The Erlangen Advanced Time Tabling System (EATTS) Version 5

Peter Wilke

Abstract The latest version of the Erlangen Advanced Time Tabling System EATTS, a development and production environment to generate optimized timetables, is introduced. EATTS allows to describe timetabling problem with its associated resources, constraints, events and solutions using XML files for information interchange. A broad bandwidth of algorithms is available to generate solutions using single-core, multi-core or cluster of networked computers.

Keywords Implementation · Distributed Time Tabling System · Time Tabling Problem

1 Introduction

The Erlangen Advanced Time Tabling System EATTS is a development and production environment to generate optimized timetables. Timetabling problems are quite common and come in different versions, among them rosters, schedules and school timetables. They have in common that given resources have to be used as efficient as possible and that this requires planning with respect to the given constraints to obtain a decent plan.

The example shown here is a school timetabling problem. EATTS is designed to be capable to solve all types of timetabling problems, so the example shown here is selected because it is quite common and requires no special explanation. Here we focus on the overall structure of the systems. Its ability to describe timetabling problems is describes in detail in a paper focusing on the XML-based specification [Ost10].

Peter Wilke
Universitaet Erlangen-Nuernberg, Informatik 5, Martensstrasse 3, 91058 Erlangen, Germany
Tel.: +49 (9131) 85-27998
E-mail: Peter.Wilke@Informatik.Uni-Erlangen.DE

2 Resources

When looking at a school timetable the events to be planned are lessons, to which the resources like teachers, classes ans rooms have to be assigned. All resources are typed. Each type has as many attributes created by the user as required. Planning a timetable usually begins with compiling the resources, either by reading in a file or typing in them manually. The screenshot fig. 1 shows the input dialogue to enter the attribute values for a room. All resources of type "Room" have attributes name, capacity and groups. Groups indicated what type of subject can be taught in this room and to which pools the rooms belongs.

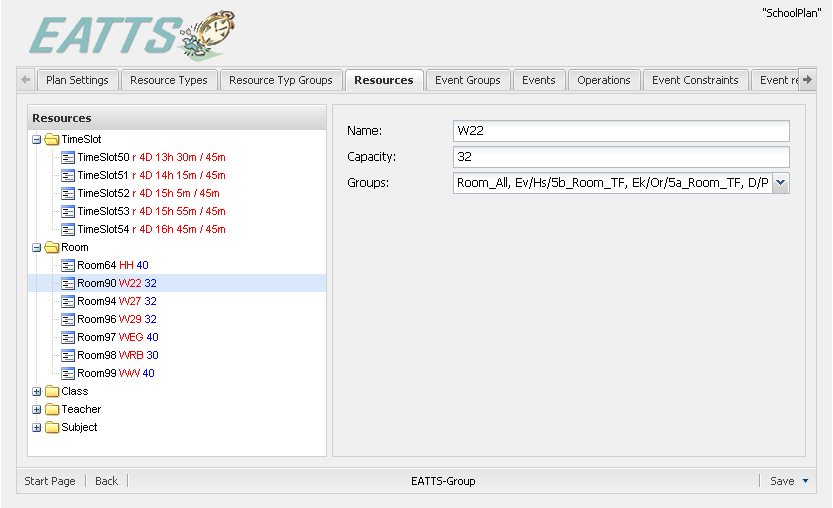

Fig. 1 The EATTS dialog to edit resources

3 Constraints

The specification of constraints is usually more complex than describing resources or events. On one hand a precise specification is required and on the other hand the current setting should be presented clearly to find gaps and/or inconsistencies, which can't be avoided automatically by EATTS but its user. Constraints come in different flavours, therefore a flexible way to specify them is necessary. EATTS allows to refer to resources, their object classes and all attributes. Depending on the type of the attributes, among them are integer, double and string, arithmetic and logical operators can be used to specify the constraint. In addition the parameters of the cost function are set, to compute the correct penalty point if the constraint is violated. The screenshot fig. 2 shows a constraint assigned to an event. Here the capacity of the class room "Room Size" must be greater than or equal (geq) to the number of students "Class Size". The cost function is specified as $50 * (1 * x^0 + 2)$. X denotes the number of violations of this constraint and the constants a chosen by the user according to the problem, e.g. severity of the constraint violation.

4 Modes of Operation

EATTS could be run in normal mode which interprets constraints the useual way. A unique property of EATTS is the option to specify a constraint not only as "has to be fulfilled (hard)" or "should be fulfilled (soft)" but also as "can be violated in exceptional case (soft hard)". This allows to violate constraints when it is acceptable, e.g. a room isn't available due to a broken water pipe, a teacher isn't available due to a traffic jam. In these cases a timetable should be created which is similar to the one currently in effect but violates some constraints to minimize changes. The mode is called the emergency mode as this is a typical approach in these cases, but it is not restricted to theses cases, e.g. finding out what effect a change in resources has on the resulting timetable: is it critical or is an investment in this resource required or efficient.

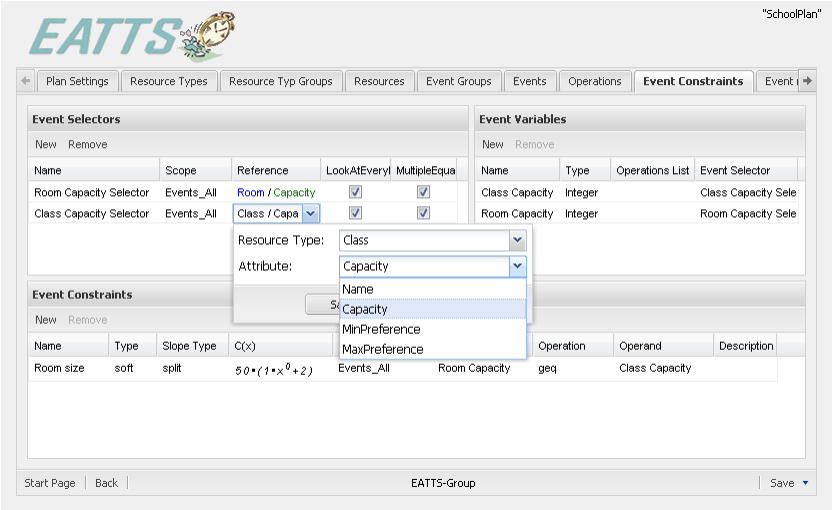

Fig. 2 The EATTS dialog to edit constraints

5 Events

The planning process is all about events. The timetable - as a result of the planning process - determines which resource is assigned to which event. For every event it can be declared how many resources of a certain resource type have to be assigned as minimal or maximal value so that the resulting timetable is a valid solution. Two additional subsets of resources of this type can be defined: one which is assigned fixed to this event and/or one subset of resources which could be assigned, at least one of them must exist.

6 Algorithms

Core of the planning are the algorithms. Depending on the nature of the given constraints and desired properties of the timetable the user can choose among a large number of available algorithms, among them Genetic Algorithms - Evolutionary Algorithms - Branch-and-Bound - Tabu Search - Simulated Annealing - Graph Coloring - Soft Computing - Swarm Intelligence - Great Deluge - Ant Colony - Jump Up Walk Down. The beginner might choose one of the pre-configured algorithms while the experienced user might prefer to set up his own parameter set or implement his own algorithms. All algorithms can be arranged in experiments which allows arbitrary computational sequences. The screenshot fig. 3 shows the parallel execution "ParallelAlgorithm" of two algorithms, namely "Simulated Annealing" and "Tabu Search". The results are compared and the best result is returned "BestResultMerger". All parameters of an algorithms can be displayed and edited. The configuration of an experiment can be saved and be used as template for a new experiment or just executed one more time.

EATTS is written in Java. As all algorithms use the same interface to the framework it is possible to excecute implementations of algorithms provided by the user. A tool to generate a template for the source code is under development, the byte code will be accessed via a class loader.

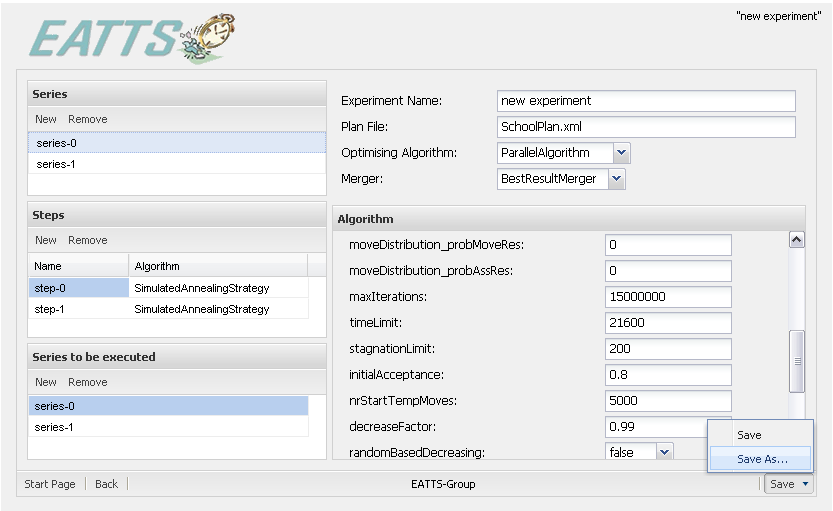

Fig. 3 The EATTS dialog to create experiments and to control algorithms

7 Running Experiments

The algorithms can be executed on a dedicated server or can be distributed over a TCP/IP network on additional computers. The screenshot shows the dialogue where the user can select the experiments and start their execution. The browser contacts the server regularly and updates the status information, including the costs of the best

plan found so far and an estimated remaining computation time. At the end of the computation results are stored and the data required for the views are generated. Now the user decided whether the results are satisfactory or additional computations are required.

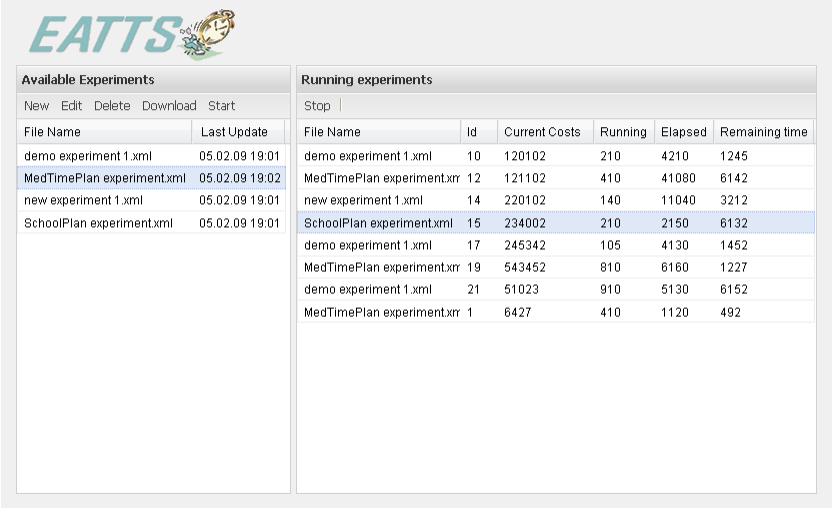

Fig. 4 The EATTS dialog to control the parallel computations

8 Results

Timetables are the output of the planning algorithms and can be stored in different file formats and views. The screenshot fig. 5 shows the view of a student on the generated plan, e.g. he sees his personal timetable consisting of the lessons he has to attend to. Other views can be created instantly, for a teacher, e.g. a headmaster or a caretaker. All users access the EATTS through a browser providing an interface according to the privileges of the user.

A common view indicates which constraints are currently not satisfied naming the events and resources. Based on this information the administrator can decide if he would like to edit the resources, events or constraints or use a different algorithm. The screenshot shows timeslot clashes (Monday, 9:45, and Tuesday, 9:45) and therefore the algorithm should be given more time to find a solution or another algorithm should be given an shot.

9 Implementation

The software is implemented in Java and available for numerous platforms, among them Windows and Linux operating systems. To run EATTS the following free-ware software products are required:

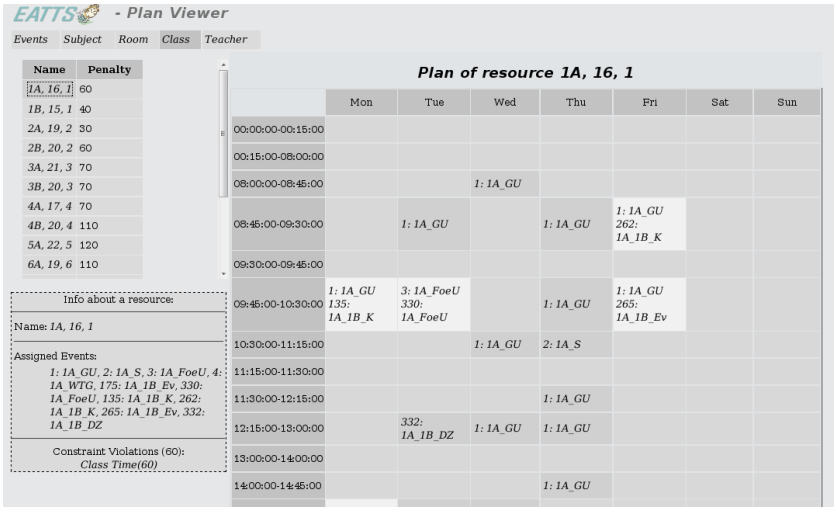

Fig. 5

– Browser to display the user interface, JavaScript must be enabled
– Java Runtime Environment (JRE), v5.0 or above
– computers connected via TCP/IP, if additional computing power is required (optional)

The EATTS Server is a software server, i.e. a PC or compute server can be used to host the EATTS Server.

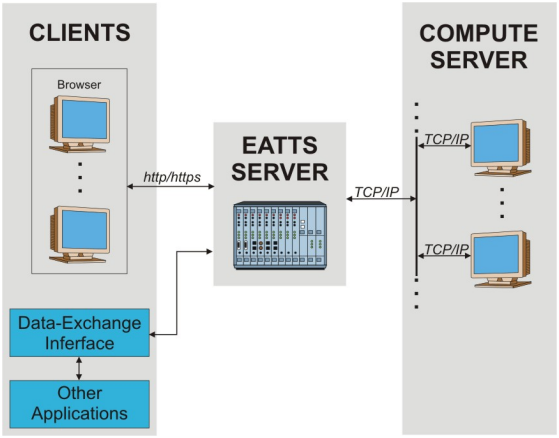

Fig. 6 The components of the Erlangen Advanced Time Tabling System

10 Conclusion

the Erlangen Advanced Time Tabling System is a framework to design and develop solutions for all kinds of timetabling problems. It can be used to experiment with algorithms and used as production system for real world systems.

If you would like to test the Erlangen Advanced Time Tabling System please feel free to contact the author for details.

References

[Ost10] Peter Osterler, Johannes; Wilke. The erlangen advanced timetabling system (eatts) unified xml file format for the specification of timetabling systems. page 18p, 2010.

Acknowledgements We would like to thank all our colleagues at EATTS for their support and special thanks to: Norbert Oster, Stefan Büttcher, Dr. Matthias Gröbner, Dimo Korsch, Sabine Helwig, Hichem Essafi, Marlene Gagesch, Monic Klöden, Georg Götz, Andreas Konrad, Tarek Gasmi, Sabeur Sarai, Kerim Merdenoglu, Helmut Killer, Eugen Kremer and Johannes Ostler.