The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)

Luca Di Gaspero[1], Barry McCollum[2], and Andrea Schaerf[1]

[1] DIEGM, University of Udine
via delle Scienze 208, I-33100, Udine, Italy
{l.digaspero|schaerf}@uniud.it

[2] School of Electronics, Electrical Engineering
and Computer Science, Queen's University
SARC Building, Belfast, United Kingdom
b.mccollum@qub.ac.uk

Technical Report

QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1 August 2007

Abstract

Following the success of the First International Timetabling Competition in 2002, the timetabling research community is organising a new competition on this problem (opening August 1st). This new competition will be on three different timetabling problems, and one of the tracks concerns the course timetabling formulation that applies to Italian universities (called Curriculum-based Course Timetabling). The dataset is composed by real-world instances provided by the University of Udine. In this work, we overview the general rules of the competition and we describe in details the problem formulation and the instances proposed for this track.

1 Introduction

Timetabling within Universities has long been recognised as a difficult combinatorial problem of practical relevance. Whether it be timetabling exams or courses, much (human or computing) effort is spent in producing solutions that are both workable and of a high quality (see, e.g., Schaerf, 1999).

Recently, timetabling has also been a subject for a competition. In 2002, the International Metaheuristic Network organised the First International Timetabling Competition (ITC-2002), which attracted 24 feasible submissions from all over the world. Information relating to the problem definition, instances, rules, and solution evaluation of ITC-2002 is available at the webpage: `http://www.idsia.ch/Files/ttcomp2002/`.

Based on a specific problem model proposed for the competition, the formulation contained many of the characteristics found in certain Universities. Artificially generated instances were used in the competition. Thanks to the competition, this formulation has successively become a standard within the research area, and many researchers have used it within their work [1–4]. ITC-2002 therefore has had a positive effect of generating common ground for cross-fertilisation of ideas within research groups in the timetabling community.

The Second International Timetabling Competition (ITC-2007), opening August 1st 2007, follows the main ethos of the first edition, but also aims at advancing upon it in a number of respects.

In this paper, we describe one of the competition tracks, namely the Curriculum-based Course Timetabling, which is under the responsibility of the authors. We present the general rules of ITC-2007, the problem formulation of the track, and the description of the instances. The paper closes with some discussion.

The information presented here can be regarded as the official documentation for the Track 3 and is complemented with the content of the ITC-2007 web site: `http://www.cs.qub.ac.uk/itc2007/`

2 Competition rules

The competition has a set of rules that the participants have to satisfy. The rules of ITC-2007 are in large part taken from those of ITC-2002, but obviously some modifications have been made based on the previous experience. Differences between ITC-2007 rules and ITC-2002 ones will be discussed in Section 6.

Rule 1: This competition seeks to encourage research into automated timetabling methods, and to offer prizes to the most successful methods in particular tracks. It is the spirit of these rules that is important, not the letter. With any set of rules for any competition it is possible to work within the letter of the rules but outside the spirit. The organisers ask that you please don't do this. It's not fair, it's not good science, and it will result in disqualification.

Rule 2: The organisers reserves the right to disqualify any participant from the competition at any time if the participant is determined by the organisers to have worked outside the spirit of the competition rules.

Rule 3: The information on the competition web-site forms part of these rules, but where there is any conflict, this set of numbered rules takes precedence.

Rule 4: The organisers decision is final in any matter.

Rule 5: The organisers reserves the right to change the rules at any time and without notice. Any change of rules will be accompanied by a general email to all participants.

Rule 6: The competition has an opening day, August 1^{st}, 2007, and a deadline, January 25^{st}, 2008, when all submissions must be uploaded.

Rule 7: Participants have to implement an algorithm to tackle the problem on a single processor machine; they can use any programming language.

Rule 8: The goal is to produce feasible timetables, in which a number of hard constraints are satisfied, and to minimize the number of broken soft constraints. Where feasibility can not be reached, infeasible solutions can be provided according to the rules of the specific track. Competitors should refer to the information associated with each track for further specifics.

Rule 9: Instances of different size and type will appear on the web site from the opening day. Two weeks before the deadline more instances will be placed on the web. A third set of datasets will be used to internally rank the top competitors. The datasets are therefore classified as *Early Instances*, *Late Instances* and *Hidden Instances.* Competitors should refer to the information associated with each track for further specifics on datasets.

Rule 10: Participants have to benchmark their machine with the program provided in order to know for how much time they can run their program on their machines.

Rule 11: The algorithms should take as input a problem file in the format described, and produce as output a solution in the allowed CPU time. Obviously the algorithm should not take account of additional knowledge about the instance (e.g. results from previous runs). The same version (and fixed parameters) of the algorithm must be used for all instances. That is, the algorithm should not "know" which instance it is solving - while your particular algorithm might analyse the problem instance and set parameters accordingly, it should not "recognise" the particular instance.

Rule 12: The algorithm can be either deterministic or stochastic. In both cases, participants must be prepared to show that these results are repeatable in the given computer time. In particular the participants that use a stochastic algorithm should make their program in such a way that the exact run that produced each solution submitted can be repeated (by providing the random seed, etc..). They can try several runs to produce each submitted solution (each with the allowed computer time), but they must be able to repeat the run for any solution submitted.

Rule 13: Participants should submit for each instance (*Early* and *Late*) the best score found by their algorithm in the specified computer time, by uploading it onto the web site. Competitors should refer to the information associated with each track for further specifics.

Rule 14: Participants should also submit a concise and clear description of their algorithm, so that in principle others can implement it. A template will be made available one month before the end date for this purpose. This is a fundamental part of a competitors submission.

Rule 15: For each track, as set of 5 finalists will be chosen after the competition deadline. Ordering of competitors will be based on the scores provided on the early and late instances. The actual list will be based on the ranks of solvers on each single instance. The average of the ranks will produce the final place list. More details on how the orderings will be established can be found in Section 2.1here.

Rule 16: Based on the place list a set of top solvers, the finalists, will be asked to provide the executable that will be run and tested by the organisers. The finalists' solver will be rerun by the organisers on all instances (including the *Hidden* ones). It is the responsibility of the competitor to ensure all information is provided to enable the organisers to recreate the solution.

The solver submitted by the finalist should detail ths command-line arguments, input and output file names and, for stochastic solvers only, the random seed. For example (stochastic solver):

```
> my_solver.exe comp05.ctt my_sol5.out 1542955064
```

All parameters should be hard coded in the source code. If appropriate information is nor received or indeed the submitted solutions can not be recreated, another finalist will be chosen from the original competitors.

Rule 17: In addition to considering the results on the early and late datasets, Finalists' eventual place listings will be based on the ranks on each single instance for a set of trials on all instances (including the *Hidden* ones). As with Rule 10, an explanation of the procedures to be used can be found in Section 2.1.

Rule 18: In some circumstances, finalists may be required to show source code to the organisers. This is simply to check that they have stuck to the rules.

Rule 19: Entries from participating organising partners will not be permitted. However, results from participants who choose to work on the problems will be presented for comparison.

2.1 Solution ranking

The following illustrates how the *Finalists* for each track will be chosen. Let m be the total number of early and late instances and k be the number of participants that produce a solution for all m instances.

Let X_{ij} be the result supplied (and verified) by participant i for instance j. Each X_{ij} is a pair (d, s) composed by the distance to feasibility d and the score of the objective function s (remember that results are compared based on d first, and then on s in the case where competitors have the same value for d).

The matrix X of results is transformed into a matrix of ranks by assigning to each X_{ij} a value R_{ij} from 1 to k. That is, for the instance j the supplied values X_{1j}, X_{2j}, ..., X_{kj} are compared with each other and the rank 1 is assigned to the smallest observed value, the rank 2 to the second smallest, and so on up to the rank k, which is assigned to the largest value for the instance j. We use average ranks in case of ties.

Consider the following example with $m = 6$ instances and $k = 7$ participants. Let X be the matrix of Table 1, the computed rank matrix R is reported in Table 2.

For each solver we then define the mean of the ranks for all instances. The finalists of the competition will be the 5 solvers with the lowest mean ranks. In case of a tie for entering the last positions, all the last equal-mean solvers are included in the final (in this case the finalists will be more than 5). In the example, the mean ranks are reported in the last column of Table 2, therefore, in this example, the finalists would be solvers 1, 4, 5, 6, and 7.

Instance	1	2	3	4	5	6
Solver 1	(0,34)	(0,35)	(1,42)	(1,32)	(0,10)	(0,12)
Solver 2	(3,32)	(1,24)	(1,44)	(1,33)	(0,13)	(0,15)
Solver 3	(0,33)	(0,36)	(2,30)	(5,12)	(1,11)	(0,17)
Solver 4	(0,36)	(0,32)	(1,46)	(1,32)	(0,12)	(0,13)
Solver 5	(0,37)	(0,30)	(1,43)	(1,29)	(0,9)	(0,4)
Solver 6	(2,68)	(0,29)	(1,41)	(0,55)	(0,10)	(0,5)
Solver 7	(0,36)	(0,30)	(2,43)	(0,58)	(0,10)	(0,4)

Table 1: Score matrix

Instance	1	2	3	4	5	6	Mean rank
Solver 1	2	5	2	4.5	3	4	3.42
Solver 2	7	7	4	6	6	6	6.00
Solver 3	1	6	6	7	7	7	5.67
Solver 4	3.5	4	5	4.5	5	5	4.50
Solver 5	5	2.5	3	3	1	1.5	2.67
Solver 6	6	1	1	1	3	3	2.50
Solver 7	3.5	2.5	7	2	3	1.5	3.25

Table 2: Rank matrix and mean ranks

The runs of the candidate finalist will be checked with the submitted seed to make sure that they are repeatable. If they are not then another entrant will be chosen for the top five.

For the final the same evaluation process is repeated for the finalists with the following differences:

1. All instances, including *hidden* ones, will be used.

2. The solvers will be run by the organisers, thus the finalists should give us support in the process of compiling and running the solvers.

3. For each instance, we will run 10 independent trials providing the solvers with a sequence of random seeds (the same for all solvers, chosen at random). For each trial, we will proceed as in the previous stage by computing the ranks and averaging them on all trials on all instances.

The final winner of the competition is the solver with the lowest mean rank. In case of a tie, additional trials will be added for all instances until a single winner is found.

3 Problem formulation

The Curriculum-based timetabling problem consists of the weekly scheduling of the lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula published by the University and not on the basis of enrolment data.

This formulation applies to University of Udine (Italy) and to many Italian and indeed International Universities, although it is slightly simplified with respect to the real problem to maintain a certain level of generality.

The problem consists of the following entities:

Days, Timeslots, and Periods. We are given a number of *teaching days* in the week (typically 5 or 6). Each day is split in a fixed number of *timeslots*, which is equal for all days. A *period* is a pair composed of a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.

Courses and Teachers. Each course consists of a fixed *number of lectures* to be scheduled in distinct periods, it is attended by given *number of students*, and is taught by a *teacher*. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.

Rooms. Each *room* has a *capacity*, expressed in terms of number of available seats. All rooms are equally suitable for all courses (if large enough).

Curricula. A *curriculum* is a group of courses such that any pair of courses in the group have students in common. Based on curricula, we have the *conflicts* between courses and other soft constraints.

The solution of the problem is an assignment of a period (day and timeslot) and a room to all lectures of each course.

3.1 Hard constraints

Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods. A violation occurs if a lecture is not scheduled.

RoomOccupancy: Two lectures cannot take place in the same room in the same period. Two lectures in the same room at the same period represent one violation . Any extra lecture in the same period and room counts as one more violation.

Conflicts: Lectures of courses in the same curriculum *or taught by the same teacher* must be all scheduled in different periods. Two conflicting lectures in the same period represent one violation. Three conflicting lectures count as 3 violations: one for each pair.

Availabilities: If the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period. Each lecture in a period unavailable for that course is one violation.

3.2 Soft constraints

RoomCapacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures.*Each student above the capacity counts as 1 point of penalty.*

MinimumWorkingDays: The lectures of each course must be spread into the given minimum number of days. *Each day below the minimum counts as 5 points of penalty.*

CurriculumCompactness: Lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day. *Each isolated lecture in a curriculum counts as 2 points of penalty.*

RoomStability: All lectures of a course should be given in the same room. *Each distinct room used for the lectures of a course, but the first, counts as 1 point of penalty.*

4 Instances and File Formats

There will be 21 instances available: 7 for each set (early, late, and hidden). All instances are real data and come from the University of Udine. For all instances there exists at least one feasible solution (no hard constraint violations), but at present it is not known which is the optimal value for the soft constraints.

In order to model cases in which the number of timeslots is not the same for all days (e.g. Saturday afternoon free), there might be periods unavailable for all courses. Furthermore, for all instances there cannot be two curricula composed by exactly the same courses.

4.1 Input Format

Each instance comes in a single file, and the files are named `comp01.ctt`, `comp02.ctt`, ..., `comp21.ctt`. Every file contains a file header and four sections: courses, rooms, curricula, and constraints. The header provides all scalar values and each section provides the arrays for that specific aspect of the problem. The exact format is shown by the following example:

```
Name: ToyExample
Courses: 4
Rooms: 2
Days: 5
Periods_per_day: 4
Curricula: 2
Constraints: 8

COURSES:
SceCosC Ocra 3 3 30
ArcTec Indaco 3 2 42
TecCos Rosa 5 4 40
Geotec Scarlatti 5 4 18

ROOMS:
A 32
B 50

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec

UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
```

```
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

END.
```

The lines in the header and the following sections are always provided in the precise order as the example. The data in the various sections must be interpreted as follows:

```
Courses: <CourseID> <Teacher> <# Lectures> <MinWorkingDays> <# Students>
Rooms: <RoomID> <Capacity>
Curricula: <CurriculumID> <# Courses> <MemberID> ... <MemberID>
Unavailability_Constraints: <CourseID> <Day> <Day_Period>
```

All `ID`s are strings without blanks. Days and periods start from 0. For example, the constraint `TecCos 3 2` states that course `TecCos` cannot be scheduled in the third (i.e., `2`) period of Thursday (i.e., `3`)

4.2 Output Format

The output also must be provided in a single file such that each line represents the assignment of the room and the timeslot to one lecture (lines can be in any order).

```
<CourseID> <RoomID> <Day> <Day_Period>
```

The following file is a solution for the toy problem provided in the previous section.

```
SceCosC B 3 0
SceCosC A 3 1
SceCosC A 4 0
ArcTec B 0 1
ArcTec B 1 1
ArcTec B 1 2
TecCos B 0 0
TecCos A 0 1
TecCos B 2 2
TecCos B 4 2
TecCos B 4 3
Geotec A 2 2
Geotec A 2 3
Geotec B 3 0
Geotec A 3 1
Geotec A 4 2
```

For example, the first line states that a lecture of `SceCosC` takes place on Thursday (i.e., `3`) in the first period (i.e., `0`) in room `B`.

5 Solution validation

We provide the C++ source code of a solution *validator*, so that the participants can compile it themselves at their machine and also inspect the code. In case it is necessary, executables for various platforms could be provided on request.

The validator takes two command-line arguments: the input file and the output file and it produces on the standard output the evaluation of the solution along with the detailed description of all violations (hard and soft). The very same validator will be used by the organizers for the validation of the scores of the finalists during their final phase.

For example, for the above input and output examples the validator outputs the following messages:

```
[H] Courses ArcTec and TecCos have both a lecture at period 1 (day 0, timeslot 1)
[H] Courses TecCos and Geotec have both a lecture at period 10 (day 2, timeslot 2)
[H] Courses TecCos and Geotec have both a lecture at period 18 (day 4, timeslot 2)
[H] 2 lectures in room B the period 12 (day 3, timeslot 0)
[H] 2 lectures in room A the period 13 (day 3, timeslot 1)
[S(8)] Room A too small for course TecCos the period 1 (day 0, timeslot 1)
[S(5)] The course SceCosC has only 2 days of lecture
[S(5)] The course TecCos has only 3 days of lecture
[S(5)] The course Geotec has only 3 days of lecture
[S(2)] Curriculum Cur1 has an isolated lecture at period 10 (day 2, timeslot 2)
[S(2)] Curriculum Cur1 has an isolated lecture at period 16 (day 4, timeslot 0)
[S(1)] Course SceCosC uses 2 different rooms
[S(1)] Course TecCos uses 2 different rooms
[S(1)] Course Geotec uses 2 different rooms

Violations of Lectures (hard) : 0
Violations of Conflicts (hard) : 3
Violations of Availability (hard) : 0
Violations of RoomOccupation (hard) : 2
Cost of RoomCapacity (soft) : 8
Cost of MinWorkingDays (soft) : 15
Cost of CurriculumCompactness (soft) : 4
Cost of RoomStability (soft) : 3

Summary: Violations = 5, Total Cost = 30
```

In squared brackets we have the type of violation (Hard or Soft) and the penalty (only for soft constraints). For example, line

```
[S(5)] The course SceCosC has only 2 days of lecture
```

indicates that there is a soft constraint violation (i.e., `S`) whose penalty is `5` and the text explains the violation in human readable form.

As another example, the line

```
[H] Courses TecCos and Geotec have both a lecture at period 18 (day 4, timeslot 2)
```

indicates a hard constraint violation (`H`) and reports its explanation. At the end of the outupt there is a summary of the total number of hard constraints violations (i.e., `Violations`) and the total sum of soft constraints (i.e., `Total Cost`), which are the values that will be employed for ranking the solution.

If the output file is not formatted correctly, the validator produces an error message on the standard error and aborts. Conversely, the input is assumed always correct. A validator for the input file, in case the participants want to create new instances for their internal use, is also available and can be provided upon request.

6 Discussion

We now discuss the main characteristics of the competition. We start highlighting the novelties w.r.t. the previous ITC-2002; then, we discuss the key issues underlying the selection the rules; and, we conclude discussing the motivations for the definition of the specific formulation of the problem for our track.

6.1 Differences with ITC-2002

The first innovation consists in being subdivided in three tracks so as to better cover the main formulations of the field of educational timetabling problems. Specifically, the tracks will be on: *Examination timetabling*, *Post Enrolment Course Timetabling* (the evolution of the ITC-2002 formulation), and *Curriculum-based Course Timetabling.*

The second innovation aims at bridging the gap between research and practice: the competition introduces a significant degree of complexity in all tracks so that the new formulations employed are closer (in more aspects, although not all) to those of 'real world' problems [5] and data is coming from the real world.

The third main innovation regards feasibility of solutions. In ITC-2002 only feasible solutions were accepted and it was purposely rather simple to produce a feasible one for all instances. This time, participants that reach only infeasible solutions for some instances can submit their solution, although all instances are guaranteed to have at least a feasible one.

In order to compare different solvers in cases of unfeasible solutions for some instances, we use an evaluation based on ranking of solutions on each instance, rather than on the actual scores (which might be incomparable). Due to this scoring based on rankings, an infeasible solution on one instance does not necessarily prejudice the overall performance of the participant. In addition, instances for the competition have been selected from a larger set of interesting cases, without the limitation of 'easy feasibility'.

Finally, in ITC-2002 the ranking was fully based on the solution provided by the participants. In case of stochastic solvers, this CPU time was to grant the maximum time *for each single trial.* Therefore, the participants could take advantage of what we call the *Mongolian Horde* approach ([7]): "Run as many trials as you can and report only the best of all of them". In ITC-2007, the re-running of finalist solvers on organisers machine (with new seeds) and the use of hidden instances should be able to provide against this approach. Moreover we believe that, in the case of stochastic algorithms, this fosters the design of *robust* solvers.

It is worth mentioning that in order to provide against the excessive use of the Mongolian Horde approach, the competition organisers of ITC-2002 tested the best few algorithms also on unseen instances, and indeed the results were found to be broadly in-line with the known instances.

Apart from the above differences, the rules are basically the same of ITC-2002, as they proved to be adequate to run the previous competition successfully.

6.2 Comments about rules on the running time

One of the main issues about the competitions in general is the running time limits. The reason why it has been decided to have a fixed running time is mainly to remove one degree of variability from the scoring system. We hope that future competitions will take into account in some principled way the trade-off between solution quality and running times.

For the selection of the fixed amount of running time, the key question is which is a realistically feasible running time for the actual timetabling. Given that the timetabling is performed usually a few times a year, one might think that a running time much longer than the 10 minutes (approximatively) granted for the competition would also be reasonable.

In practical cases however, as many researchers have pointed out, the solution of a real case is an interactive process, during which it is necessary to solve a large number of instances. In fact, constraints and objectives are manually adjusted between runs of a working session for one single case (for various reasons: what-if scenarios, last minute changes, etc.). As a rule of thumb, a running time longer than a few minutes makes the process very tiresome and difficult for the human operator.

6.3 Comments about the problem formulation

The actual formulation used at the University of Udine, with respect to the one issued for ITC-2007, has the following extra features:

- A cost component dealing with the lunch break for students: at least one free slot among those around the lunch time.
- The curriculum compactness feature is more complex, and specific patterns are more penalized than others.
- There is a maximum daily student load for each curriculum.
- Some specific lectures must be (must not be) in consecutive periods.
- Rooms might not be available in certain periods, and they must be not suitable for specific lectures.
- If a room is too big for a class, this is also penalized (this is not only for the unpleasant feeling that an empty room provoke, but also to same big rooms for unforeseen activities).
- Weight assigned to soft violations are not complex, and they depend also on the number of students in the curriculum.
- Teacher preferences on periods and rooms are only included as soft constraints.

The only reason for which we have decided to remove all the above features is to maintain a certain degree of generality, so as to do not inflict to the participant the burden to understand all the details of the formulation. The selection of the features to include in the formulation has been based on the aim to balance different types of constraint. Needless to say, if in the future this formulation will prove to be inappropriate (e.g., too simple), some features could be reintroduced for future research.

References

[1] S. Abdullah, E. K. Burke, and B. McCollum. Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In K.F. Doerner, M. Gendreau, P. Greistorfer, W.J. Gutjahr, R.F. Hartl, and M. Reimann, editors, *Metaheuristics - Progress in Complex Systems Optimization*, Springer Operations Research / Computer Science Interfaces Book. 2007. To appear.

[2] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid approach for the university course timetabling problem. *Journal of Scheduling*, 9(5):403–432, 2006.

[3] Luca Di Gaspero and Andrea Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modeling and Algorithms*, 5(1): 65–89, 2006. DOI: 10.1007/s10852-005-9032-z.

[4] Philipp Kostuch. The university course timetabling problem with a three-phase approach. In Edmund Burke and Michael Trick, editors, *Proc. of the 5th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2004), selected papers*, volume 3616 of *Lecture Notes in Computer Science*, pages 109–125, Berlin-Heidelberg, 2005. Springer-Verlag.

[5] Barry McCollum. University timetabling: Bridging the gap between research and practice (invited paper). In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006)*, pages 15–35, Brno, The Czech Republic, 2006.

[6] Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2): 87–127, 1999.

[7] Andrea Schaerf and Luca Di Gaspero. Measurability and reproducibility in timetabling research: State-of-the-art and discussion (invited paper). In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006)*, pages 53–62, Brno, The Czech Republic, 2006.