(c2,sg2,t2,r1)-	(c2,sg2,t2,r1)-	(c1,sg3,t2,r1)-
h5	(c2,sg3,t2,r1)-	(c2,sg1,t1,r1)-(c2,sg3,t2,r2)	(c3,sg3,t3,r1)-(c5,sg1,t1,r2)	(c4,sg3,t4,r1)-	(c3,sg2,t4,r1)-
h6	(c3,sg1,t3,r1)-(c2,sg2,t2,r2)	(c5,sg2,t3,r1)-	(c3,sg2,t4,r1)-	(c2,sg1,t1,r1)-	(c3,sg3,t3,r1)-(c1,sg1,t1,r2)

	d1	d2	d3	d4	d5
h1	(c3,sg1,t3,r1)-(c1,sg3,t2,r2)	(c2,sg2,t2,r1)-(c3,sg1,t3,r2)	(c4,sg1,t3,r1)-(c1,sg2,t1,r2)	(c2,sg1,t1,r1)-(c5,sg2,t3,r2)	(c5,sg1,t1,r1)-(c1,sg3,t2,r2)
h2	(c2,sg2,t2,r1)-	(c3,sg3,t3,r1)-	(c3,sg3,t3,r1)-	(c5,sg3,t4,r1)-	(c1,sg2,t1,r1)-
h3	(c2,sg1,t1,r1)-(c4,sg3,t4,r2)	(c1,sg2,t1,r1)-	(c1,sg1,t1,r1)-(c3,sg2,t4,r2)	(c3,sg2,t4,r1)-	(c1,sg1,t1,r1)-(c4,sg3,t4,r2)
h4		(c4,sg1,t3,r1)-	(c4,sg3,t4,r1)-	(c3,sg3,t3,r1)-	(c3,sg2,t4,r1)-
h5	(c4,sg2,t4,r1)-	(c5,sg3,t4,r1)-	(c2,sg1,t1,r2)	(c3,sg1,t3,r1)-(c2,sg2,t2,r2)	(c4,sg1,t3,r1)-(c2,sg3,t2,r2)
h6	(c2,sg3,t2,r1)-(c5,sg1,t1,r2)	(c5,sg2,t3,r1)-(c1,sg1,t1,r2)	(c2,sg3, t2,r1)-(c4,sg2,t4,r2)	(c1,sg3,t2,r1)-	(c4,sg2,t4,r1)-

Appendix 3. GAMS code of our MILP model to solve University Timetables

* In this exemple we consider 3 student groups, 2 rooms, 5 days per week, 6 hours per day, 5 disciplines, 4 teachers and 2 teacher and Student Group preferences

Sets
s available rooms disponible /s1*s2/
d week days /d1*d5/
h hours between 8am-3pm /h1*h6/
c discipline /c1*c5/
t teacher /t1*t4/
p teacher and Student Group preferences /p1*p2/
sg student group /sg1*sg3/;

alias(d1,d);
alias(h1,h);
alias(s1,s);
alias(t1,t);
alias(c1,c);
alias(sg1,sg);

* In this version we give more relevance to Teachers' Preferences
* And each Discipline, *c*, has *Nhours_c(c)* Hours per week for each Student Group that selected it

Parameter
t_c_sg(t,c,sg),
* Is 1 if teacher *t* gives classes of discipline *c* to the student group *sg*
Discipline_sg(sg,c),
* Is 1 if the Student group, *sg*, have classes of discipline, *c*
Weights_prod(t),
Weights_pref(p),
Categ(t),
* Defines category of teacher *t*:1-Assistant Prof, 2-Associate Prof, 3-Full Prof
Pref_d1_sg(sg,c,p),
Pref_d1(t,c,p),

```
Pref_d2_sg(sg,c,p),
Pref_d2(t,c,p),
Pref_h1_sg(sg,c,p),
Pref_h1(t,c,p),
Pref_h2_sg(sg,c,p),
Pref_h2(t,c,p),
Teacher_hours(t,c),
Discipline_2(c),
Nhours_c(c)
 *  Defines the number of hours of classes associated to discipline c             ;

* Parameter values definition:
Discipline_sg(sg,c)=0;
Discipline_sg('sg1','c1')=1;
Discipline_sg('sg1','c2')=1;
Discipline_sg('sg1','c3')=1;
Discipline_sg('sg1','c4')=1;
Discipline_sg('sg1','c5')=1;
* snip!: some instructions ommitted




      N_h_max  Maximum Number of Consecutive Hours of Classes that a Student Group Must
Have in a Day     /4/
      N_h_max_d  Maximum Number of Hours of Classes that a Student Group CAN Have in a
single Day        /4/
      N_classes_max_prof Maximum Number of Hours  of Classes that a Teacher can Give in a
single Day d /6/
      Flag_eq_h  When 1 Searches the equilibrium of classes hours through the week /1/
      h_max  Maximum Number of Hours of Classes in a Single Day        /6/
      N_classes_sg Numero de Aulas Minimo   q 1 Aluno DEVE Ter Num dia /2/
      N_classes_max_sg Numero de Aulas Maximo q 1 Aluno PODE Ter /5/;


* Fundamental Binary variable that defines the timetabling:
Binary Variable  ocupation(s,d,h,c,t,sg);

Binary Variable ocupation_prod_h(sg,s,s1,d,c,c1,t,t1);

* Definition of Relaxation Binary variables:
Binary Variable Relax_constr_n_h(sg,d), n_h_bin(sg,d,h);
Binary Variable Relax_constr9_cad(s,s1,d,h,h1,c,c1,t);
Binary Variable Relax_constr2_st(d),  Relax_constr3_st(sg,d);
Binary Variable Relax_constr4_st(sg,d), Relax_pref(t,c,p,sg), Relax_pref_sg(sg,c,p);
Binary Variable Relax_dist_c(t,c,sg,sg1);


*****************************************************
*  CONSTRAINTS
*****************************************************


* Hard Constraint: A Student Group, sg, CANNOT have MORE THAN N_classes_max_sg PER DAY,
d
constr_n_classes_max(sg,d).. n_classes(sg,d)=l=N_classes_max_sg;
```

* Hard Constraint: A Student Group, sg, can have a Maximum of 1 Class at the same time, *h*
constr_fund_ocup2(sg,d,h).. sum ((s,c,t), ocupation(s,d,h,c,t,sg))=l=1;

* Hard Constraint: A Teacher, *t*, can give a Maximum of 1 Class of Discipline *c* at the same time *t*
constr_fund_ocup3(t,d,h).. sum ((s,c,sg), ocupation(s,d,h,c,t,sg))=l=1;

constr_t_c_sg(c,t,sg).. sum((s,d,h), ocupation(s,d,h,c,t,sg)*t_c_sg(t,c,sg))=e=Nhours_c(c)*t_c_sg(t,c,sg);

* Try to distribute the classes with equilibrium by the days

calc_n_t_h.. n_t_h=e=sum((s,d,h,c,t,sg), ocupation(s,d,h,c,t,sg));

calc_n_h_d(d).. n_h_d(d)=e=sum((s,h,c,t,sg), ocupation(s,d,h,c,t,sg));

calc_aux_eq(d).. n_h_d(d)=e=aux_eq(d)+n_t_h/card(d);

constr_aux_eq.. aux_eq_sum=e=sum(d, aux_eq(d));

* Soft Constraint: A student group must not have only one class in a day
calc_n_h_st(sg,d).. n_h(sg,d)=e=sum((s,h,c,t), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c)) ;

* The following two constraints 'translate' the variable n_h(sg,d), the number of hours of classes that a Student Group, *sg*, have in a day, *d*, in the índex *h* of the binary variable n_h_bin(sg,d,h)
calc_n_h_bin(sg,d).. sum(h, n_h_bin(sg,d,h)*(ord(h)-1)) =e=n_h(sg,d);
calc_n_h_bin2(sg,d).. sum(h, n_h_bin(sg,d,h))=l=1;

* Soft Constraint: If possible, a Student Group Must have classes all days; this constraint can be relaxed by the binary variable Relax_constr_n_h(sg,d), multiplied by 10 to give a small relevance to this objective

constr_n_h(sg,d).. 10*Relax_constr_n_h(sg,d) + sum(h $(ord(h) < 7), n_h_bin(sg,d,h)*((ord(h)-1)=0)) + sum(h $(ord(h) < 7), n_h_bin(sg,d,h)*((ord(h)-1) ge 3))=g=1 ;

*Hard Constraint: each student group, *sg*, must have all classes of the associated disciplines
constr_fund_ocup(s,d,h,c,t,sg).. ocupation(s,d,h,c,t,sg)=l=Discipline_sg(sg,c);

* HARD CONSTRAINT- in a room *s*, day *d*, and hour *h*, it can be associated only one discipline *c*, teacher *t* and student group *sg*
hard_constr_ocup_s(s,d,h).. sum((c,t,sg), ocupation(s,d,h,c,t,sg))=l=1 ;

* Hard Constraint: A student group, *sg*, cannot have more than a class *c* at the same time *h*
constr1_student(sg,d,h).. sum((s,c,t), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c))=l=1;

* Soft Constraint: If possible prevent classes at the end of the day, ord(h)=h_max
constr2_student(d).. sum((s,t,h,c,sg), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c)*(ord(h) = (h_max)))=l=Relax_constr2_st(d)*1000;

* Soft Constraint: IF Possible a student group must have less than N_h_max consecutive classes
constr3_student(sg,d).. sum((s,s1,t,t1,c,c1)$(Discipline_sg(sg,c)*Discipline_sg(sg,c1)), ocupation_prod_h(sg,s,s1,d,c,c1,t,t1))=l=N_h_max+Relax_constr3_st(sg,d)*1000;

* Soft Constraint: A Student Group MUST have less than N_h_max_d hours of classes per day
constr4_student(sg,d).. n_h(sg,d)=l=N_h_max_d + 1000 * Relax_constr4_st(sg,d) ;

calc_ocupation_prod(s,s1,d,h,h1,c,c1,t)$((ord(h) > ord(h1)))..
2*ocupation_prod(s,s1,d,h,h1,c,c1,t)=g=sum(sg,ocupation(s,d,h,c,t,sg))-50*(1-sum(sg,ocupation(s,d,h,c,t,sg)))+sum(sg,ocupation(s1,d,h1,c1,t,sg))-50*(1-sum(sg,ocupation(s1,d,h1,c1,t,sg)));
calc_ocupation_prod2(s,s1,d,h,h1,c,c1,t)$((ord(h) > ord(h1)))..
2*ocupation_prod(s,s1,d,h,h1,c,c1,t)=l=sum(sg,ocupation(s,d,h,c,t,sg)+ocupation(s1,d,h1,c1,t,sg));

calc_ocupation_prod_h(sg,s,s1,d,h,h1,c,c1,t,t1)$(((ord(h)-ord(h1))=1)*Discipline_sg(sg,c)*Discipline_sg(sg,c1))..
2*ocupation_prod_h(sg,s,s1,d,c,c1,t,t1)=g=ocupation(s,d,h,c,t,sg)-50*(1-ocupation(s,d,h,c,t,sg))+ocupation(s1,d,h1,c1,t1,sg)-50*(1-ocupation(s1,d,h1,c1,t1,sg));
calc_ocupation_prod_h2(sg,s,s1,d,h,h1,c,c1,t,t1)$(((ord(h)-ord(h1))=1)*Discipline_sg(sg,c)*Discipline_sg(sg,c1))..
2*ir(sg,s,s1,d,c,c1,t,t1)=l=ocupation(s,d,h,c,t,sg)+ocupation(s1,d,h1,c1,t1,sg);

* Hard Constraint: Each Discipline *c* associated to a given Student Group must have Nhours_c(c) per week
constr1_cad(c,sg).. **sum**((s,d,h,t), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c))=e=Nhours_c(c)*Discipline_sg(sg,c);

* Hard Constraint: In each day and hour, each room s can only be associated with one Discipline
constr1_sala(d,h,s).. sum((c,sg,t), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c))=l=1;

*Hard Constraint: Each Student Group can only have one class of a given discipline in a given day
constr2_cad(c,d,sg).. sum((h,s,t), ocupation(s,d,h,c,t,sg))=l=1;

*Soft Constraint: If Possible a Student Group cannot have classes of a discipline in consecutive days
constr3_cad(s,s1,h,h1,d,d1,t,t1,c,sg)$(ord(d1) > ord(d)).. (1-Discipline_sg(sg,c))*50+ocupation(s1,d1,h1,c,t1,sg)*ord(d1)+100*(1-ocupation(s1,d1,h1,c,t1,sg))-ocupation(s,d,h,c,t,sg)*ord(d)+100*(1-ocupation(s,d,h,c,t,sg))=g=Discipline_2(c);

*Hard Constraint: Each discipline, *c*, can only be in one room, s, in the same day, *d*, and hour, *h*
constr4_cad(s,d,h).. sum((c,t,sg), ocupation(s,d,h,c,t,sg)*Discipline_sg(sg,c))=l=1;

*Hard Constraint: Each Discipline, *c*, may be taughted by only one Professor
Constr5_discipline_professor(c,sg,t).. sum((s,d,h,t1), ocupation(s,d,h,c,t1,sg)*Discipline_sg(sg,c)) =e=Nhours(c)*Teacher_course(t,c,sg);

* Hard Constraint: Each group of students, sg, has only one Professor, *t*, for each discipline, *c*
Teacher_students(sg,c).. sum(t, Teacher_course(t,c,sg)=l=1

* Soft Constraint: Verify Teachers Preferences
constr5_cad(c,s,d,h,t,p,sg).. Relax_pref(t,c,p,sg)+ocupation(s,d,h,c,t,sg)*(ord(h) ge Pref_h1(t,c,p))*(ord(h) le Pref_h2(t,c,p))*(ord(d) ge Pref_d1(t,c,p))*(ord(d) le Pref_d2(t,c,p)) + (1-ocupation(s,d,h,c,t,sg)) + (Teacher_hours(t,c)=0) =g=1 ;

*Soft Constraint: If Possible a Professor must give classes of the same discipline with an interval of at least two days
constr7_cad(c,s,s1,d,d1,h,h1,t,sg,sg1)$(ord(d)>ord(d1))..
50*Relax_dist_c(t,c,sg,sg1)+ocupation(s,d,h,c,t,sg)*ord(d)-
ocupation(s1,d1,h1,c,t,sg1)*ord(d1)+ 50*(1-ocupation(s,d,h,c,t,sg)) + 50*(1-
ocupation(s1,d1,h1,c,t,sg1))+ 50*(Teacher_hours(t,c)=0) =g=2 ;

* Hard Constraint: A Professor cannot make more than N_aulas_max classes in the same day
constr8_cad(t,d).. sum((s,s1,h,h1,c,c1)$((ord(c) ne ord(c1))*(ord(h)>ord(h1))),ocupation_prod(s,s1,d,h,h1,c,c1,t))=l=N_classes_max_prof;

* Soft Constraint: IF POSSIBLE make the classes of a professor consecutive
constr9_cad(s,s1,d,h,h1,c,c1,t)$(ord(h)>ord(h1))..
ocupation_prod(s,s1,d,h,h1,c,c1,t)*(ord(h)-
ord(h1))=l=1+Relax_constr9_cad(s,s1,d,h,h1,c,c1,t)*1000;

* Soft Constraint: Verify Preferences of All students
constr5_cad_sg(c,s,d,h,sg,t,p).. Relax_pref_sg(sg,c,p)+ocupation(s,d,h,c,t,sg)*(ord(h) ge Pref_h1_sg(sg,c,p))*(ord(h) le Pref_h2_sg(sg,c,p))*(ord(d) ge Pref_d1_sg(sg,c,p))*(ord(d) le Pref_d2_sg(sg,c,p)) + (1-ocupation(s,d,h,c,t,sg)) + (Teacher_hours(t,c)=0) =g=1 ;

* Soft Constraint number of hours in a discipline =< Pref2
constr6_cad(c,s,d,h,t).. ord(h)*ocupation(s,d,h,c,t) - (1-ocupation(s,d,h,c,t))*100 - prod(p,Pref_h2(t,c,p)=0) * 100 =l= sum(p,Pref_h1(t,c,p)*(ord(h) le Pref_h2(t,c,p)));

* Hard Constraint: a teacher can give only a class at the same time
constr1_teacher(t,d,h).. sum((s,c,sg), ocupation(s,d,h,c,t,sg))=l=1;

* Hard Constraint: For each discipline, c, a teacher cannot give more hours than Teacher_hours(*t*,*c*)
constr2_teacher(t,c).. sum((s,d,h,sg), ocupation(s,d,h,c,t,sg))=l=Teacher_hours(t,c);

calc_obj.. obj=e=Flag_eq_h*1000*aux_eq_sum + sum((c,sg),(n(c,sg)-Nhours_c(c)))+5 * sum((t,c,p,sg),Categ(t)*Weights_pref(p)*Relax_pref(t,c,p,sg)) + 5*aux3 + 5 * aux4 + 50*aux5 + 1400*aux6 + 1350*aux61+ 1800* aux62 + 1900*aux7 + 150*aux8 + 5000*aux9 + aux10 + aux11 + 1200*aux12 + 500*aux13 + 5* aux2_sg;

calc_aux3.. aux3=e=sum((s,d,h,c,t,sg), ocupation(s,d,h,c,t,sg)*ord(h));
**Categ(t)
constr_aux3.. aux3=g=0;

* Soft Constraint: If possible maximize the number of hours of each teacher *t* in the same day *d*
calc_aux4.. aux4=e=15000-sum((s,s1,d,h,h1,c,c1,t)$(ord(h) > ord(h1)),Categ(t)*Weights_prod(t)*ocupation_prod(s,s1,d,h,h1,c,c1,t));
constr_aux4.. aux4=g=0;

calc_aux5.. aux5=e=sum((t,c,sg,sg1), Categ(t)*Relax_dist_c(t,c,sg,sg1));
constr_aux5.. aux5=g=0;

calc_aux6.. aux6=e=sum((s,d,h,c,t,sg), ocupation(s,d,h,c,t,sg)*ord(d));

```
constr_aux6..  aux6=g=0;

calc_aux61.. aux61=e=sum((s,d,h,c,t,sg), ocupation(s,d,h,c,t,sg)*ord(h));

calc_aux62.. aux62=e=sum((s,d,h,c,t,sg), ocupation(s,d,h,c,t,sg) * (ord(h)=h_max) );

calc_aux9.. aux9=e=sum((sg,d),Relax_constr3_st(sg,d));

calc_aux10.. aux10=e=sum((sg,d), Relax_constr_n_h(sg,d));

calc_aux11.. aux11=e=sum((sg,d),Relax_constr4_st(sg,d));

calc_aux12.. aux12=e=sum((s,s1,d,h,h1,c,c1,t)$(ord(h)>ord(h1)),
Relax_constr9_cad(s,s1,d,h,h1,c,c1,t)*(ord(h)-ord(h1)) );

*  Calculation of the number of hours of classes that a Student Group, sg, have in a day, d
calc_n_classes(sg,d).. n_classes(sg,d)=e=sum ( (s,c,t,h), ocupation(s,d,h,c,t,sg) ) ;

constr_obj.. obj=g=0;

calc_aux2_sg.. aux2_sg=e=sum((sg,c,p), Weights_pref(p)*Relax_pref_sg(sg,c,p));

Model Timetable /all/;

Solve Timetable using mip minimizing obj;
```