An XML format for Benchmarks in High School Timetabling

Gerhard Post[12], Samad Ahmadi[3], Sophia Daskalaki[4], Jeffrey H. Kingston[5], Jari Kyngas[6], Cimmo Nurmi[6], David Ranson[7], and Henri Ruizenaar[18] ⋆

[1] Department of Applied Mathematics, University Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands.
[2] ORTEC, Groningenweg 6k, 2803 PV Gouda, The Netherlands.
[3] School of Computer Science, De Montfort University,
The Gateway, Leicester, LE1 9BH, UK
[4] Engineering Sciences Department, University of Patras, 26500 Rio Patras, Greece
[5] School of Information Technologies, The University of Sydney, Australia
[6] Satakunta University of Applied Sciences, Tiedepuisto 3, 28600 Pori, Finland
[7] Department of Informatics, University of Sussex, Falmer, Brighton, BN1 9QH, UK
[8] Stedelijk Lyceum, Locatie Kottenpark, Lyceumlaan 30,
7522 GK Enschede, The Netherlands.

Abstract. The High School Timetabling Problem is amongst the most widely used timetabling problems. This problem has a varying structure in different high schools even in the same countries or educational systems. Due to lack of standard benchmarks and data formats this problem has been studied less than other timetabling problems in the literature. In this paper we describe the High School Timetabling Problem in several countries in order to find a common set of constraints and objectives. Our main goal is to provide exchangeable benchmarks for this problem. To achieve this we propose a standardisation of data formats suitable for different countries and educational systems. This data model is defined by an xml schema. The schema and datasets are available online.

1 Introduction

One of the best known timetabling problems is the High School Timetabling Problem; most people have some experience of timetables from their school days, with probably a not always positive opinion on it. There is an implicit belief that all High School Timetabling instances are similar, and that a computer program can always 'solve' it.

In reality, the research in this area is still very active and we are not even near to solving all the instances of the High School Timetabling Problem to optimality. Moreover, continuous reforms in educational systems all over the world, generate new problems to tackle. On the literature side there are papers on general timetabling problems such as [30,9,28,7,31,4]. Some of their concepts

⋆ This research has been supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

and/or methods can be used in real-life timetabling problems. On the other hand, there are also case studies on high schools in different countries, some of which are mentioned in Section 2.

In our view, the key obstacle to scientific research on the High School Timetabling Problem is the lack of standard benchmarks and exchange of data. As a consequence the datasets used in the context of a research never seems to be examined by other independent groups of researchers. We consider this as a missed opportunity which could be explored by providing an xml-format for existing and new High School Timetabling datasets. As a starting point, the authors listed agreed to contribute a dataset in this xml-format, which will be made available on the internet at
`http://wwwhome.math.utwente.nl/~postgf/BenchmarkSchoolTimetabling/`.
The reason for choosing xml is that xml is very extensively used nowadays, like in web services, or as means to exchange data between different applications. The power of xml lies in the fact that the data is structured, and therefore it can easily be adjusted to changes, an aspect lacking in plain text files. Modern computer languages have methods to handle xml in an easy way. Mathematically speaking, xml is a rooted tree, with information attached to the nodes. This information can contain cross references. Pieces of xml are presented in Section 4; we refer also to [34].

The problem of exchanging timetabling data was discussed at the first International Conference on the Theory and Practice of Automated Timetabling [10], where it became clear that the principal difficulty was the precise expression of the many different kinds of constraints. There may be ten or more of these in any one instance, and they vary widely between institutions.

This complexity with the specifications of the Timetabling Problem has been addressed in several ways. Some papers have tried to generalize and unify the constraints [8,20]. Others have adapted existing technologies in which constraints may be expressed, such as XML and the semantic web [11,12,13,24], or object-oriented modeling and frameworks [16,26]. Others have expressed constraints as logic expressions within specifically designed specification languages [3,18,21]. There has been at least one attempt to simply enumerate every possible constraint [27]. While much of this work is more general than our approach (for example, the use of MathML functions in [24] allows essentially arbitrary constraints to be expressed in XML format), none of it has led to significant data exchange.

Another approach is to restrict the problem domain to one particular kind of timetabling, then use judicious simplification to further reduce the specification burden while maintaining the essence of the problem. The Carter data sets for examination timetabling [6] omit many details, notably all data related to rooms, and similar simplifications appear in the Traveling Tournament Problem [14] and the International Timetabling Competition [25]. These are some of the most successful examples of timetabling data exchange so far. However, judicious simplification has been criticized for contributing to the gap between research

and practice [5], at least in examination timetabling; and the data transfer has almost always been in one direction only.

2 The problem in practice

Throughout the world, timetables are constructed for all different types of schools. Before giving details for the high school timetabling problems in different countries, we provide an overview of the principal differences. In the following subsections we describe the situation in Australia, England, Finland, Greece and the Netherlands. Not only the problems, but also the terminology varies among countries. Therefore, we give a short overview of the terms we use; for more details we refer to Subsection 3.1.

The students are organized in *base groups*, and follow *lessons*. The lessons of a (usually fixed) group of students in the same *subject* (like math, arts, ...), usually with the same teacher, we will call a *course*.

Some schools demand *compact* schedules for students, which means that students have no idle times. An *idle time* for a student is a free *period* (or *time slot*), while earlier and later on the same day the student does have a lesson.

2.1 Some general remarks

The timetabling process in a given school is influenced by the school organization, and specifically the handling of the three main groups of resources: students, teachers, and rooms.

Students The most prominent resource of high school timetabling is the student. Probably all high schools have a 'base group' for the students. In our model we call them 'Student divisions'. These base groups form a partitioning of the students: each student belongs to exactly one base group. For the timetabling process, there are two principally different categories for the base groups:

1. Base groups that is never split up over different lessons, i.e. if two students of a base group have a lesson at the same time, they are in the same lesson.
2. Base groups whose students can have different lessons at the same time.

In practice not many schools will fall 'completely' in the first category. Some schools, for example, will split two base groups of boys and girls for gym lessons. Similarly, other schools may split base groups for reasons such as religion, different levels of ability in a subject, etc.

Teachers Teachers in high schools are normally preassigned to lessons by the faculties/departments, or the school management. This considers all the constraints on the number of staff, their expertise and also special cases which will be difficult to model for an automated system and resolve them as hard constraints. Some examples of such special cases are giving new teachers lessons in parallel

groups, balancing teachers assigned to a base group of students (male/female, experienced/less experienced), or avoiding parent-child combinations in lessons. However in countries where no free periods are allowed, if teachers are preassigned to lessons, it will be very hard (or impossible) to find a feasible schedule. In such cases, if teachers are preassigned to lessons, it is important to find schedules with a low idle time for the teachers.

Rooms Though most lessons will take place in a room, it is not implied that room assignment is an issue. In some schools, rooms are preassigned to a base group of students, or to a teacher. It is also possible that the number of rooms is so large, that scheduling the rooms afterwards does not constitute a problem. While in some schools this is may be the case for the general rooms, it is possible that specialized rooms are scarce and need careful scheduling. Here one can think of rooms for Physical Education, Music, Arts, or Labs for physics, biology and chemistry.

As far as the objective function is concerned, the situation is diverse as well. Here we mention two cases, that sometimes appear as hard constraints.

1. Compact schedules for students, which means schedules for students without idle times. In some countries this is automatic, as a student has as many lessons as time slots available. In cases with a large variety of optional subjects it is usually impossible to have compact schedules, and we merely try to minimize the idle times.
2. Spreading of lessons for teachers. Sometimes, like with students, this is almost automatic for full-time teachers. For part-time teachers other objectives become important as well. For example, a teacher with an assignment of only 5 time periods per week will probably be unhappy with a schedule that spreads them over all days of the week.

2.2 Australia

Australian high schools have short school days, with correspondingly higher teacher utilisations, and consequently no demand for compactness in their timetables. Most teachers (those with no special duties) teach for 75% of the times of the cycle, and the total workload of all teachers is almost 100% of the available workload permitted by the teachers' workload limits. These conditions commonly prevent teachers from being preassigned, except to a few key subjects (e.g. the senior ones), and force some courses to be shared between two teachers. Minimizing the number of these *split assignments* is a key goal. There is a very high utilization of specialized rooms, notably Science and Computer laboratories.

The above describes the situation in government funded high schools. Privately funded high schools are often better resourced, with lower teacher workloads and more laboratories. However, they typically have lower student numbers and a commitment to high student choice, leading to peculiarities such as *composite classes*, in which students of different ages study a common subject together under a single teacher, creating other difficulties.

Another Australian problem concerns the number and distribution of lessons for each subject. In the senior years this is very simple: for example, a senior student might attend six subjects, each occupying six times per week, spread across the five days, including a double period (i.e. two adjacent times not separated by a break) on one of the days. Student choice is catered for by means of *electives*: lists of subjects running simultaneously from which each student chooses one. In the junior years a desire to permit students to sample a wide range of subjects leads to a chaotic curriculum from the timetabling point of view. A few subjects (English, Mathematics, and Science) follow the senior pattern, but the remainder of the cycle is occupied by many small subjects with more or less random numbers of times.

For previous research and additional details we refer to [1,19].

2.3 England

In English Secondary Schools students study a mixture of compulsory and optional subjects. For the 11 to 14 year-olds all subjects are compulsory and students study 10 main subjects plus a modern foreign language, citizenship education, careers education and sex education. There are fewer compulsory subjects for 14 to 16 year-olds. In addition, schools must offer students some work-related learning during this phase.

Depending on the policy of the school, within the year group, students may be divided into base groups with a designated teacher who has organizational and pastoral care responsibilities.

Another common grouping structure used in British schools is a Band which is a collection of several tutorial groups. Normally, all the base groups in a year are allocated to fewer bands. The mixing of base groups is done within a band. Subjects are grouped into blocks: a block consists of one or more subjects for different groups which will be scheduled at the same time. Blocks are constructed manually before the scheduling process starts.

In British schools students, except for those attending Sixth Form, do have compact schedules. They have as many lessons as there are time slots available.

In general there is a large variety of different length of lessons, one or two weekly timetabling cycles, length of school day and hence different number of lessons in different schools. This requires any timetabling system for English schools to be highly parametrised to accommodate different patterns of delivery in different schools.

For previous research and additional details we refer to [33].

2.4 Finland

A basic goal in Finnish school timetabling is to construct a one-week schedule, which is repeated the whole season. A timetable consists of lessons of subjects, where a lesson is a predefined combination of a student group, a teacher, a room and the duration of the lesson. Every student belongs to one base group

and most of the lessons are scheduled based on this group. In addition, the student can belong to a number of optional groups, which are built according to students enrolments to optional courses. Every lesson is assigned to one and only one group, either base or optional. Base and optional groups define a boolean matrix, which in turn determines which groups can or cannot have lessons at the same time.

In all Finnish school levels teachers are preassigned to lessons. Typically also rooms are preassigned to (most of the) lessons, based on the preferences of the teachers.

In a typical curriculum a student attends ten subjects and each subject is taught two to six hours each week. Lessons of a subject can take either one or two hours and only in exceptional cases three hours. Hence, most of the subjects are taught either in two, three or four days each week.

Compact scheduling for students (student groups) is necessary and breaks are either strictly prohibited or highly inappropriate. Teacher breaks are allowed, but still unappreciated. Several teachers also prefer not to teach more than a given number of hours in a day.

Preassignment of teachers and rooms, a somewhat complicated structure of student groups and the demand for compact scheduling make Finnish School Timetabling Problem a challenge for both a manual solver and a computer software. A further description of the Finnish problem can be found in [22].

2.5 Greece

Secondary education in Greece is divided in two portions, one for grades 7 to 9 and the other for grades 10 to 12. The lower portion is called Gymnasium and the upper portion Lyceum. The last two years in the lyceum are considered as preparatory for the higher level of education and the students are asked to choose one of three directions. Similarly, in vocational lyceums the students are asked to choose one of several specializations.

Regarding the timetable construction common aspects for the two schools are the five days of the week, the number of time-periods per day (six to seven), the preassignment of courses to teachers who may be full-time or part-time, and the requirement for compact student schedules.

In the gymnasium all students of a given base group attend the same lessons during most of the time in their dedicated classroom. However, for a small portion of the weekly timetable some base groups split into sub-groups or reshuffle with some other base group and split again for attending certain special courses. For these courses it is required that two teachers are teaching simultaneously to two different sub-groups of students, or alternatively that two teachers are collaboratively teaching to the same group of students.

In the lyceum the general practice is to keep students in their regular base groups for a portion of the day to attend their common courses, then reshuffle and split again based on specializations, directions or elective courses until the end of the day.

Important objectives for timetabling in the gymnasium are to schedule core courses evenly during the week and during prime time-periods and to maintain a minimal number of idle times for the teachers. For the lyceums however, because of all the above scheduling considerations the timetabling objectives change compared to the gymnasium. Balanced distribution of the core courses during the week is still important; however, the other objectives are not possible to satisfy completely.

For previous research and additional details we refer to [2,29].

2.6 The Netherlands

In Dutch secondary schools different levels of education are offered within one school. Teachers are shared among these levels. In the lower years the students in a base group follow the same courses, while in the last years, the students choose a specialization, with some compulsory and some optional subjects. The timetable is usually weekly, and valid for a period of 6 weeks, a trimester, semester, or the whole year. In the lower levels compact schedules often are compulsory. In the higher levels compact schedules are impossible to realize for all students, and in the opinion of the school administration not necessary. However, avoiding idle times as much as possible for students as well as teachers is still important.

Usually the teachers are preassigned to the lessons by the school administration, to ensure a good spreading of the teachers. The majority of the teachers work part-time. According to collective labour agreements these teachers are entitled to one or more days off.

The rooms are generally not preassigned to lessons. A lesson requires a room of a certain kind, and lessons should be assigned such that enough rooms of each roomkind are available. High utilization occurs for specialized rooms, like the gyms.

Some schools have special arrangements for students good at sports, dance or ballet, allowing them to skip lessons at the beginning or the end of some days. Some schools experiment with large groups of students (50 to 60 students) in a learning studio. Effectively there are two teachers visiting them in a period, of which one acts as supervisor, and the other one teaches the subject. All these special constructions make the task of timetabling even more challenging.

For previous research and additional details we refer to [15,32,17].

3 Modeling the problem

The basic game we play in modeling is between abstraction and concreteness. In an abstract sense, we could do with events, and express all extra constraints in the events. For accessibility we decided to keep a few extra terms, because these are important to describe constraints. These aspects we call the *assets* of the problem. The assets we use fall into four categories: Time, Courses, Subjects, and Resources.

3.1 Assets

We divide *time* into *time slots*. In many constraints it is essential to know the relation between time slots, for example for idle times, or the number of time slots per day. For this reason, we attach the following properties to a time slot.

- *SequenceNumber*. The time slots are ordered by the sequence numbers (which are assumed to be unique non-negative integers).
- *Day*. This (optional) field describes the day of the week the time slot belongs to.
- *Week*. Also optional, now for the week.

In the constraints there is no direct reference to days or weeks; if these are needed, they are replaced by time slot groups, where each group contains the time slots of one day or week.

A *course* is a collection of events that involves a group of students and a subject. Events that refer to the same course, are called *lessons* of the course. In other papers the set of lessons might be called a class. The most common constraint related to courses concerns the distribution of the lessons of a given course in the week (or in different weeks). In situations where the teachers are not preassigned to courses, an important constraint is that all lessons of a given course should be assigned to the same teacher.

We think of *subjects* in the general sense, like 'mathematics' or 'history'. One can associate a subject to a course. The reason to include subjects is that sometimes one wants to avoid sequences of lessons in certain subjects. For example a student might not be happy with the lessons in 'Chemistry', 'Physics', 'Mathematics', and 'Biology' in a row.

A *resource* is an entity with time restrictions. The most common resources are the students, teachers, and rooms. The resources may create subgroups, for example rooms of certain type or student groups. Additional resources may be considered equipment, for example a video projector. To reflect the time-limited use of resources, they carry one basic property: *multiplicity*. Using a resource more often than its multiplicity in a time slot is called a *clash*. To construct a clash-free schedule is one of the basic (hard) constraints.

We gave resources no additional properties; all needed aspects are transferred to the constraints, where the valid resources can be selected.

As stated before, we distinguish three special resources:

- *Students*. Usually a student group is preassigned to the events, but not always. Constraints that can be important for students are controlling the number of idle times, the number of lessons per day, and the sequences of subjects.
- *Teachers*. In some countries teachers are preassigned to courses, while in others they are not. In the latter case the assignment is based on the workload. General issues are the total number of idle times, the number of assigned time slots per day, the number of days with events, and the unavailabilities (either for days or time slots).

– *Rooms*. Most events take place in a room. If rooms abound, we can disregard them, if not they can be a bottleneck, especially for specialized rooms. We did not introduce any special constraints for the rooms. However we introduced one generally formulated constraint, meant for schools with more than one location. In this situation we can add a resource representing the location of an event, and we can require that the location should remain the same throughout some specified time slots.

Just like with resources, for all types of assets, and also for the events, we allow *grouping*. So it is possible to use TimeSlotGroups, SubjectGroups, CourseGroups, ResourceGroups, and EventGroups. This makes the formulation of constraints much more readable and compact. For example, an instance will usually contain the 'NoResourceClashConstraint' as hard constraint, for all resources, i.e. for the resource group consisting of all resources.

3.2 Events

Events are the basic scheduling object. An event can represent either a single lesson, or a set of ('linked') lessons, that have to be taught at the same time. Other events, such as staff meetings, are also permitted. An event has the following properties.

– The *duration* is the number of time slots that have to be scheduled to the event. It is assumed that the time slots assigned to the event are *consecutive*.
– The *courses* belonging to the event. If several courses are united in one event, we have no clue about which student belongs to which course.
– The first *time slot* that the event is scheduled to. If the duration is larger than one, the event will occupy the next time slots as well. Constructing a timetable usually involves assigning a time slot to all events.
– In some countries the teachers have to be assigned to the lessons, based on sharing the *workload*. Usually the workload will be related to the duration, but to be more flexible, we introduced the workload.
– The *resource groups* are preassigned, the resources within the resource group might be subject of scheduling. We give an example for a resource group of students.
  Suppose we have an optional subject at some level, which is taught in two courses. We have the freedom to divide the students, that chose this subject, over the two courses. The way we can model this is by creating events for the two courses, and adding resource groups StudentGroup1 and StudentGroup2 to these events, respectively. Using the 'PartitionResourceGroupConstraint', the students can be assigned to the groups StudentGroup1 and StudentGroup2.
– The *resources* of the event, either preassigned, or to be assigned. The resources of the event are marked by a 'role', to be able to distinguish them, when assigning them to the event. We give some examples.
  • The role can be 'room', and the room has to be selected for a given group of rooms, which were selected by room type and capacity.

- The role can be 'teacher', and the teacher has to be selected from a given group of qualified teachers.
- There could be two teachers required in the roles of 'senior teacher' and 'junior teacher'.

3.3 Constraints

For the moment we have listed 13 constraints. We expect that the set of constraints will grow in the future. A constraint can be hard ('Required') or soft. In both cases the constraint generates a cost, which either contributes to the infeasibility value, or to the objective value. The constraints describe all aspects of the scheduling, even the (elementary) assignment part. The big advantage of this approach is that we can distinguish between levels of infeasibility: if not all events can be scheduled, we can give preferences among them. In Subsection 4.1 we give an example how a constraint contributes to the objective function.

The cost of a schedule consists of four parts: the cost of a resource, cost of a course, cost of an event, and cost of a resource group.

We group the constraints in three groups: constraints describing the basic scheduling problem, other constraints for events and courses, and constraints for resources.

Scheduling constraints

- *AssignTimeSlotConstraint* (Cost per event). Assign each of the selected events to one of the selected time slots. Note that if the duration of an event is 2, we usually excluded the last time slots on the day in the selected time slots.
- *AssignResourceConstraint* (Cost per event). Assign a resource from the selected resource groups to the role in each of the selected events, taking preferences into account.
- *PartitionResourceGroupConstraint* (Cost per resource group). Partition a total resource group over a number of resource groups, taking sizes into account.

Course and event constraints

- *LinkedEventsConstraint* (Cost per event). Schedule the selected events at the same (starting) time slot.
- *CourseSpreadingConstraint*. (Cost per course). Schedule the events of the selected courses at most once in each of the selected time slot groups.
- *AvoidSplitAssignmentsConstraint*. (Cost per course). Schedule the role (of the events of) the selected courses to the same resource.

Resource constraints Resource constraints describe the quality of the timetable of a single resource; the corresponding cost is attributed to the resource.

- *NoResourceClashConstraint* Schedule the selected resources without clashes. This is one of the basic (hard) constraints.
- *WorkloadAssignmentConstraint* Schedule workload to the selected resources between a minimum and a maximum.
- *IdleTimesConstraint* The number of idle times in the selected time slot groups should lie between a minimum and a maximum for each of the selected resources. Typically the time slot groups are a day or all days.
- *TimeSlotAmountConstraint* The number of occupied time slots for the selected resources should lie between a minimum and a maximum for each of the selected time slots. Typically the time slot groups are the days, or a single time slot, where the resource prefers an off hour.
- *ClusterTimeSlotConstraint* The number of time slot *groups* with an assigned time slot should lie between a minimum and a maximum for the selected resources. Typically the time slot groups are days; for example a teacher requiring at most 3 days with lessons.
- *SameResourceConstraint* The resource assigned to (the event of) the selected resources should remain the same after the selected time slots.
- *SubjectSequenceConstraint* Avoid sequences, longer than a maximum, of events with the selected subjects for the selected resources within the selected time slot groups. Typically the time slot groups are the days.

4 The xml-format for benchmarks

One of the mark-up languages that has been gaining grounds in the last several years is xml [34]. Other mark-up languages are LaTeX or html. Xml is used extensively for exchanging data between different applications. Most object-oriented languages, like Java, C++, C#, and Delphi, have extensive libraries to work with xml. For this reason xml seems to be very useful for benchmarking. A comparable project in Nurse Rostering is done by Tim Curtois, see [23].

The xml-definition is defined by an xml-schema, which is an xsd-file (and is itself written in xml). This schema defines what elements have to be present and can be present in the xml-file. For a complete description, and an additional explanation of the present model, we refer the reader to the URL:

http://wwwhome.math.utwente.nl/~postgf/BenchmarkSchoolTimetabling/,

where a list with terms used can be found, as well as a detailed description of the constraints that we introduced.

At top level the xml-file looks like:

```
<Problem>
  <Instance>
    <Assets>
        ...
    </Assets>
    <Events>
```

```
      ...
    </Events>
    <Constraints>
       ...
    </Constraints>
  </Instance>
  <Solutions>
     ...
  </Solutions>
</Problem>
```

Hence the file contains two main sections: the instance (precisely one), and a number of solutions. The instance contains the sections 'Assets', 'Events', and 'Constraints', as in the data model (Section 3). Here we will give some details on the IdleTimesConstraint, and the Solutions.

4.1 Constraints

An idle time is relevant to a resource and to a group of time slots. If there is a time slot without event scheduled to the resource, but with scheduled events before and after in the same time slot group, we call it an idle time for the resource within this time slot group. Usually the time slot group will consist of all time slots of a day. In the constraints section of the xml document, we could have, among other constraints, the following constraint:

```
<IdleTimesConstraint Id="StudentIdleTimes">
  <Name>At most five idle times for students per week</Name>
  <Required>false</Required>
  <Weight>10</Weight>
  <CostFunction>Quadratic</CostFunction>
  <TimeSlotGroups>
    <TimeSlotGroup Reference="Monday"/>
    <TimeSlotGroup Reference="Tuesday"/>
    <TimeSlotGroup Reference="Wednesday"/>
    <TimeSlotGroup Reference="Thursday"/>
    <TimeSlotGroup Reference="Friday"/>
  </TimeSlotGroups>
  <Minimum>0</Minimum>
  <Maximum>5</Maximum>
  <ResourceGroups>
    <ResourceGroup Reference="AllStudents"/>
  <ResourceGroups>
</IdleTimesConstraint>
```

The attribute 'Id' and the first four fields are present in all constraints. They define a unique reference (Id) and a display name (Name), and give information on how to interpret violating this constraint. Required="false" means that this

constraint is a soft constraint, and hence the cost will be added to the objective value. The Weight and CostFunction explain how to calculate the cost for the deviation of the violation. In this constraint a violation is that one of the students in the resource group 'Students' has more than 5 idle times in the week. The deviation is the surplus (per week), so the number of idle times minus 5. If a student has on the five days 1, 0, 2, 3, 1 idle times, respectively, the total number of idle times is 7, and the deviation is 2. Since we calculate quadratically and multiply by the weight, we obtain cost 40 for deviation 2. Hence the (constraint, student) combination adds an amount of 40 to the objective function.

Here we sum the idle times per day. If we only are interested in the idle times on Monday, we can omit the time slot groups for Tuesday, Wednesday, Thursday and Friday.

4.2 Solutions

The Solutions part at the top level is:

```
<Solutions>
  <Solution Id="best">
    <InfeasibilityValue>2</InfeasibilityValue>
    <ObjectiveValue>367</ObjectiveValue>
    <ResourceGroups>
      ...
    </ResourceGroups>
    <Events>
      ...
    </Events>
    <Report>
      ...
    </Report>
  </Solution>
</Solutions>
```

The main thing a solution should do is tell the result of the three scheduling constraints. For this the resource groups are filled with resources of the partitioning constraint, and the events are filled with time slot and resources of the corresponding assigning constraints. Once these aspects are known, all other information can be calculated, and a summary can be added. First the infeasibility value, and the objective value are required. In this example the infeasibility value is 2, indicating that the schedule is considered infeasible. For debugging purposes, we introduce the possibility to give a full report on the violations. This report gives for each resource, resource group, course, and event the constraint that is violated, the deviation and cost of the violation, and a text explaining the violation.

5 Conclusion

The XML format advocated here has been designed in order to better model the complete high school timetabling problem and facilitate data exchange between high school timetabling researchers. We used as input the situation in five different countries, and we think that the presented format can deal with these countries. It is hoped that researchers and practitioners will consider adopting this format in their work and that this will lead to further discussions and improvements to the model. As previously stated we fully expect the number of included constraints to expand and we actively encourage others to contribute datasets in this format. This should not only lead to improvement of the model but also to the development of better and/or more general algorithms.

References

1. D. Abramson, 'Constructing school timetables using simulated annealing: sequential and parallel algorithms', Management Science Vol. **37**, pp. 98-113, (1991).
2. T. Birbas, S. Daskalaki, and E. Housos, 'Timetabling for Greek high schools', Journal of the Operational Research Society **48**, pp. 1191–1200, (1997).
3. E.K. Burke, J.H. Kingston, and P.A. Pepper, 'A standard data format for timetabling instances', in 'Practice and Theory of Automated Timetabling II', E. Burke and M. Carter (Eds.), Lecture Notes in Computer Science **1408**, Springer Verlag, pp. 213–222, (1998).
4. E.K. Burke and S. Petrovic, 'Recent research directions in automated timetabling', European Journal of Operational Research **140**, pp. 266–280, (2002).
5. E.K. Burke, B. McCollum, P. McMullan, and R. Qu, 'Examination timetabling: a new formulation', Proceedings of the Sixth International Conference of the Practice and Theory of Automated Timetabling (PATAT 2006), Brno, pp. 373–375, (2006).
6. M. Carter, G. Laporte, and S. T. Lee, 'Examination timetabling: algorithmic strategies and applications', Journal of the Operational Research Society **47**, pp. 373–383, (1996).
7. M.W. Carter and G. Laporte, 'Recent developments in practical course timetabling', in 'Practice and Theory of Automated Timetabling II', E. Burke and M. Carter (Eds.), Lecture Notes in Computer Science **1408**, Springer Verlag, pp. 3–19, (1998).
8. A. Chand, 'A constraint based generic model for representing complete university timetabling data', Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004), Pittsburgh, pp. 125–148, (2004).
9. T.B. Cooper and J. Kingston, 'The solution of real instances of the timetabling problem', The Computer Journal **36**, pp. 645–653, (1993).
10. A. Cumming and B. Paechter, 'Standard formats for timetabling data', Unpublished discussion session at the First International Conference on the Practice and Theory of Automated Timetabling, Edinburgh, (2005).
11. N. Custers, P. De Causmaecker, P. Demeester, and G. Vanden Berghe, 'Semantic components for timetabling', in 'Practice and Theory of Automated Timetabling V', E. Burke and M. Trick (Eds.), Lecture Notes in Computer Science **3616**, Springer Verlag, pp. 17–33, (2005).

12. P. De Causmaecker, P. Demeester, P. De Pauw-Waterschoot and G. Vanden Berghe, 'Ontology for timetabling', Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000), Konstanz, pp. 481–482, (2000).
13. P. De Causmaecker, P. Demeester, Y. Lu and G. Vanden Berghe, 'Using web standards for timetabling', Proceedings of the Fourth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), Gent, pp. 238–257, (2002).
14. K. Easton, G. L. Nemhauser, and M. A. Trick, 'The travelling tournament problem: description and benchmarks', in 'Principles and Practice of Constraint Programming (CP 2001)', Lecture Notes in Computer Science **2239**, Springer Verlag, pp. 580–585, (2001).
15. O.B. de Gans, 'A computer timetabling system for secondary schools in the Netherlands', European Journal of Operational Research **7**, pp. 175–182, (1981).
16. M. Gröbner, P. Wilke, and S. Büttcher, 'A standard framework for timetabling problems', in 'Practice and Theory of Automated Timetabling IV', E. Burke and P. De Causmaecker (Eds.), Lecture Notes in Computer Science **2740**, Springer Verlag, pp. 24–38, (2003).
17. P. de Haan, R. Landman, G. Post, and H. Ruizenaar, 'A case study for timetabling in a Dutch secondary school', in 'Practice and Theory of Automated Timetabling VI', E. Burke and H. Rudová (Eds.), Lecture Notes in Computer Science **3867**, Springer Verlag, pp. 267–279, (2007).
18. J. H. Kingston, 'Modelling timetabling problems with STTL', in 'Practice and Theory of Automated Timetabling III', E.K. Burke and W. Erben (Eds), Lecture Notes in Computer Science **2079**, Springer Verlag, pp. 309–321, (2001).
19. J.H. Kingston, 'A tiling algorithm for high school timetabling', in 'Practice and Theory of Automated Timetabling V', E. Burke and M. Trick (Eds.), Lecture Notes in Computer Science **3616**, Springer Verlag, pp. 208–225, (2005).
20. F. Kitagawa and H. Ikeda, 'An existential problem of a weight-controlled subset and its application to school timetable construction', Discrete Mathematics **72**, pp. 195–211, (1988).
21. J. Monteiro da Mata, A. Luiz de Senna, and M. Augusto de Andrade, 'Towards a language for the specification of timetabling problems', Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling (PATAT'97), Toronto, pp. 330–333, (1997).
22. K. Nurmi and J. Kyngas, 'A Framework for School Timetabling Problem', Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications, Paris, pp. 386–393, (2007).
23. http://www.cs.nott.ac.uk/∼tec/NRP/.
24. E. Özcan, 'Towards an XML-Based Standard for Timetabling Problems: TTML', Multidisciplinary Scheduling: Theory and Applications (First International Conference, MISTA '03, Nottingham, Selected Papers), pp. 163–185, (2005).
25. Ben Paechter, 'International timetabling competition', `http://www.idsia.ch/Files/ttcomp2002/`, (2003).
26. D. Ranson and S. Ahmadi, 'An extensible modelling framework for the examination timetabling problem', in 'Practice and Theory of Automated Timetabling VI', E. Burke and H. Rudová (Eds.), Lecture Notes in Computer Science **3867**, Springer Verlag, pages 383–393, (2006).
27. L.P. Reis and E. Oliveira, 'A language for specifying complete timetabling problems', in 'Practice and Theory of Automated Timetabling III', E.K. Burke and

W. Erben (Eds), Lecture Notes in Computer Science **2079**, Springer Verlag, pp. 322–341, (2001).
28. A. Schaerf, 'A survey of automated timetabling', Artificial Intelligence Review **13**, No. 2, pp. 87–127, (1999).
29. C. Valouxis and E. Housos, 'Constraint programming approach for school timetabling', Computers & Operations Research **30**, pp. 1555–1572, (2003).
30. D. de Werra, 'An introduction to timetabling', European Journal of Operational Research **19**, pp. 151–162, (1985).
31. D. de Werra, 'On a multiconstrained model for chromatic scheduling', Discrete Applied Mathematics **94**, pp. 171–180, (1999).
32. R.J. Willemen, 'School timetable construction; algorithms and complexity', PhD-thesis, Technical University Eindhoven, The Netherlands, (2002).
33. M. Wright, 'School timetabling using heuristic search', Journal of Operational Research Society **47**, pp. 347–357, (1996).
34. `http://www.w3schools.com/xml/` or `http://en.wikipedia.org/wiki/XML`.