Local search and constraint programming for a real-world examination timetabling problem*

Michele Battistutta[1], Sara Ceschia[2][0000−0003−1191−1929], Fabio De Cesco[1], Luca Di Gaspero[2][0000−0003−0299−6086], Andrea Schaerf[2][0000−0001−6965−0536], Elena Topan[1]

[1] EasyStaff srl, Via Adriatica, 278, 33030 Campoformido (UD), Italy
{michele,fabio,elena}@easystaff.it

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

Abstract. We investigate the examination timetabling problem in the context of Italian universities. The outcome is the definition of a general problem that can be applied to a large set of universities, but is quite different in many aspects from the classical versions proposed in the literature.
We propose both a metaheuristic approach based on Simulated Annealing and a Constraint Programming model in MiniZinc. We compare the results of the metaheuristic approach (properly tuned) with the available MiniZinc back-ends on a large set of diverse real-world instances.

Keywords: Examination Timetabling; Simulated Annealing; MiniZinc.

1 Introduction

Examination timetabling (ETT) is one of the classical problems that every university has to deal with on a regular basis. Many formulations of the ETT problem have been proposed in the literature, some of which have received considerable attention [5, 12].

We propose a novel formulation of ETT, which applies to Italian universities. This formulation is quite different from the ones proposed in the literature [15, 16], as it involves many specific constraints and objectives. For example, some exams are composed by separate written and oral part, which must be scheduled at suitable distance and have different overlap acceptability levels in relation to other exams. In addition, the same exam might be repeated more than once in a session, with prescribed minimal distances among rounds. As another quite distinctive feature, exams might require multiple rooms, typically in exclusive use.

In this work, we propose a metaheuristic approach based on a tailored neighborhood structure, a Simulated Annealing procedure, and a statistically-principled

* The final authenticated version of this paper is available online at https://doi.org/10.1007/978-3-030-58942-4_5

parameter tuning. The main motivation for the choice of Simulated Annealing is that it already turned out to be capable of obtaining state-of-the-art results in many educational timetabling problems (see, e.g., [1, 2, 6]).

We also developed a MiniZinc model that allows us to compare the results of the metaheuristic, by testing many available back-ends on this challenging problem.

The outcome is that Simulated Annealing is able to solve real-world instances to a good quality. Conversely, most instances are currently beyond the reach of exact Constraint Programming methods.

As a byproduct of this research, we are collecting many real-world instances, which are made available to the community for future comparisons, and could potentially become a new benchmark. Instances, solutions, and the MiniZinc model are available at `https://bitbucket.org/satt/ExamTimetablingUniudData`. The repository includes also a Python validator that checks the cost of a solution, so as to provide against possible misunderstanding about the constraints and the objectives.

2 Problem Formulation

Our problem consists of scheduling an examination session at the end of a semester for a university campus. The problem is based on the following entities:

Courses, Exams, & Events: For each course, we have to schedule one or more exams within the session. Each exam might be a single event (either written or oral) or composed by two events, identified as the written part (first) and the oral part (second), to be handed out in this strict order.

Rooms & Roomsets: Some events require one or more rooms, others do not, as they take place in teacher's office or in external rooms. Rooms are classified as small, medium, or large, and for each written event we set the number and the type of rooms requested (mixed requests are not considered). Due to logistic issues, not all combinations of homogeneous rooms can be assigned to a single event. The available ones, called *roomsets*, are explicitly listed in the input data. Oral events might require at most one room (of any size).

Days, Timeslots, & Periods: The session is divided in days and each day in divided in timeslots, with the same number of timeslots for each day. Each pair day/timeslot represents a period of the session.

Curricula: A curriculum is a set of courses that have students in common, which might enroll in the corresponding exams. The set of courses of a curriculum is split into *primary* courses, that are the ones taught in the current semester, and the *secondary* ones, that have been taught in the previous semester, but such that some students might still have to undertake them. The level of conflict between primary and secondary exams of a curriculum varies, as detailed below.

COURSES

Name	Teacher	#Exams	Exam Distance	Exam Type	$\mathcal{W}/\mathcal{O}$ Distance	Location
Databases	Andrea	2	10	$\mathcal{W}+\mathcal{O}$	(4,8)	2 Large
Algorithms	Luca	2	10	$\mathcal{W}$	—	1 Large
Program.	Elena	2	8	$\mathcal{W}$	—	2 Large
Oper. Res.	Sara	2	8	$\mathcal{W}+\mathcal{O}$	(1,1)	1 Small + 1 Small
Analytics	Michele	1	—	$\mathcal{O}$	—	—

ROOMS

Name	Type
A	Large
B	Large
C	Large
G	Small
H	Small

ROOMSETS

Roomset	Size	Type
{A, B}	2	Large
{A, C}	2	Large

PERIODS

20

CURRICULA

Name	Primary	Secondary
Man. Eng.	Databases	Analytics
	Oper. Res.	
Ele. Eng.	Databases	Oper. Res.
	Algorithms	Analytics

Fig. 1. A toy instance.

The problem consists in assigning a period and a *location* (a room, a roomset, or nothing) to each event, satisfying the hard and soft constraints explained in the following paragraphs.

All the data representing a toy instance is shown in Fig. 1. The table COURSES shows the structure of the courses, with teacher, number of exams, distance between exams, exam type ($\mathcal{W}$ for written, $\mathcal{O}$ for oral), and location requested. The other tables show the respective features, including the available roomsets. Preferences and constraints are not shown for the sake of brevity.

We propose a possible solution in Fig. 2. We can see that the 5 courses are "exploded" into 13 events. Each single event has its own conflict and distance constraints, as discussed in the following paragraphs.

The hard constraints are the following:

H1. RoomRequest: for each written event, type and number of the rooms assigned must be correct; for oral exams, a single room, of any type, must be assigned, if requested.

H2. RoomOccupation: in one period, a room can be used by at most one event.

H3. HardConflicts: Two events in *hard conflict* cannot be scheduled in the same period. Two events are in hard conflict in the following cases:
- They are part of courses that are both primary courses of one curriculum.
- They have the same teacher.
- There is an explicit constraint stating that the overlap of the two events is forbidden.

H4. Precedences: When two events have a *precedence* constraint, the first must be scheduled strictly before the second. Two events have a precedence constraint in the following cases:
- They are part of two exams of the same course.

Solution

Event				Assignment	
Course	Exam	Part	ID	Period	Rooms
Databases	#1	$\mathcal{W}$	0	2	A, B
		$\mathcal{O}$	1	5	—
	#2	$\mathcal{W}$	2	12	A, B
		$\mathcal{O}$	3	15	—
Algorithms	#1	$\mathcal{W}$	4	0	B
	#2	$\mathcal{W}$	5	10	A
Programming	#1	$\mathcal{W}$	6	0	A, C
	#2	$\mathcal{W}$	7	17	A, B
Oper. Res.	#1	$\mathcal{W}$	8	6	H
		$\mathcal{O}$	9	7	G
	#2	$\mathcal{W}$	10	16	H
		$\mathcal{O}$	11	17	G
Analytics	#1	$\mathcal{O}$	12	8	—

Fig. 2. A solution for the toy instance.

– They are part of the same exam (written and oral).

H5. Unavailabilities: An event might be explicitly stated as unavailable in a specific period, so that it cannot be assigned to that period. Similarly, a room might be unavailable for a specific period, so that no event can use it in that period.

The objectives (soft constraints) are they following:

S1. SoftConflicts: Two events in *soft conflict* should not be scheduled in the same period. Two events are in soft conflict in the following cases:
- They belong to courses that are in the same curriculum, either as primary and secondary or both as secondary.
- There is an explicit constraint stating that their overlap is undesirable.

S2. Preferences: Like Unavailabilities, preferences between events and periods and between periods and rooms stating the undesirability of an assignment can be expressed explicitly. For periods, it is also possible to state a positive preference for a specific event, so that in presence of preferred periods for an event, all indifferent ones are assumed undesired (and explicitly undesired one are given a larger penalty).

S3. Distances: Among events there might be requested separations in term of periods. Distances can be either *directed*, imposing that one event must precede the other, or *undirected* so that any order is acceptable. The situations that require a distance are the following:
- Same exam (directed): different parts of the same exam have a minimum and a maximum distance, stated specifically for each course (e.g., events 0 and 1 in the example).

- Same course (directed): different exams of the same course must be separated. The separation constraint is applied between the first (or single) part of each of the two exams (e.g., events 0 and 2 in the example).
- Same curriculum (undirected): if two courses belong to the same curriculum, there should be a separation between the exams (as above, for two-part exams, we consider the first one). The amount of separation and the weight for its violation depend on the type of the two (primary or secondary) memberships.
- Additional requests can be added explicitly.

The weight of the violation of the various types of soft conflicts are set by the end-user. Similarly, all distance limits and the corresponding weights are configurable.

We can see that the solution of Fig. 2 has a few soft constraint violations. For example, the distance between the two parts of Databases is 3 periods for both exams, whereas the minimum is 4. Another violation is related to the curriculum Ele. Eng. for which the written part of the first exam of Databases (at period 2) is too close to the written part of the single exam of Algorithms (at period 0), given that the minimum distance for primary/primary exams is assumed to be 6.

3 Related Work

The literature on the examination timetabling problem is rather vast. We refer to Qu *et al* [15] for a relatively up-to-date survey.

Among the large set of proposed formulations, two have received considerable attention in the literature. The first is the classical one from Carter *et al* [5]. This is a very essential version of the problem, in which each exam is a single event and rooms are not considered. Differently from our formulation, which is curriculum-based, in the work of Carter *et al* conflicts are based on student enrollments, so that each student contributes to the penalty of a schedule whenever the exams (s)he takes are too close. Given also that rooms are not involved, the only constraint is HardConflicts (H3) and the only objective is Distances (S3). Distances are penalized in a fixed *exponential* patterns: the cost of scheduling two exams with k common students at distance 1, 2, 3, 4, and 5 periods is $16k$, $8k$, $4k$, $2k$, and k, respectively.

This formulation comes with a challenging dataset of 13 real-world instances form North American universities that are still not solved to proven optimality. The dataset has been subsequently extended and generalized by other authors (see, e.g., [13]). Recent results on this formulation have been obtained by Leite *et al* [10].

The other popular formulation is the one coming from Track 1 of the 2nd International Timetabling Competition [12]. This is a much richer formulation, taken mainly from British universities, that includes rooms and many specific constraints. Among others, periods and exams can have different length, so that

some periods might be unsuitable for certain exams. Rooms might be used either exclusively or shares among exams (preferably of the same length). Exams might have precedence constraints. The objective function components regard mainly the spreading of exams for students and the assignment of large exams in the initial periods.

For this formulation 12 instances are available which are also quite challenging and not solved to optimality up to today. The contributions on this problem include [1, 4, 11].

Another notable formulation is the one proposed by Müller [14], which represents a complex real-world problem. The formulation comes along with 9 challenging instances in XML format.

4 Local Search Solver

We now introduce our local search method. We first describe the search space, the cost function, and the initial solution (Section 4.1), then we introduce the neighborhood relations (Section 4.2), and finally we discuss our metaheuristic strategy (Section 4.3).

4.1 Search space, cost function, and initial solution

A state in the search space is represented by two vectors that store for each event the period when it is scheduled and the location where it takes place. As already mentioned, the location can be either a single room, a roomset, or the *dummy* room (no room assigned). Any period can be assigned to an event unless it is unavailable for the event. For the location assignment, instead, we consider only locations of the correct type, according to the requirements.

This means that the constraints H1 and H5 are always satisfied. Conversely, we let the search method visit also infeasible state, by violating constraints H2, H3, and H4.

The cost function is thus the weighted sum of the penalties of the soft constraints S1–S3 and the violations of hard constraints H2–H4 multiplied by a suitably high weight, such that a single hard constraint violation is never preferred to any combination of soft ones.

The initial solution is generated at random, but satisfying constraints H1 and H5. In particular, for the selection of the location, the choice is among the compatible ones. For example, an event that does not require a room is compatible only with the dummy room, so that this is the location that is always selected.

4.2 Neighborhood relations

The typical basic neighborhood relation in examination timetabling is the reposition of a single event. We call this neighborhood MoveEvent:

MoveEvent (ME) Given an event e, a period p and a location l, the move $\mathsf{ME}\langle e, p, l\rangle$ repositions e at p in l.
Preconditions: p is available for e and l is compatible with e.

In our case, the presence of exams composed by two events suggests that it might be useful to consider a neighborhood that could move the two paired events jointly. However, we should consider the possibility of moving either the single event or the complete exam. This leads us to the following neighborhood:

MoveEventOrExam (MEE) Given an event e, a period p, a location l, a Boolean b, a period p', and a location l', the move $\mathsf{MEE}\langle e, p, l, b, p', l'\rangle$ repositions e at p in l. If b is true, it also repositions the event e' associated with e at p' in l'. If b is false, then p' and l' are ignored.
Preconditions: p is available for e and l is compatible with e; if e is not part of a composite exam then b is false; if b is true, p' is available for e' and l' is compatible with e'.

4.3 Simulated Annealing

As a metaheuristic to guide the local search we use Simulated Annealing (SA) [9]. For an up-to-date exhaustive introduction to Simulated Annealing and its variants we refer to the work of Franzin and Stützle [7].

Our SA procedure starts from an initial random state and at each iteration draws a random move in the MEE neighborhood. For the MEE move selection, we first select uniformly the event e, the period p, and the location l. If e is not part of a composite exam, then b is set to false. If instead e is part of a composite exam, we select $b = true$ with probability p_b and $b = false$ with probability $1 - p_b$. If b is true, p' and l' are randomly selected, whereas if b is false p' and l' are ignored.

As customary for SA, calling Δ the difference of cost induced by the selected move, this is accepted if $\Delta \leq 0$, whereas it is accepted based on time-decreasing exponential distribution (called Metropolis) in case $\Delta > 0$. Specifically, a worsening move is accepted with probability $e^{-\Delta/T}$, where T is the *temperature*. The temperature starts at the initial value T_0, and decreases by being multiplied by a value α (with $0 < \alpha < 1$), each time a fixed number of samples n_s has been drawn.

To the basic SA procedure, we add the *cut-off* mechanism that speeds up the early stages. The idea is to decrease the temperature also when a given number of moves has been accepted. That is, we add a new parameter n_a, that represents the maximum number of accepted moves at each temperature. The temperature is decreased when the first of the following two conditions occurs: (i) the number of sampled moves reaches n_s, (ii) the number of accepted moves reaches n_a.

We use the total number of iterations $\mathcal{I}$ as stop criterion. This guarantees that the running time is the same for all configurations of the SA parameters. With respect to a criterion based on a strict time limit, our criterion has the advantage that it is deterministic (given the random seed) and it is not dependent on the environment, so that each run can be reproduced precisely.

In order to keep total number of iterations $\mathcal{I}$ fixed, one of the parameters, namely n_s, is not left free, but is computed from $\mathcal{I}$ and from the others using the formula:

$$n_s = \mathcal{I} / \left(\frac{\log(T_f/T_0)}{\log \alpha} \right)$$

where T_f is the *expected* final temperature. Notice that T_f is used to compute n_s, but the actual final temperature might fall below, as the potential iterations saved in the early stages, due to the cut-off mechanism, are returned at the end of the search.

Given that n_s is not fixed, n_a is not fixed directly, but is set to be a fraction ρ of the computed n_s, where ρ is a new parameter (that replaces n_a).

The running time is equal for all configurations on the same instance, but may be different from instance to instance, as the computation of the costs depends on the size of the instance.

5 Constraint Programming Model

In order to use MiniZinc, the input format needs to be preprocessed in such a way to flatten all the data (conflicts, distances, ...) at the level of each single event, and write it in terms of a set of arrays.

The decision variables are two vectors of size equal to the number of events, called `EventPeriod` and `EventLocation`, storing the assigned period and the assigned location, respectively. As for the local search solver, the location assigned can be either a single room, a roomset, or the dummy room.

The constraint stating that the same room cannot be used simultaneously by two events is the following, where `LocationOverlap` is a binary input matrix, stating of two locations overlap (`1`) or not (`0`).

```
constraint
  forall(e1 in 1..Events-1, e2 in e1+1..Events)
    (EventPeriod[e1] != EventPeriod[e2] \/
       LocationOverlap[EventLocation[e1], EventLocation[e2]] = 0);
```

Obviously, two locations overlap if they have a room in common; the dummy room does not overlap with any room, not even with itself, so that more than one event can be placed in the dummy room at the same time.

This is the most critical constraint, as it involves a disjunction. The other constraints are relatively straightforward, and are not shown here.

The objective function is obtained as the weighted sum of various components. For example, the variable carrying the count of the soft conflict violations is connected to the main variables by the following constraint.

```
constraint
  ConflictCost = sum(e1 in 1..Events-1, e2 in e1+1..Events
                     where Conflicts[e1,e2] > 0)
        ((EventPeriod[e1] = EventPeriod[e2]) * Conflicts[e1,e2]);
```

Dept.	#inst	Courses		Events		Periods		Rooms		Slots	
		min	max	min	max	min	max	min	max	min	max
D1	7	239	281	239	281	26	52	64	65	2	2
D2	3	57	58	61	62	156	204	0	0	6	6
D3	9	76	89	78	177	48	188	14	15	4	4
D4	6	223	240	235	514	38	88	34	34	2	2
D5	5	125	156	132	426	24	136	17	20	2	2
D6	8	189	207	346	539	52	90	29	29	2	2
D7	2	60	63	136	150	155	330	22	22	5	10

Table 1. Main features of the instances.

The symmetric matrix `Conflicts` carries the value of the penalty of the conflict between two events. It has the conventional value `-1` in case of hard conflict. The other components have similar structure and they are not shown here.

Finally, for variable and value selections, we use the default strategy (i.e., no search annotation is used).

6 Problem Instances

At present, we have collected 40 instances coming from 7 different departments (of 6 different universities), which show a good variety of diverse practical situations. Table 1 summarizes, for each department, the values (minimum and maximum) of the main features of the corresponding instances.

Notice that one department (D2) has 0 rooms, so that all events are assigned to the dummy room. This means that the management has decided to leave outside the system the assignment of the rooms. Notice also that the number of rooms and courses is rather stable within the instances of the same department, whereas the number of events might change considerably. This is due to the fact that in different sessions during the year the exam of the same course might be repeated a different number of times.

7 Experimental Analysis

The experiments have been run on an Ubuntu Linux 18.4 machine with 4 cores Intel® i7-7700 (3.60 GHz), with a single core dedicated to each experiment.

7.1 Tuning

The tuning phase of the local search solver has been performed using the tool JSON2RUN [17], which samples the configurations using the *Hammersley point set* [8] and implements the F-Race procedure [3] for comparing them.

The resulting best configuration is shown in Table 2, which shows also the initial intervals selected based on preliminary experiments.

Name	Description	Value	Range
T_0	Start temperature	118.75	100 — 500
T_f	Final temperature	0.28	0.1 — 1
α	Cooling rate	0.99	0.8 — 0.999
ρ	Accepted moves ratio	0.2	0.1 — 0.3
p_b	Move written/oral together rate	0.75	0.0 — 1.0

Table 2. Parameter settings.

7.2 Comparative results

For the MiniZinc model we have tested the back-ends available in the standard distribution (v. 2.3.2), plus `cplex` (v. 12.9). For all of them we set a timeout of 1 hour for each run.

The results obtained are shown in Table 3, along with the average and best results out of 30 runs of the Simulated Annealing solver, with $\mathcal{I} = 10^8$. The outcome is that all instances have feasible solutions. In addition, some cases can be easily solved to a *perfect* solution (0 cost), some others are more challenging and result in relatively high costs. For the MiniZinc back-ends, the symbol $\times$ means that the solver exhausted the memory, and the symbol — that it has not been able to produce any feasible solution within the time limit. Optimality has been proved only for the 0 cost solutions.

The metaheuristic approach has been able to obtain satisfactory solutions, and it proved to be quite robust, as the gap between the best costs and the average ones is relatively low.

On the contrary, the straightforward CP model in MiniZinc turned out to be unusable for most practical instances, leaving room for search strategies and/or smarter encodings to be developed.

Only for the department D2, `cplex` has been able to provide better results than SA in all three instances, and `coin-bc` in two of them. In particular, in those two instances, both have found the perfect solution, whereas SA is consistently stuck in a solution of cost 22.

8 Conclusions and Future Work

We have modeled a complex real-world version of the examination timetabling problem.

The metaheuristic solver has found good results on most instances, although the presence of a few results far from the optimum is a clue that further improvements are possible. To this aim, we plan to devise new neighborhood relations and different metaheuristic strategies.

The results show that the problem, in its straightforward modeling, is beyond the reach of MiniZinc back-ends. Smarter encodings are necessary in order to try to improve the performances of all the back-ends.

The future work includes also the extension of the model to features that appear in a few cases, which have been neglected in our current formulation. The

Instance	SA			MiniZinc			
	avg.	best	time	chuffed	gecode	coin-bc	cplex
D1-1-16	**180.40**	180	568.1	×	—	×	×
D1-1-17	**134.00**	134	499.2	×	—	×	×
D1-2-16	**258.63**	257	541.6	×	—	×	×
D1-2-17	**352.00**	351	698.9	×	—	×	×
D1-3-16	**478.37**	477	483.3	×	—	×	×
D1-3-17	**354.57**	354	493.1	×	—	×	×
D1-3-18	**80.00**	80	536.6	×	—	×	×
D2-1-18	427.77	426	94.7	—	8731	906	**406**
D2-2-18	22.00	22	88.7	1543	4022	**0**	**0**
D2-3-18	22.00	22	95.0	1873	3985	**0**	**0**
D3-1-16	**0.00**	0	61.9	—	75947	×	—
D3-1-17	**0.00**	0	83.5	—	82948	×	—
D3-1-18	**0.00**	0	83.9	—	82433	×	—
D3-2-16	**0.00**	0	0.8	**0**	—	—	**0**
D3-2-17	**0.00**	0	3.1	**0**	—	—	**0**
D3-2-18	**0.00**	0	3.5	—	—	—	**0**
D3-3-16	**0.00**	0	1.0	**0**	—	—	**0**
D3-3-17	**0.00**	0	2.3	**0**	—	—	**0**
D3-3-18	**0.00**	0	2.2	**0**	—	—	**0**
D4-1-17	**132.43**	18	312.0	×	—	×	×
D4-1-18	**567.73**	563	1401.4	×	—	×	×
D4-2-17	**575.50**	566	1307.3	×	—	×	×
D4-2-18	**11609.33**	9685	39.3	×	—	×	×
D4-3-17	**137.03**	137	462.6	—	—	×	×
D4-3-18	**379.50**	372	555.2	—	—	×	×
D5-1-17	**7361.03**	5870	2.9	—	—	×	×
D5-1-18	**38.00**	36	824.2	—	—	×	×
D5-2-17	**60.20**	60	930.0	—	—	×	×
D5-2-18	**274.37**	270	1237.8	×	—	×	×
D5-3-18	**0.00**	0	68.4	—	—	×	—
D6-1-16	**898.83**	872	1429.1	×	—	×	×
D6-1-17	**740.87**	723	1385.5	×	—	×	×
D6-1-18	**881.50**	873	1420.1	×	—	×	×
D6-2-16	**948.00**	935	1551.3	×	—	×	×
D6-2-17	**943.13**	920	1428.5	×	—	×	×
D6-2-18	**692.03**	683	1650.4	×	—	×	×
D6-3-16	**355.40**	355	882.4	×	—	×	×
D6-3-17	**381.57**	381	969.0	×	—	×	×
D7-1-17	**373.20**	360	222.5	—	56891	×	—
D7-2-17	**766.50**	758	219.3	—	114385	—	—

Table 3. Comparative results.

most important ones are: events that span over several periods, heterogeneous roomsets, conflicts at the level of the day (not only of the single period), exams to be given in the same day and in the same location, and uniform spreading of the primary courses of a curriculum.

References

1. Battistutta, M., Schaerf, A., Urli, T.: Feature-based tuning of single-stage simulated annealing for examination timetabling. Annals of Operations Research **252**(2), 239–254 (2017)
2. Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., Urli, T.: Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Computers & Operations Research **65**, 83–92 (2016)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated F-race: An overview. In: Experimental methods for the analysis of optimization algorithms, pp. 311–336. Springer, Berlin (2010)
4. Bykov, Y., Petrovic, S.: An initial study of a novel step counting hill climbing heuristic applied to timetabling problems. In: Proc. of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-13). pp. 691–693 (2013)
5. Carter, M.W., Laporte, G., Lee, S.Y.: Examination timetabling: Algorithmic strategies and applications. Journal of the Operational Research Society **74**, 373–383 (1996)
6. Ceschia, S., Di Gaspero, L., Schaerf, A.: Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Computers & Operations Research **39**, 1615–1624 (2012)
7. Franzin, A., Stützle, T.: Revisiting simulated annealing: A component-based analysis. Computers and Operations Research **104**, 191–206 (2019)
8. Hammersley, J.M., Handscomb, D.C.: Monte Carlo methods. Chapman and Hall, London (1964)
9. Kirkpatrick, S., Gelatt, D., Vecchi, M.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
10. Leite, N., Fernandes, C., Melício, F., Rosa, A.: A cellular memetic algorithm for the examination timetabling problem. Computers and Operations Research **94**, 118–138 (2018)
11. Leite, N., Melício, F., Rosa, A.: A fast simulated annealing algorithm for the examination timetabling problem. Expert Systems with Applications **122**, 137–151 (2019)
12. McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A.J., Di Gaspero, L., Qu, R., Burke, E.K.: Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing **22**(1), 120–130 (2010)
13. Merlot, L., Boland, N., Hughes, B., Stuckey, P.: A hybrid algorithm for the examination timetabling problem. In: Burke, E., Causmaecker, P.D. (eds.) Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers. Lecture Notes in Computer Science, vol. 2740, pp. 207–231. Springer-Verlag, Berlin-Heidelberg (2003)
14. Müller, T.: Real-life examination timetabling. Journal of Scheduling **19**(3), 257–270 (2016)

15. Qu, R., Burke, E., McCollum, B., Merlot, L., Lee, S.: A survey of search methodologies and automated system development for examination timetabling. Journal of Scheduling **12**(1), 55–89 (2009)
16. Schaerf, A.: A survey of automated timetabling. Artificial Intelligence Review **13**(2), 87–127 (1999)
17. Urli, T.: json2run: a tool for experiment design & analysis. CoRR **abs/1305.1112** (2013)