THOR[1]: A tool for school timetabling

Fernando Melício[1,3], João P. Caldeira[2,3], Agostinho Rosa[3]

[1] ISEL-IPL, R. Conselheiro Emídio Navarro, 1900 Lisboa, Portugal
fmelicio@deea.isel.pt

[2] EST-IPS, R. Vale de Chaves-Estefanilha, 2810 Setúbal, Portugal

[3] LaSEEB-ISR-IST Av. Rovisco Pais, 1, TN 6.21, 1049-100 Lisboa, Portugal
{caldeira, acrosa}@isr.ist.utl.pt

Abstract. This system is the result of our previous work on the subject of school timetabling. It was designed to respond mainly to Portuguese schools from various educational levels. It consists of three main blocks; a graphical user interface; an automatic scheduler and a relational database. This system is now in use by more than 100 schools in Portugal with significant success (http://www.fmaismais.pt).

1 Implementation Framework

School timetabling is a classical combinatorial optimization problem which is associated with a set of constraints. It consists of assigning a set of lessons to time slots within a time period (typically a week), satisfying a set of constraints of various kinds 3. The constraints that we have used are related to Portuguese schools and are the result of multiple discussions we had previously with people from several Portuguese schools. In Table 1 it can be seen a summary of the constraints we have considered in solving this problem.

The main issue regarding the constraints is that each user can weigh each constraint with a particular value and in this way different schools may have different values to each constraint.

A lesson is the teaching unit. It is characterized by the triple $\langle T^*, C^*, S^* \rangle$. Where T^* is a subset of the teachers set, C^* is a subset of the classes set and S^* is a subset of the subjects set. Each lesson has a duration measured in time slots.

There are two types of lessons:

1. *Simple lesson*. Where $|C^*| = 1$ and $|S^*| = 1$.
2. *Compound lesson*. Where $|C^*| \geq 1$ and/or $|S^*| \geq 1$.

[1] In Portuguese *THOR* stands for Tabelas Horárias which can be roughly translated by timetabling charts.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 532–535. ISBN 80-210-3726-1.

Constraint	*Description*
C_{0}	Number of time slots of lessons that aren't yet scheduled
$C_{1,2}$	Number of time slots of overlapped lessons (1-classes; 2-teachers)
$C_{3,4}$	Number of time slots exceeding the maximum allowed per day (3-classes; 4-teachers)
$C_{5,6}$	Number of time slots exceeding the maximum consecutive time slots allowed (5-classes; 6-teachers).
$C_{7,8,9}$	Number of preferable time slots filled (7-classes; 8-teachers; 9-subjects).
$C_{10,11}$	Number of idle time slots (10-classes; 11-teachers)
C_{12}	Number of time slots of lessons without a room assigned.
$C_{13,14,15}$	Number of time slots that are forbidden and are filled with lessons (13-classes; 14-teachers; 15-subjects)
C_{16}	Total number of teaching days for teachers
C_{17}	Number of repetitions of lessons of the same subject in the same class per day
C_{18}	Number of time slots that doesn't satisfy the predefined space between lessons.

Table 1. Constraint set

In general, a *compound lesson* means that we have several classes joined together to attend a certain subject or it means that a class can be subdivided into subgroups to attend special subjects, like laboratories, etc.

It is associated with each subject the kind of room it must have, i.e., the resources that there must exist in the room for a lesson of that subject should happen.

This software tool was designed to respond mainly to Portuguese schools from various educational levels. It is based on a modular implementation, and consists of three main blocks; a graphical user interface; an automatic scheduler and a relational database (Fig. 1).

It was developed in C++ using an object oriented technique. It runs on Microsoft Windows ® and the database is implemented in Microsoft Access®.

There were two main objectives with the development of this system:

1. It should be fairly easy to interact with it.
2. It would generate good timetables in an automatic way.

The graphical user interface (GUI) consists of several forms where it is possible to enter the school data (i.e., teachers, rooms, classes, subjects), and also change the individual schedules.

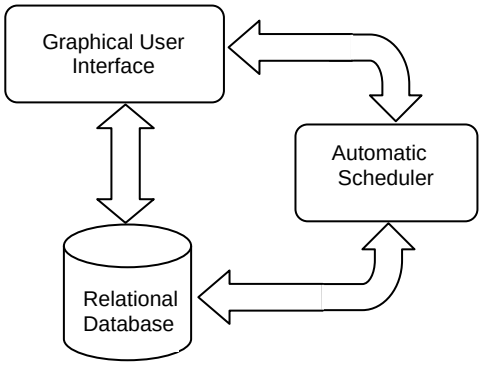

Fig. 1. Block diagram of the implemented system.

An individual schedule looks like an Excel sheet where there is always one or more time slots (cells) selected and we can do all the basic editing operations like delete, insert, undo, etc.

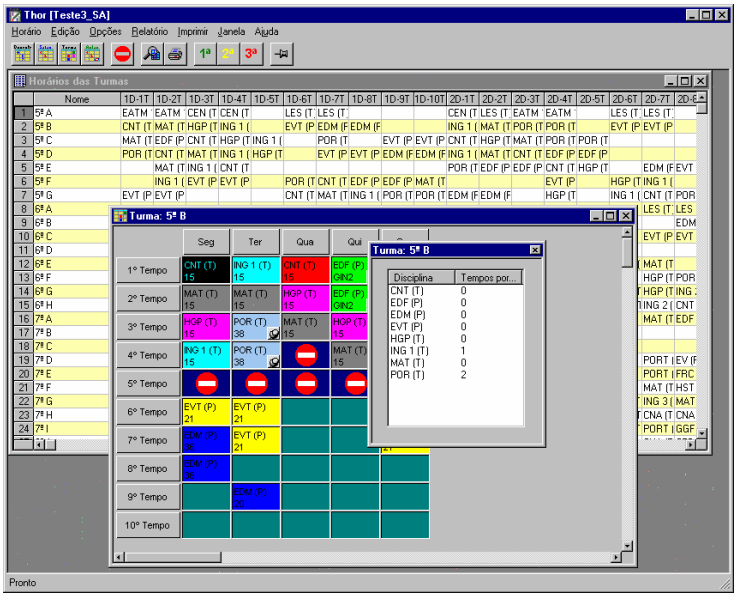

Fig. 2. Graphical user interface.

An interesting feature in this editor is that if you change the class schedules all the others are updated accordingly (teacher, room, etc.).

The Automatic scheduler contains two different and complementary algorithms:

1. An iterative algorithm based on a Fast Simulated Annealing implementation 5.
2. A heuristic constructive algorithm 4.

The Fast Simulated Annealing implementation is basically a typical Simulated Annealing algorithm 1 initiated at a lower temperature depending on the quality of the initial solution.

Also another important aspect of this particular implementation is the fast evaluation of a new solution as it is only evaluated the difference between the two solutions. It can be shown that this is most effective when a new solution only differs a small portion from the original solution 6.

There is also a deterministic algorithm for lessons that aren't scheduled. Its main objective is to schedule these lessons in any time slot available that satisfies the hard constraints.

2 Conclusion

As it is well recognized people in general are not interested in solving their optimization problems to optimality or even close to optimality. As Burke et. al. 2 stated people are more often interested in *"good enough – soon enough – cheap enough"* solutions to their problems.

We also think that good choices of specific parts of each problem are fundamental for the success of any search algorithm. As it is well known this specific problem is one with a large search space so it is very important to devise efficient techniques to search a good solution in it.

With this tool we intend to solve this problem if not to optimality at least to a very good stage with scarce resources.

References

1. Aarts, E. H. L., Van Laarhoven, P. J. M., Korst, J. H. M., "Simulated annealing", *Local Search in Combinatorial Optmization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley & Sons, 1997.
2. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S., *"Hyper-Heuristics: An Emerging Direction in Modern Search Technology"*. In Handbook of Meta-Heuristics, Glover, F., Kochenberger, G., (eds.), pp. 457-474, Kluwer, 2003.
3. de Werra, D. "An introduction to timetabling". *European Journal of Operational Research Society*, v. 19, pp. 151-162, 1985.
4. Hilbert H., "High School Timetabling in Germany-Can it be done with MIP?". *In Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling*, pp. 325-327, 1997.
5. Melício, F., Caldeira, P., Rosa, A., "Solving Timetabling Problem with Simulated Annealing". Filipe, J. (ed.), Kluwer Academic Press, pp.171-178, 2000.
6. Melício, F., Caldeira, P., Rosa, A., "Two Neighbourhood Approaches to the Timetabling Problem", *PATAT 2004, Practice and Theory of Automated Timetabling*, pp. 267-282. Pittsburgh, USA, Aug. 2004.