Extended Abstract: Algorithm for the STP.

Gonçalo Peres
Universidade Aberta
1800301@estudante.uab.pt
me@goncaloperes.com

Abstract – In this extended abstract, I review the timetabling problem, implement an algorithm for the school timetabling problem (STP), and report computational results on tests from a high diversity of situations.

Keywords – Scholar Scheduling, Algorithm, Education.

INTRODUCTION

The Academic Timetabling Problem consists scheduling a sequence of activities between teachers and students, satisfying a set of constraints [1]. There are different variants of the Academic Timetabling Problem and Schaerf [2] categorizes them as: Course Timetabling, Examination Timetabling and School Timetabling.

This Extended Abstract focuses on the School Timetabling Problem (STP) also known as Class-Teacher Timetabling Problem. This problem consists of scheduling all lessons for a week, for both classes and teachers, satisfying a variety of constraints.

The School Timetabling Problem is defined as follows:
. A set of teachers T
. A set of classes C
. Groups of lessons L
. A day of the week D.

The problem is to assign meetings between teachers and classes in such a way that no teacher or class can have 4 periods, knowing that a week has 5 days, 6 periods of lessons per day. The same pair Class/Teacher can only have one lesson per day.

In this context, there are penalty units, referred as "holes". They represent idle periods between lectures in the classes/teachers assignment of a day (more than one idle periods count as one "hole") and/or a day with only one lesson. The sum of all the penalty units represent the Cost.

SOLVING THE STP

I. Algorithm for the STP

The overview of the algorithm implemented to solve the STP is as follows:

```
initialize()
While improvement:
        // Try to change all lesson
        for L1 in lesson_list for L2 in lesson_list
                try_to_change_lessons(L1, L2)
                if there are improvement:
                        improvement = true

        // Try to change lessons of one teacher
        For t in teacher_list for L1 in lesson_list for L2 in
lesson_list
                try_to_change_teacher_lesson(t, L1, L2)
                if there are improvement:
                        improvement: true

        // Try to change lessons of one class
        For c in class_list for L1 in lesson_list for L2 in
lesson_list
                try_to_change_class_lesson(t, L1, L2)
                if there are improvement:
                        improvement: true
print result
```

In the algorithm, the penalty of the state is calculated by the following:

```
(the number of over-lessons where the teacher has more
than 4 lessons + the number of over-lessons where the
class has more than 4 lessons + the number of same pair
Class/Teacher in one day) * 100 + the number of holes
```

So, if there are improvement, this penalty is smaller than before.

II. Tests

To better understand the value of the Algorithm for the STP, ought to implement it and test it in a wide range of situations revolving around variation of certain controlled parameters, namely: number of classes (from 6 to 12), number of teachers (from 6 to 9), number of weekly lessons (generally equal in all of the classes – either 2 or 3 -, but varying in one of the instances from 1-4). The data containing the number of classes, the number of teachers and other relevant information of the instance is coming from .txt files created following the pre-stablished test conditions (to get these files, please contact the author).

III. Computational results

The Algorithm for the STP was coded Python and in C++, compiled using Dev-C++, and run on an Asus Laptop Intel® Core™ i7-55000U CPU @ 2.40 GHz with 12 GB of RAM using Windows 10. In the following Tables 1 and 2 and 3, the results are presented including number of classes (NC), number of teachers (NT), weekly lessons (WL), Cost (C) and Running Time (RT). The results for the C++ implementation will include the time taken for the process to exit, process exit time - PET.
(Note: The times are rounded to the 4th decimal place.)

TABLE 1. COMPUTATIONAL RESULTS USING PYTHON.

Instance	NC	NT	WL	Cost	RT (s)
A	6	6	2	12	84.8768
B	6	6	2	12	85.6323
C	6	6	3	28	114.6127
D	6	6	3	28	112.8635
E	9	6	2	22	287.9711
F	9	6	2	22	288.3887
G	12	9	2	20	648.9527
H	12	9	2	30	555.8060
I	9	6	1-4	25	284.2598

TABLE 2. COMPUTATIONAL RESULTS USING C++.

Instance	NC	NT	WL	Cost	RT (ms)	PET (s)
A	6	6	2	12	62.000	0.2081
B	6	6	2	12	62.000	0.2087
C	6	6	3	28	78.000	0.2054
D	6	6	3	28	78.000	0.1899
E	9	6	2	22	156.000	0.2866
F	9	6	2	22	171.000	0.2792
G	12	9	2	94006	203.000	0.3271
H	12	9	2	67817	187.000	0.2828
I	9	6	1-4	25	218.000	0.4756

It is possible to visualize certain patterns from the observation of the Tables.

In Table 1, we see that increasing the number of WL increases the Cost. Also, if we increase the NC and NT, the RT is higher, behaving as expected.

In Table 2, we see that by increasing the number of WL the Cost increases. The higher the NC and NT, the higher the RT and PET. In the instances G and H, with an higher NC and NT, we see a really high cost, meaning that there is an adjustment to be made in the C++ implementation, regarding higher numbers of NC and/or NT.

In terms of RT, even though the complexity is the same, the C++ implementation was considerably faster producing results after a couple of seconds.

CONCLUSIONS

The proposed algorithm finds interesting results for different number of classes/teachers/weekly lessons.

I found experimenting with different classes and teachers necessary to ensure that the method is correct and general enough.

In all practical cases, the algorithm, specially the one implemented in C++ was able to find a feasible timetable in a reasonable amount of time.

Acknowledgments. I wish to thank professor José Coelho for his guidance and availability during the whole Semester, specially during this final project, and Edmaro Peres and Tiago Peres for the comments on an earlier part of this project.

REFERENCES

[1] Lara, C., Flores, J. J., Calderón, F. (2008). *Solving a School Timetabling Problem Using a Bee Algorithm*.

[2] Schaerf, A. (1996). *Tabu search techniques for large high-school timetabling problems*.

ABOUT THE AUTHOR

Gonçalo Peres is a master's student at Universidade Aberta.