Modeling and Solving the Thesis Defense Timetabling Problem*

Michele Battistutta[1] Sara Ceschia[2] Fabio De Cesco[1]
Luca Di Gaspero[2] Andrea Schaerf[2]

1. EasyStaff s.r.l., Via Adriatica, 278 - 33030 Campoformido (UD), Italy
{michele,fabio}@easystaff.it

2. DPIA, University of Udine, Via delle Scienze 206, 33100 Udine, Italy
{sara.ceschia,luca.digaspero,andrea.schaerf}@uniud.it

Abstract

The *thesis defense timetabling* problem consists in composing the suitable committee for a set of defense sessions and assigning each graduation candidate to one of the sessions.

In this work, we define the problem formulation that applies to some Italian universities and we provide three alternative solution methods, based on Integer Programming, Constraint Programming and Local Search, respectively. We also develop a principled instance generator, in order to expand the set of available instances.

We perform an experimental analysis and we compare our solvers among themselves, using a testbed composed of both real-world and artificial instances. Even though there is no dominant method, the outcome is that Integer Programming gives the best average results, with Local Search being second, and Constraint Programming last on our testbed. All data is available on the web for verification and future comparisons.

Keywords: Timetabling, Local Search, Simulated Annealing, Integer Programming, Constraint Programming

*This is an Accepted Manuscript of an article published by Taylor & Francis in Journal of the Operational Research Society on January 29th, 2019, available online: http://www.tandfonline.com/doi/abs/10.1080/01605682.2018.1495870

1 Introduction

The thesis defense and the graduation ceremony are unavoidable activities of the management of a university. Large departments may have many students graduating at the same time, and consequently they need to split the graduation procedure into several sessions, with separate committees and possibly running on different days.

The corresponding timetabling problem consists in both assigning each candidate to one session and composing the suitable committees, satisfying various constraints and objectives.

This is an interesting and original NP-hard problem that, up to our knowledge, has been little studied in the relatively-large literature on educational timetabling problems. Its peculiarity, with respect to other timetabling problems, stems from the simultaneous assignment of two types of individuals (students and faculty members), and how they correlate to each other.

In this work, we propose a formulation that models the real-world problem of many Italian universities. We have collected a few instances coming from three Italian universities. Furthermore, in order to have enough instances available to tune and test our solvers, we have also implemented an instance generator that produces realistic cases. These artificial instances have been selected using a Principal Component Analysis (PCA) based on the instance features, in order to produce a good mix of different cases.

We have developed two exact methods based on Integer Programming (IP) and Constraint Programming (CP), using `cplex` (v. 12.7.1) and `Gecode` (v. 5.1.0), respectively. In addition, we have developed a local search method guided by a Simulated Annealing algorithm, which is the one currently in use in the industrial software product.

We compare the different solution methods on the available dataset. From the comparison it is clear that, on the given instances, the IP solver outperforms both local search and CP ones. Nevertheless, in a few cases, the IP method is not able to contribute a good solution in reasonable time, whereas local search is more robust in this respect and always gets to an acceptable solution.

As a recognized good practice (see, e.g., Kendall et al. 2016, Sect. 3.2), all the instances and their solutions are made available through our web application `OptHub` at `https://opthub.uniud.it`. Using `OptHub`, all users can contribute new solutions, which are validated by the system and immediately made publicly available with an associated timestamp. This way, only certified solutions are stored and posted to the community.

2 Problem formulation

The thesis defense timetabling (TDTT) problem consists in the assignment of graduation candidates (students) to different sessions. In addition, for every session the graduation committee must be composed by selecting suitable faculty members, on the basis of their skills and their qualifications. Usually, there are two sessions per day (morning and afternoon) and their duration depends on the number of students assigned. If these sessions are not enough to schedule all the enrolled students, the university adds new ones, typically to be held in parallel to the ones already scheduled.

Students must be assigned exactly to one session, whereas faculty members can participate in more than one committee (or none). The number of members of the committee is not fixed, but it is bound by limits prescribed by university rules and customs. In addition, as already mentioned, there are precise limits on the committee composition in terms of qualified members, such as ensuring the participation of a minimum number of full and associate professors.

The presence of students and faculty members is interconnected because each student has a *supervisor*, that must be part of the committee examining the student. In addition, the committee should include for each student an *opponent* or *challenger*, who is a faculty member with expertise in the area of the thesis, that is designated to ask questions and provide an additional informed evaluation.

Summarizing, the main entities of the problem are:

Sessions: Meetings in which students defend their thesis in front of a committee. Some sessions might overlap in time.

Faculty members: The list of university staff that can be assigned as committee members. They are characterized by their academic level and a list of sessions which they are unable to attend.

Students: The list of candidates of the graduation. For each student, it is given their supervisor and a list of the potential opponents. Supervisors and opponents are faculty members.

As customary, constraints are divided into hard and soft ones. The former must be always satisfied, whereas the latter compose the objective function.

The hard constraints are:

H1. Supervision: For each student, the supervisor must be present at the session assigned to the student.

H2. StudentsPerSession: The number of students assigned to a session must be less than or equal to the given maximum.

H3. OverlappingSessions: In case that two sessions overlap in time, a faculty member can be assigned at most to one of the two committees.

H4. Availability: Each faculty member can be unavailable for some sessions, meaning that she cannot be assigned to sessions for which she expressed her unavailability.

H5. CommitteeComposition: Each committee must be formed respecting a minimum and maximum number of professors for each *academic level*. In detail, there are four levels, which are (starting from the highest): full professor, associate professor, assistant professor, and external teacher (or research associate).

Minimum and maximum values for H5 must be interpreted as minimum and maximum number of members with the given level *or a higher one*. For example, if a problem instance requires a minimum of four associate professors, this constraint can be also fulfilled with two full professors and two associate professors. Consequently, the limits for the lowest academic level represent the limits on the total number of members, as all faculty members have the lowest level or a higher one.

Notice that a minimum number of students per session is not explicitly defined because the number of sessions is already regulated to fit with the number of students that have to graduate.

The soft constraints are:

S1. MultipleDuties: Each time a faculty member sits in more than one committee a penalty is assigned. In order to avoid high workloads, the penalty of the violation is quadratic. That is, if p is the number of presences of a faculty member, the associated penalty is $(p-1)^2$ for $p > 0$, and 0 for $p = 0$.

S2. OpponentPresence: At least one of the suitable opponents of each student must sit in the committee of her corresponding session. Each student assigned to an unsuitable faculty member counts as one violation of this constraint.

The objective function is obtained by summing up the violations of the two soft constraints, each one multiplied by the corresponding weight. The weights are in general selected by the user; in this work, we set them to the values: $w_{\mathsf{S1}} = 1$ and $w_{\mathsf{S2}} = 3$.

In practical cases, the separation in hard and soft constraint can also be altered by the user, who could for example relax some of the hard constraints, by turning them into soft ones and assigning them a weight. For the sake of simplicity, in this work we stick to the classification provided above.

Although the size of actual cases will be discussed in detail in Section 6, we just mention here that they can get quite large, with more than 60 sessions, 500 candidates, and 400 faculty members. The typical size of a committee is between 7 and 10 members. The number of possible opponents for each student in general ranges between 1 and 3. However, in some occasions, no possible opponent is proposed for a student; in this case, we assume that all faculty members are suitable opponents for that student. In such a case, no penalty is added for the soft constraint S2 for that specific student.

3 Mathematical model and computational complexity

In this section we propose a formal model of the problem and we prove the NP-completeness of the underlying decision problem.

3.1 Mathematical model

In order to define the mathematical model of the problem, we start introducing the input data that characterize each instance.

At first we have the following sets and scalar: a set of *candidates*: $\mathcal{C} = \{1, \dots, C\}$, a set of *faculty members* $\mathcal{F} = \{1, \dots, F\}$, a set of *sessions*: $\mathcal{S} = \{1, \dots, S\}$, a set of *academic levels*: $\mathcal{L} = \{1, \dots, L\}$, and a maximum number of candidates for a session $H \in \mathbb{N}$.

Next, we define a few input indicator vectors and matrices:

- $T_{cf} \in \{0, 1\}$: 1 if $f \in \mathcal{F}$ is the supervisor of $c \in \mathcal{C}$, 0 otherwise
- $Q_{cf} \in \{0, 1\}$: 1 if $f \in \mathcal{F}$ is a suitable opponent of $c \in \mathcal{C}$, 0 otherwise
- $L_{fl} \in \{0, 1\}$: 1 if $f \in \mathcal{F}$ has level $l \in \mathcal{L}$ or higher, 0 if she has lower level
- $A_{fs} \in \{0, 1\}$: 1 if $f \in \mathcal{F}$ is available for session $s \in \mathcal{S}$, 0 if unavailable
- $O_{s_1 s_2} \in \{0, 1\}$: 1 if sessions s_1 and s_2 overlap, 0 otherwise
- m_l and M_l: min and max faculties per level l per session

The decision variables concern the assignment of students and faculty members to sessions:

- $x_{cs} \in \{0,1\}$: 1 if student $c \in \mathcal{C}$ is assigned to session $s \in \mathcal{S}$, 0 otherwise
- $y_{fs} \in \{0,1\}$: 1 if faculty member $f \in \mathcal{F}$ is assigned to session $s \in \mathcal{S}$, 0 otherwise

Moreover, in order to express the objective function, and in particular the S2 component, we add the following additional variables:

- $z_{cs} \in \{0,1\}$: 1 if there is no suitable opponent for student $c \in \mathcal{C}$ in session $s \in \mathcal{S}$, 0 otherwise

The model is therefore the following one:

$$\min \quad w_{\mathsf{S1}} \cdot \sum_{f=1}^{F} (\max(0, \sum_{s=1}^{S} y_{fs} - 1))^2 \quad + \quad w_{\mathsf{S2}} \cdot \sum_{c=1}^{C} \sum_{s=1}^{S} x_{cs} \cdot z_{cs} \tag{1}$$

$$\sum_{s=1}^{S} x_{cs} = 1 \qquad \forall c \in \mathcal{C} \tag{2}$$

$$x_{cs} T_{cf} \leq y_{fs} \qquad \forall c \in \mathcal{C}, \forall f \in \mathcal{F}, \forall s \in \mathcal{S} \tag{3}$$

$$\sum_{c=1}^{C} x_{cs} \leq H \qquad \forall s \in \mathcal{S} \tag{4}$$

$$y_{fs_1} + y_{fs_2} \leq 1 \qquad \forall f \in \mathcal{F}, \forall s_1, s_2 \in \mathcal{S} \mid O_{s_1 s_2} = 1 \tag{5}$$

$$y_{fs} \leq A_{fs} \qquad \forall f \in \mathcal{F}, \forall s \in \mathcal{S} \tag{6}$$

$$m_l \leq \sum_{f=1}^{F} y_{fs} L_{fl} \leq M_l \qquad \forall s \in \mathcal{S}, \forall l \in \mathcal{L} \tag{7}$$

$$\sum_{f=1}^{F} Q_{cf} y_{fs} + z_{cs} \geq 1 \qquad \forall s \in \mathcal{S}, \forall c \in \mathcal{C} \tag{8}$$

Constraint (2) states that each student is assigned to exactly one session. Constraints (3)–(7) correspond to constraints H1–H5, respectively. Constraint (8) relates variables z to variables y, stating that if all suitable opponents of a student c are not in session s, then the corresponding variable z_{cs} must necessarily be set to 1.

Regarding the objective function (1), the number of duties of a faculty member $f \in \mathcal{F}$ is given by the expression $\sum_{s=1}^{S} y_{fs}$; therefore the number of *extra* duties is $\max(0, \sum_{s=1}^{S} y_{fs} - 1)$. On overall, the contribution of the component $\mathsf{S1}$ to the objective function is $\sum_{f=1}^{F}(\max(0, \sum_{s=1}^{S} y_{fs} - 1))^2$.

The contribution of the component $\mathsf{S2}$ to the objective function is given by $\sum_{c=1}^{C}\sum_{s=1}^{S} x_{cs} \cdot z_{cs}$. The weights are set to $w_{\mathsf{S1}} = 1$ and $w_{\mathsf{S2}} = 3$, as mentioned before.

3.2 Computational complexity

We now prove that the decision problem underlying TDTT is NP-complete. The fact that the problem is in NP derives from the fact that given a solution Constraints (2)-(8) can be checked in polynomial time. Therefore, we focus on the proof of NP-hardness.

The proof considers only one single session in which all students are assigned to it. This means that the subproblem of selecting the appropriate faculty members for a single session is already NP-complete.

The proof focuses on the relation between students and opponents, and is based on a reduction from the *set covering* problem. Consider an arbitrary set covering instance with universe U, the n sets $S_1, \ldots, S_n \subseteq U$, and an integer c, with $c < n$. It is proven (see Garey & Johnson 1979, problem SP4) that checking if there is a set of c sets that cover the whole U is an NP-complete problem.

We now show that starting from this instance, we can build an instance of TDTT such that it has a 0 cost solution if and only if there is a covering of U composed of c sets.

We associate elements of U to students and sets to the faculty members. In detail, we create a TDTT instance with $m = |U|$ students and $n+1$ faculty members, such that each element of U corresponds to one student, and all students have as supervisor the faculty member with number $n + 1$. For each faculty member i from 1 to n, we assign her to be a suitable opponent for all students corresponding to element in S_i. We set all faculty members available for the session. Finally we set a maximum number of members equal to $c + 1$ (this number is $c + 1$ and not c because one spot is necessary for the supervisor, i.e., the member $n + 1$).

If there is a committee in which all students have an opponent assigned, then the set of members of the committee (excluding the supervisor) corresponds to a selection of the sets that covers all elements in U.

4 Related work

The thesis defense problem belongs to the area of educational timetabling, in which the problem is to assign timeslots and resources to teaching events, satisfying several kinds of constraints. The literature on educational timetabling is very vast, therefore we refer the interested reader to a few surveys for a general introduction to the field (Schaerf 1999, Lewis 2008, Kingston 2013).

In typical educational timetabling problems, events are exams or course lectures (with predefined teachers) and the main resources to be assigned are rooms. Constraints normally involve time overlapping, resource conflicts, and availabilities. The two most studied problems in this area are examination and course timetabling. Recent work on these problems (see, e.g., Goh et al. 2017, Battistutta et al. 2017, Bellio et al. 2016) have obtained several best known results on benchmark instances using Simulated Annealing.

The thesis defense timetabling problem, however, is quite different from the classical timetabling problems, as there is no clear correspondence between the key features of the problem and the notions of event, timeslot, and resource. Indeed, students themselves somewhat represent the events and the *sets of teachers* represent the resources to be assigned; whereas sessions here play the role of timeslots in classical timetabling. The type of constraints and objectives are also very different, as there is no notion in timetabling of concepts corresponding to MultipleDuties and MissingOpponent.

Indeed, the thesis defense timetabling problem is rather an extension of the set covering problem than the graph coloring one, as typical timetabling problems. In fact, the problems more similar to it are the conference scheduling (Stidsen et al. 2018) and student presentation scheduling (Akkan et al. 2016) in which the *grouping* component is more relevant.

A thesis defense timetabling problem has been considered by Bui & Hoang (2012), Huynh et al. (2012), Kochaniková & Rudová (2013). In their work however constraints and objectives are rather different from ours; for example, for the problems considered in all those papers, the committee changes for each single student, so that the order of presentation is also important to minimize the idle time of committee members. On the contrary, in our problem the committee is the same for all the graduation candidates assigned to the same session.

5 Solution methods

We propose both exact (Sect. 5.1) and metaheuristic (Sect. 5.2) methods in order to be able to compare them and give an assessment of the solution

quality.

5.1 Exact methods: IP and CP

The IP method is obtained by a direct implementation of the model described in Section 3.1, in which the cost function is linearized by decomposing it in a sum of products between a constant and an auxiliary binary variable (which is one when the number of duties is greater or equal to a given value), correlated to the y variables by a BigM technique.

For the definition of the CP model, we use the MiniZinc specification language (Nethercote et al. 2007), taking advantage of its modeling capabilities based on integer-valued and set-valued variables.

The CP model makes use of the following decision variables:

- an array of integers, called `StudentSession`, that assigns to each student the selected session,
- an array of integer sets, called `SessionMembers`, that assigns to each session the set of faculty members that compose the committee.

Below are the corresponding MiniZinc definitions, where `Candidates`, `Sessions`, and `FacultyMembers` are input parameters.

```
array [1..Candidates] of var 1..Sessions: StudentSession;
array [1..Sessions] of var set of 1..FacultyMembers: SessionMembers;
```

Using set variables for the committee composition implicitly satisfies the constraint that all members must be distinct.

The constraint about the presence of the supervisor at the defense of each student (H1) is expressed using the built-in `member` set constraint.

```
constraint
  forall (s in 1..Candidates)
     (member(SessionMembers[StudentSession[s]],Supervisor[s]));
```

The maximum number of students per session (H2) is expressed by the `at_most` global constraint.

```
constraint
  forall (s in 1..Sessions)
    (at_most(MaxCandidates,StudentSession,s));
```

This states that for every session `s`, at most `MaxCandidates` elements of the array `StudentSession` can have value `s`.

Simultaneous sessions are stored as an array of pairs (i.e., a matrix with two columns, one for each pair element) such that the overlapping sessions

are in columns 1 and 2. The constraints regarding simultaneous sessions (H3), can then be expressed using the `disjoint` set constraint.

```
constraint
  forall (sim in 1..SimultaneityPairs)
   (disjoint(SessionMembers[SimultaneousSessions[sim,1]],
             SessionMembers[SimultaneousSessions[sim,2]]));
```

The unavailability constraints (H4) are expressed by means of a `if-then-else` conditional expression that adds to the model only the relevant constraints for removing the unavailable members from a session.

```
constraint
   forall (s in 1..Sessions, p in 1..FacultyMembers)
    (if s in Unavailability[p]
     then not (p in SessionMembers[s])
     else true
     endif);
```

Similarly, for expressing the limits about the committee composition (H5) we use an input array of pairs, called `LimitMembersForLevel`, such that the column 1 of the elements of the array contains the minimum and the column 2 contains the maximum. The constraints then are expressed as follows.

```
constraint
   forall (s in 1..Sessions, lv in 1..Levels)
      (((sum (p in 1..FacultyMembers)(AcademicLevel[p] <= lv
                                   /\ p in SessionMembers[s]))
         >= LimitMembersForLevel[lv,1]) /\
       ((sum (p in 1..FacultyMembers)(AcademicLevel[p] <= lv
                                   /\ p in SessionMembers[s]))
         <= LimitMembersForLevel[lv,2]));
```

In order to define the objective function to be minimized, we introduce new decision variables that are functionally related to the main ones.

In particular, for the objective MultipleDuties (S1), we introduce the following variables, where the one that contributes to the objective function is `multiple_use`.

```
array [1..FacultyMembers] of var 0..Sessions: Presences;
var 0..1000: multiple_use;
```

The constraints that link these additional variables to the main ones are the following.

```
constraint
   forall (p in 1..FacultyMembers) (Presences[p] =
        sum (s in 1..Sessions)(p in SessionMembers[s]));
```

```
constraint
   multiple_use = (sum (p in 1..FacultyMembers)((Presences[p] > 0)
                * (Presences[p] - 1) * (Presences[p] - 1)));
```

Similarly, for the objective component OpponentPresence (S2), we introduce the array `NumOpponents` that stores, for each student, the number of potential opponents present at their session. In addition, we introduce a single variable `missing_opponent` that stores the number of students without opponents and is included in the objective function.

```
array [1..Candidates] of var 0..FacultyMembers: NumOpponents;
var 0..Candidates: missing_opponent;
```

These new (redundant) variables are linked to the main ones by the following channeling constraints.

```
constraint
   forall (s in 1..Candidates)
     (NumOpponents[s] = sum (op in Opponents[s])
       (op in SessionMembers[StudentSession[s]]));

constraint
   missing_opponent = sum (s in 1..Candidates)
        (NumOpponents[s] == 0 /\ card(Opponents[s]) > 0);
```

Finally, we introduce the objective function, in which the input weights are set to 1 and 3 respectively as mentioned above.

```
solve minimize WeightOfMultipleUse * multiple_use
            + WeightOfMissingOpponent * missing_opponent;
```

The file with the full model is available along with the instances and the results on the dedicated website mentioned above.

Although the MiniZinc model might, in principle, be directly run with different CP solvers, for a finer solver tuning, the proposed model has also been implemented directly in C++ using the Gecode framework. This allowed us to make full use of all the branching strategies that are available in that framework and are not directly accessible through a MiniZinc specification.

5.2 Metaheuristic method

The experience with the manual solution of the problem suggests that the most constrained parts of the problem are the assignment of students to sessions and the selection of their opponents. On the contrary, the fulfillment

of committee composition constraints is usually rather easy, using additional available members.

For this reason, we decide to use a reduced search space for our local search solver, in which only students (and, consequently, supervisors) and opponents are assigned to sessions. This will possibly lead to a committee that does not satisfy the composition constraints (H5). However, this situation could be recovered at the end of the local search by executing a constructive post-processing step that completes the committee with available members and simultaneously ensures to minimize the number of possible duties.

The main features of our local search algorithm are:

Search space: The search space consists in the assignment of all students to any session, and the selection of one opponent for each of them. The supervisor and the selected opponent are inserted in the committee of the candidate. Infeasible solutions with respect to the hard constraints H2, H3 and H5 are part of the search space and their violation is penalized in the cost function, whereas constraints H1 and H4 are always satisfied. In detail, H1 is satisfied by construction, as the supervisor "follows" the student in the committee, H4 is satisfied by preventing assignments to unavailable sessions both in the initial solution and in the neighbor selection.

Cost function: The cost function is composed by the number of violations of H2, H3 and H5, and the objective function, which is composed by components S1 and S2 multiplied by their nominal weights.

For H2, we count one violation for each student in excess in any session. Similarly, for H5 we count one violation for each faculty member in excess or in defect with respect to the limits. For H3, each faculty member sitting in two overlapping sessions counts for one violation.

The number of violations is multiplied by a high weight (100 in our implementation), so that a single violation costs more than all soft penalties.

Initial solution: The algorithm starts from a solution in which students are assigned to a random session and their opponent is chosen randomly from the list of potential ones. Assignments of students to sessions where their supervisor is unavailable are prevented.

Neighborhood relation: The neighborhood relation is composed by the union of four different basic neighborhoods:

1. Assign the student to a different session and/or change the assigned opponent.
2. Swap the session assignments of two students.
3. Assign to a different session a group of students that are assigned to the same session and have the same supervisor.
4. Swap the sessions for two groups of students: each group is formed by students in the same session and with the same supervisor.

In all four neighborhoods, the supervisor always follows the movement of the corresponding student. In neighborhoods 2–4 the selected opponent remains unchanged, and she also moves to the new session along with the student.

Neighborhoods 3 and 4 were added for a better space exploration since the solver, affected by the constraint of the maximum number of members of a committee and by soft constraint S1 (MultipleDuties), tends to reach local minima in which students with the same supervisor are grouped in the same session.

In general, *swap* moves make smoother changes to the solution without altering the number of students in the sessions, whereas *assignment* moves are more perturbative in that respect.

All four move types have the same probability of being chosen when the algorithm generates a random move.

The local search is driven by a Simulated Annealing (SA) metaheuristic. As customary for SA, the move is always accepted if it is improving or sideways, whereas it is accepted based on time-decreasing exponential distribution in case it is worsening.

In detail, a worsening move is accepted with probability $e^{-\Delta/T}$, where Δ is the difference of total cost induced by the move, and T is the *temperature*. The temperature starts at value T_0, and it is decreased during the search by multiplying it by a value α (with $0 < \alpha < 1$) after a fixed number of samples N has been drawn. The temperature evolves according to the standard geometric cooling scheme of SA.

In order to speed up the early stages of the SA procedure, we use the *cut-off* mechanism (Johnson et al. 1989). To this aim we add a new parameter ρ (with $0 < \rho \leq 1$), representing the fraction of N representing the maximum number of accepted moves at each temperature. That is, the temperature is decreased (i.e., multiplied by α) when the first of the following two conditions occurs: (i) the number of sampled moves reaches N, (ii) the number of accepted moves reaches $\rho \cdot N$.

6 Datasets

We first describe the instances that we have collected from real-world cases and then we discuss the instance generator and the way it has been tuned to generate the instances used in our experimental analysis.

6.1 Real-world instances

The set of real-world instances we consider come from three Italian universities, namely University of Bicocca (Milan), University of Milan, and University of Modena and Reggio Emilia.

For these instances we extract a set of features that we consider relevant in order to tune a specially developed instance generator to create realistic instances. The selected features and their values for the real-world instances are shown in Table 1.

6.2 Instance generator

We developed an instance generator parametrized by the some of the most relevant instance features as identified by running a two-dimensional Principal Component Analysis (PCA) (Jolliffe 2011) on the real-world instances.

Namely, the main generator parameters are the following instance scale quantities: number of sessions, number of students, number of faculty members. Other relevant parameters are the percentage of all faculty members acting as supervisors, the relative unavailability ratio, and the fraction of simultaneous sessions. Given these values, the generator randomly selects the other instance features. The generator pairs a random number of simultaneous sessions for the constraint H3. The committee size and composition are fixed: the committee goes from 7 to 10 members with at least 1 full professor (highest level), 3 associates and 7 other members. We choose to use these fixed values since they were common for the real cases that we analyzed.

Regarding the data on supervisors and opponents, first we decide to assign either 2 or 3 to each student, according to what happens in the real cases. In order to select them in a realistic way, all faculty members are assigned to an area of expertise, which are numbered from one to a fixed value. Once the supervisor is randomly selected, the potential opponents are selected among the faculty members of the area of the supervisor and the two adjacent ones. If not enough members, besides the supervisor, exist in these three areas, the student is left with less suggested opponents. In the extreme case, if a student has no suggested opponents, we leave the set empty meaning that the opponent can be anyone in the committee.

Table 1: Features of the real-world instances.

Features / Instance	01	02	03	04	05	06	07	08	09	10
Sessions	16	5	18	18	37	10	8	6	15	31
Students	143	34	144	171	442	79	102	55	173	209
Faculty Members	155	79	77	101	296	99	66	67	280	435
Number Of Supervisors	66	25	58	64	171	51	36	31	59	109
Opponents Per Students	1.0	1.97	2.98	3.0	1.0	3.0	1.0	1.0	1.02	1.0
Max Students Per Supervisor	7	3	9	6	11	5	6	4	15	7
Max Candidates	9	10	8	10	12	8	13	10	14	7
Unavailability Constraints Per Member	4.25	1.15	6.26	5.86	10.91	3.86	1.18	1.37	0.16	9.0
Unavailable Ratio	0.27	0.23	0.35	0.33	0.29	0.39	0.15	0.23	0.01	0.29
Faculty Members Per Session	9.69	15.8	4.28	5.61	8.0	9.9	8.25	11.17	18.67	14.03
Percentage Of Supervisors	0.43	0.32	0.75	0.63	0.58	0.52	0.55	0.46	0.21	0.25
Students Per Faculty Member	0.92	0.43	1.87	1.69	1.49	0.8	1.55	0.82	0.62	0.48
Students Per Supervisor	2.17	1.36	2.48	2.67	2.58	1.55	2.83	1.77	2.93	1.92
Students Per Sessions	8.94	6.8	8.0	9.5	11.95	7.9	12.75	9.17	11.53	6.74

Features / Instance	11	12	13	14	15	16	17	18	19	20	21
Sessions	16	16	35	45	66	13	2	18	7	14	4
Students	164	222	430	551	428	124	18	172	69	146	58
Faculty Members	287	322	435	454	454	84	76	80	79	81	31
Number Of Supervisors	79	91	165	189	173	59	15	63	46	60	20
Opponents Per Students	1.01	1.0	1.0	1.0	1.01	2.99	3.0	3.0	3.0	3.0	1.0
Max Students Per Supervisor	11	8	12	13	20	5	3	8	4	6	10
Max Candidates	12	14	13	13	7	10	10	11	11	11	15
Unavailability Constraints Per Member	0.03	0.27	10.67	12.51	18.13	5.27	0.8	6.95	2.37	5.79	1.42
Unavailable Ratio	0.0	0.02	0.3	0.28	0.27	0.41	0.4	0.39	0.34	0.41	0.35
Faculty Members Per Session	17.94	20.12	12.43	10.09	6.88	6.46	38.0	4.44	11.29	5.79	7.75
Percentage Of Supervisors	0.28	0.28	0.38	0.42	0.38	0.7	0.2	0.79	0.58	0.74	0.65
Students Per Faculty Member	0.57	0.69	0.99	1.21	0.94	1.48	0.24	2.15	0.87	1.8	1.87
Students Per Supervisor	2.08	2.44	2.61	2.92	2.47	2.1	1.2	2.73	1.5	2.43	2.9
Students Per Sessions	10.25	13.88	12.29	12.24	6.48	9.54	9.0	9.56	9.86	10.43	14.5

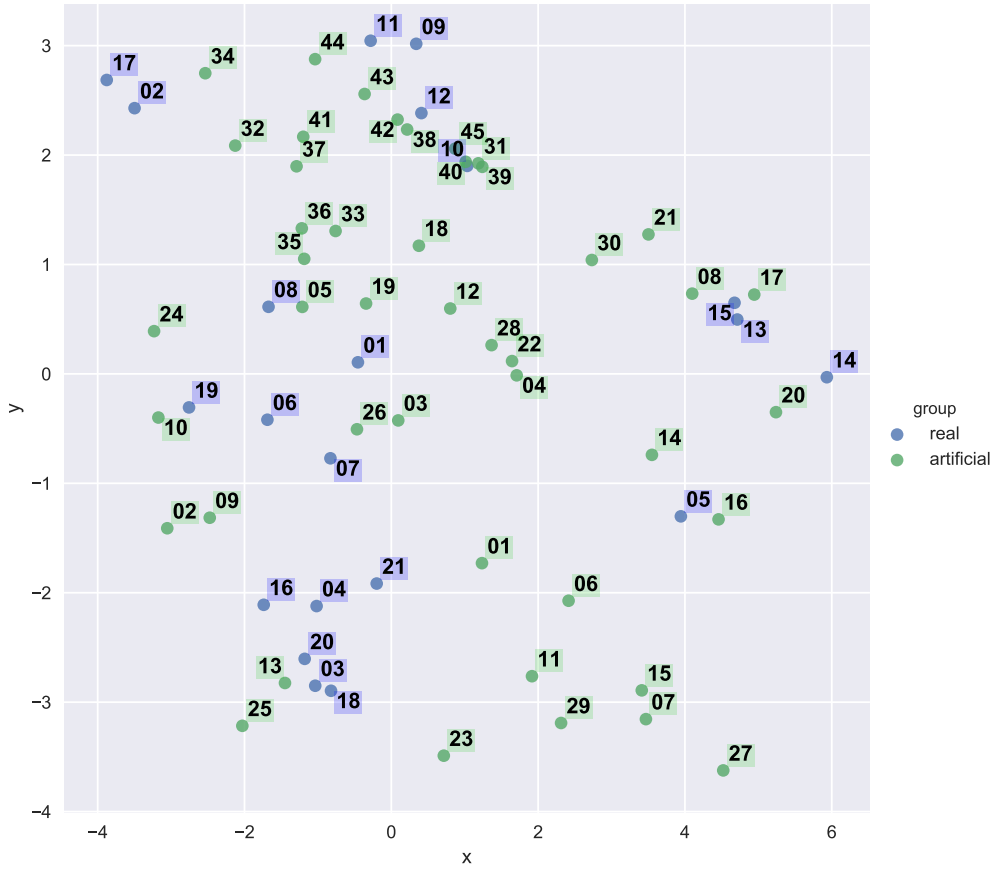

Figure 1: Representation of the instances in the PCA plane.

6.3 Instance selection

In order to obtain instances that although structurally similar to the original ones, are different enough, we employ the results of the PCA run on the real-world instances. The PCA combines the correlated features of the instances and allows us to reduce the number of dimensions to analyze (in our case just two). In particular, each principal component is a linear combination of features and they are extracted in order of importance.

We apply the linear model to map the generated instances on the $PC_1 \times PC_2$ plane. This way we were able to inspect the instance similarity and discard the instances that are too similar to others. We have generated 45 instances, and Figure 1 shows the placement of real-world and artificial instances in the $PC_1 \times PC_2$ plane.

7 Experimental analysis

The whole code is written in C++ and compiled using `gcc` v. 5.4.0. The IP solver is implemented using the C++ API provided by the cplex/concert technology, whereas the CP solver has been implemented using Gecode 5.1.0.

The parameter tuning, described in the next section, has been performed using the tool JSON2RUN (Urli 2013) that implements the F-Race procedure (Birattari et al. 2010).

All experiments ran on an Ubuntu Linux 15.04 machine with 4 Intel® i7-4770 (3.40 GHz) physical cores, hyper-threaded to 8 virtual cores. A single virtual core has been dedicated to each experiment.

For the sake of reproducibility, we fix the time limit of each experiment to 542 seconds, which corresponds to the time granted on our PC for the participation to ITC-2011, the third timetabling competition (Post et al. 2016).

7.1 Tuning

Simulated Annealing has several control parameters: the cooling rate (α), the number of neighbors sampled at each temperature (N), the starting and final temperatures (T_0 and T_{min}), and the cut-off ratio (ρ). In order to find the best configuration of these parameters using a statistically-principled approach, we resort to F-Race with a 95% statistical confidence (p-value = 0.05).

With the aim of equalizing approximately the running times for all different configurations, we fix the total number of iterations $I = 1.51^8$ and compute the parameter N from the others so that the total I is always the same. This specific value of I is set to fix the average running time to 542 seconds.

Preliminary results demonstrate that our solution method is not sensitive to small variations of the cooling rate, thus we decided to set $\alpha = 0.99$ and focus on the other control parameters T_0 T_{min}, and ρ.

We tested 30 different configurations generated according to the *Hammersley point set* (Hammersley & Handscomb 1964) for the ranges whose bounds are $T_0 = [1, 10]$, $T_{min} = [.05, 0.5]$ and $\rho = [0.05, 0.2]$. One configuration turned out to be statistically superior to all the others, and its values are reported in Table 2. All the following experiments have been performed using these values for the parameters.

Only the artificial instances have been used for the tuning phase, so as to separate, as customary, training instances (i.e., the artificial ones) and validation ones (i.e., the real-world ones).

Table 2: Parameter configuration for Simulated Annealing.

Parameter	Description	Value
T_0	Start temperature	1.94
T_{min}	Final temperature	0.106
ρ	Fraction of N before cut-off	0.117
N	Max iterations at each temperature	35788
α	cooling rate	0.99

Table 3: Best branching strategy for the CP solver.

Branching strategy	Value
`StudentSession` variable	*maximum accumulated failure count*
`StudentSession` value	*minimum value*
`SessionMembers` variable	*minimum size*
`SessionMembers` value	*median value*
Branching order	`SessionMembers` first

The CP model implemented in Gecode has been tuned with respect to the branching strategies (i.e., variable and value selection heuristics) of the two families of decision variables, namely the `StudentSession` integer variables and the `SessionMembers` set variables. The variable selection heuristics tested deal with some measure of variable constrainedness. Namely, besides the *first unassigned* heuristic that was employed as a baseline, the heuristics tested are *maximum degree*, *maximum accumulated failure count*, *maximum activity*, and *minimum size*. The integer value selection heuristics employed in the experimentation are: *min*, *median*, *max*, *split min*, *split max*. As for the set variables, the *min*, *med*, and *max* value selection heuristics were employed.

Moreover, the branching order has been subject to test. In fact, we tested two variants of the model, the first one branching first on the `StudentSession` integer variables and then on the `SessionMembers` set variables, and the second with the role of the two array of variables reversed (i.e., `SessionMembers` first and `StudentSession` second).

Summing up, we have 5 options for variable selection for each family of decision variables, 5 and 3 options for value selection of the two families, and branching orders. Since the CP solver is deterministic, the $5 \times 5 \times 5 \times 3 \times 2 = 750$ branching variants have been evaluated by means of a F-Race procedure with a 95% confidence level, using a single repetition per instance of a shorter run. The best setting found by F-Race is reported in Table 3.

Regarding the IP method, `cplex` has a large number of parameters (over

100) that could be tuned. We decided to consider only the use of heuristics, and in particular to focus on two classical ones, namely *Local Branching* (Fischetti & Lodi 2003) and *Relaxation Induced Neighborhood Search* (RINS) (Danna et al. 2005).

The results of the experiments on the corresponding parameters showed that, besides a few combinations that performed worse, there is no relevant effect on the results with respect to the default behavior. Therefore, for the comparison we used the default configuration of `cplex`.

7.2 Comparison of results

Tables 4 and 5 show best cost, average cost, and standard deviation of the results of our local search solver, for 30 repetitions, obtained with the parameter configuration described in Section 7.1, running for 542 seconds (ITC-2011 time limit).

The tables report also the cost of the deterministic solution obtained by CP and IP within the same time limit and the running time of those solvers. The † in the time columns of the complete solvers indicate that the solvers run up to the timeout. The ‡ symbol in the cost columns means that the instance has been proven unsatisfiable, whereas the dashes indicate that no feasible solution has been found by the algorithm within the allotted time. The * symbol close to the cost found by the complete solvers indicates that the solution has been proven optimal by that solver.

The results show that, apart the CP method that performs quite poorly on this problem, even though the IP solver works better than local search in average, there is no dominant method between these two.

As intrinsic in the nature of its search methods, IP may find the optimal solution, but sometimes fails to get even to a feasible one. Conversely, being heuristic, local search is more robust but frequently fail to get to the optimal value.

Indeed, on the artificial testbed the local search is able to find good quality solutions, in particular in 8 cases the method is able to find a solution whereas the IP method could not reach any within the timeout. Conversely, the IP method clearly outperforms the local search on the real-world testbed, being able to consistently find solutions of lower cost in all but two cases. Moreover, it is able to prove solution optimality in all but one case. Also on the artificial testbed, the IP method is able to find and prove optimality on 30 out of 45 instances.

Given these results, it is possible to say that the generated instances are challenging and they are not easily solved with a single method.

Table 4: Comparison of the solution methods on the real testbed.

	Local search			CP		IP	
Instance	best	avg	stdev	cost	time	cost	time
tdtt-real01	57	59.33	0.96	325	†	**36***	88.48
tdtt-real02	**0**	0.00	0.00	48	†	**0***	0.06
tdtt-real03	52	52.33	0.66	654	†	**49***	18.18
tdtt-real04	44	44.27	0.52	670	†	**25***	427.47
tdtt-real05	88	88.00	0.00	—	†	**0***	13.36
tdtt-real06	1	1.57	0.63	107	†	**0***	0.79
tdtt-real07	33	38.10	2.52	—	†	**31***	457.14
tdtt-real08	21	21.00	0.00	37	†	**17***	0.04
tdtt-real09	**42**	42.00	—	527	†	62	†
tdtt-real10	43	47.13	2.01	—	†	**40**	†
tdtt-real11	**13**	14.41	0.72	463	†	14	†
tdtt-real12	**34**	42.50	12.02	—	†	64	†
tdtt-real13	80	80.00	0.00	—	†	**0***	5.18
tdtt-real14	124	126.80	1.45	—	†	**1***	19.48
tdtt-real15	**306**	319.13	6.37	—	†	—	†
tdtt-real16	15	15.77	0.82	—	†	**7***	108.22
tdtt-real17	**21**	21.00	0.00	**21**	0.09	**21***	†
tdtt-real18	48	48.67	0.61	531	†	**46***	23.51
tdtt-real19	**0**	4.80	2.66	146	†	**0***	0.30
tdtt-real20	20	22.23	0.97	351	†	**17***	135.51
tdtt-real21	109	109.00	0.00	**9**	†	**9***	0.01

As a final comment about the CP method, we notice that we tried to improve it by using a different model, based on the same underlying assumptions of the local search one (i.e., working on a different search space). The results with this CP model were in general worse than those obtained by the CP model presented in the paper and therefore are omitted.

8 Discussion and conclusions

We have modeled and solved a timetabling problem that, up to our knowledge, has not yet been formalized in the scientific literature.

For this new problem, we have collected 21 real-world cases and developed an instance generator in order to abstain from overtuning on such a relatively small number of available instances. Random instances have been generated and selected based on a principled statistical method (PCA). All instances and solutions are available at `https://opthub.uniud.it`, for inspection and future comparisons.

For tackling the problem we have developed a family of exact and metaheuristic solvers. Namely we developed a Simulated Annealing algorithm, an IP model implemented in `cplex` and a CP model specified in MiniZinc and then directly implemented in Gecode. The experimental analysis shows that

Table 5: Comparison of the solution methods on the artificial testbed.

	Local search			CP		IP	
Instance	best	avg	stdev	cost	time	cost	time
tdtt-art01	88	90.03	1.38	—	†	**82***	446.10
tdtt-art02	17	17.00	0.00	70	†	**16***	0.05
tdtt-art03	3	5.00	1.44	—	†	**1***	52.61
tdtt-art04	6	9.13	1.76	—	†	**0***	44.46
tdtt-art05	**0**	0.60	0.56	465	†	**0***	1.86
tdtt-art06	**110**	113.30	2.05	—	†	—	†
tdtt-art07	**201**	210.20	4.20	—	†	—	†
tdtt-art08	3	6.47	1.93	—	†	**0***	74.14
tdtt-art09	13	14.03	0.76	—	†	**8***	0.22
tdtt-art10	16	16.00	0.00	73	†	**12***	0.10
tdtt-art11	**215**	221.13	3.01	3598	†	—	†
tdtt-art12	**0**	1.20	0.71	—	†	**0***	10.68
tdtt-art13	83	83.03	0.18	355	†	**78***	2.37
tdtt-art14	**36**	42.07	2.98	—	†	—	†
tdtt-art15	**169**	172.97	2.28	—	†	—	†
tdtt-art16	**63**	69.60	2.66	—	†	—	†
tdtt-art17	2	7.50	2.49	—	†	**0***	530.22
tdtt-art18	**0**	0.93	0.58	834	†	**0***	14.39
tdtt-art19	**0**	1.57	1.17	—	†	**0***	23.51
tdtt-art20	**12**	26.23	5.14	—	†	46	†
tdtt-art21	**0**	3.40	1.65	—	†	**0***	509.68
tdtt-art22	3	5.27	1.76	—	†	**0***	42.46
tdtt-art23	180	183.97	2.28	—	†	**171***	344.39
tdtt-art24	**0**	0.10	0.31	50	†	**0***	0.05
tdtt-art25	—	—	—	—‡	0.08	—‡	0.00
tdtt-art26	26	27.20	0.96	643	†	**14***	42.53
tdtt-art27	**219**	232.37	5.30	—	†	—	†
tdtt-art28	2	4.63	2.03	—	†	**0***	111.38
tdtt-art29	**203**	210.60	3.64	2889	†	—	†
tdtt-art30	**0**	1.73	1.23	—	†	**0***	29.15
tdtt-art31	35	36.53	0.78	—	†	**26**	†
tdtt-art32	**0**	0.37	0.49	—	†	**0***	0.57
tdtt-art33	1	2.03	0.67	—	†	**0***	2.26
tdtt-art34	1	1.00	0.00	—	†	**0***	0.44
tdtt-art35	**0**	0.27	0.45	381	†	**0***	1.43
tdtt-art36	**0**	0.43	0.50	—	†	**0***	1.63
tdtt-art37	4	4.23	0.43	—	†	**0***	0.44
tdtt-art38	16	16.27	0.52	828	†	**1***	23.91
tdtt-art39	24	25.57	0.94	—	†	**11**	†
tdtt-art40	38	38.67	0.76	—	†	**9**	†
tdtt-art41	3	3.57	0.63	—	†	**0***	22.53
tdtt-art42	19	19.03	0.18	701	†	**4**	†
tdtt-art43	1	1.73	0.58	633	†	**0***	6.78
tdtt-art44	1	1.00	0.00	276	†	**0***	0.46
tdtt-art45	32	32.10	0.31	—	†	**7**	†

both IP and local search perform rather well on the (different) instances of the testbeds. Conversely, the CP method features performances that are far from its two other competitors.

The system, equipped with the local search engine, is currently in use in three Italian universities. According to the personnel involved, it has reduced the time spent in the overall procedure by several man-days for each graduation period (which takes place 3-4 times a year) . This abatement is due to both the optimization module and the web-based interface for collecting the input data and publishing the results. In addition, the system has improved the fairness of the solution in terms of multiple duties of the members, reducing the number of complaints significantly.

For the future, we plan to investigate the management of the thesis defense procedure in other universities (also outside Italy), in order to design a more general formulation. Moreover we will try to investigate which are the instance features that explain the (huge) difference in performances of the IP method on the homogeneous testbed of artificial instances.

Another future development is to collect more real-life instances, and to use them also for the improvement of the generator, in such a way that it can create more realistic instances.

Acknowledgments

We would like to thank Professors Gallace and Macchi Cassia of the University of Milano-Bicocca for their support in modeling the problem and collecting the data.

References

Akkan, C., Külünk, M. E. & Koçaş, C. (2016), 'Finding robust timetables for project presentations of student teams', *European Journal of Operational Research* **249**(2), 560 – 576. doi:10.1016/j.ejor.2015.08.047.

Battistutta, M., Schaerf, A. & Urli, T. (2017), 'Feature-based tuning of single-stage simulated annealing for examination timetabling', *Annals of Operations Research* **252**(2), 239–254. doi:10.1007/s10479-015-2061-8.

Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A. & Urli, T. (2016), 'Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem', *Computers & Operations Research* **65**, 83–92. doi:10.1016/j.cor.2015.07.002.

Birattari, M., Yuan, Z., Balaprakash, P. & Stützle, T. (2010), F-race and iterated F-race: An overview, *in* 'Experimental methods for the analysis of optimization algorithms', Springer, Berlin, pp. 311–336.

Bui, L. T. & Hoang, V. (2012), A multi-objective approach for master's thesis committees scheduling using DMEA, *in* 'Asia-Pacific Conference on Simulated Evolution and Learning', Springer, pp. 450–459.

Danna, E., Rothberg, E. & Le Pape, C. (2005), 'Exploring relaxation induced neighborhoods to improve mip solutions', *Mathematical Programming* **102**(1), 71–90. doi:10.1007/s10107-004-0518-7.

Fischetti, M. & Lodi, A. (2003), 'Local branching', *Mathematical programming* **98**(1-3), 23–47. doi:10.1007/s10107-003-0395-5.

Garey, M. R. & Johnson, D. S. (1979), *Computers and Intractability—A guide to NP-completeness*, W.H. Freeman and Company, San Francisco.

Goh, S. L., Kendall, G. & Sabar, N. R. (2017), 'Improved local search approaches to solve the post enrolment course timetabling problem', *European Journal of Operational Research* **261**(1), 17 – 29. doi:10.1016/j.ejor.2017.01.040.

Hammersley, J. M. & Handscomb, D. C. (1964), *Monte Carlo methods*, Chapman and Hall, London.

Huynh, T. T. B., Pham, Q. D. & Pham, D. D. (2012), Genetic algorithm for solving the master thesis timetabling problem with multiple objectives, *in* 'Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on', IEEE, pp. 74–79.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. & Schevon, C. (1989), 'Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning', *Operations Research* **37**(6), 865–892. doi:10.1287/opre.37.6.865.

Jolliffe, I. (2011), *Principal Component Analysis*, Springer, Berlin, Heidelberg, pp. 1094–1096. doi:10.1007/978-3-642-04898-2_455.

Kendall, G., Bai, R., Błazewicz, J., De Causmaecker, P., Gendreau, M., John, R., Li, J., McCollum, B., Pesch, E., Qu, R., Sabar, N., Vanden Berghe, G. & Yee, A. (2016), 'Good laboratory practice for optimization research', *Journal of the Operational Research Society* **67**(4), 676–689. doi:10.1057/jors.2015.77.

Kingston, J. H. (2013), Educational timetabling, *in* A. S. Uyar, E. Ozcan & N. Urquhart, eds, 'Automated Scheduling and Planning', Vol. 505 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, pp. 91–108.

Kochaniková, B. & Rudová, H. (2013), Student scheduling for bachelor state examinations, *in* 'Proc. of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-13)', Ghent, Belgium, pp. 762–766.

Lewis, R. (2008), 'A survey of metaheuristic-based techniques for university timetabling problems', *OR Spectrum* **30**(1), 167–190. doi:10.1007/s00291-007-0097-0.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J. & Tack, G. (2007), MiniZinc: Towards a standard CP modelling language, *in* 'Principles and Practice of Constraint Programming (CP 2007)', Christian Bessiere Ed., Providence, RI, USA, pp. 529–543.

Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B. & Schaerf, A. (2016), 'The third international timetabling competition', *Annals of Operations Research* **239**(1), 69–75. doi:10.1007/s10479-013-1340-5.

Schaerf, A. (1999), 'A survey of automated timetabling', *Artificial Intelligence Review* **13**(2), 87–127. doi:10.1023/A:1006576209967.

Stidsen, T., Pisinger, D. & Vigo, D. (2018), 'Scheduling euro-k conferences', *European Journal of Operational Research* **270**(3), 1138 – 1147. doi:10.1016/j.ejor.2017.10.015.

Urli, T. (2013), 'json2run: a tool for experiment design & analysis', *CoRR* **abs/1305.1112**.