Noname manuscript No.
(will be inserted by the editor)

Feature-based tuning of single-stage simulated annealing for examination timetabling

Michele Battistutta · Andrea Schaerf · Tommaso Urli

Abstract We propose a Simulated Annealing approach for the Examination Timetabling problem, as formulated in the 2nd International Timetabling Competition. We apply a single-stage procedure in which infeasible solutions are included in the search space and dealt with using suitable penalties. Upon our approach, we perform a statistically-principled experimental analysis, in order to understand the effect of parameter selection on the performance of our algorithm, and to devise a feature-based parameter tuning strategy, which can achieve better generalization on unseen instances with respect to a *one-fits-all* parameter setting.

The outcome of this work is that this rather straightforward search method, if properly tuned, is able to compete with all state-of-the-art specialized solvers on the available instances. As a byproduct of this analysis, we propose and publish a new, larger set of (artificial) instances that could be used for tuning and also as a ground for future comparisons.

Keywords Examination Timetabling · Local Search · Simulated Annealing · Metaheuristics · Linear Regression · Feature-based Parameter Tuning

1 Introduction

We consider the Examination Timetabling problem in the version used in the 2nd International Timetabling Competition (ITC2007 [27], Track 1). For this

Michele Battistutta · Andrea Schaerf
DIEGM, University of Udine
Via delle Scienze 206, 33100 Udine, Italy
E-mail: {michele.battistutta,andrea.schaerf}@uniud.it

Tommaso Urli
Optimisation Research Group, NICTA & the Australian National University
7 London Circuit, Canberra ACT 2601, Australia
E-mail: tommaso.urli@nicta.com.au

problem, we propose a single-stage Simulated Annealing (SA) procedure, taking also inspiration from our previous work on the other two tracks of ITC2007 [**?**,4, 11]. The search space of the procedure includes also infeasible solutions, and the violation of (hard) constraints is included in the cost function using proper weights. In addition, the procedure employs a composite neighborhood relation and a *cutoff* mechanism which speeds up the cooling in the early stage of the search.

We perform an experimental analysis of our solver on the 12 instances released for the ITC2007, which are, up to now, the only available ones. Specifically, we propose a parameter tuning procedure to obtain a good configuration of the solver for the general case. The tuning procedure works in two steps. In the first step, we identify the most important parameters and we fix the value for all the other ones. In the second one, we develop, through a regression model, a linear function that correlates the value of the parameters to the features of the instances. The outcome is that this rather straightforward search method, properly tuned with a statistically-principled procedure, is able to compete with all state-of-the-art specialized solvers, producing also the best results for some instances.

Furthermore, in order to overcome the limitations created by having a small set of instances, we propose a new set of (artificial) instances that are designed to mimic as much as possible the properties of the real ones. To this aim, from a large pool of artificially generated instances, we select the ones whose features are most similar to the real world instances by means of a Principal Component Analysis (PCA [37]). This new data set allows us to separate the *training instances* (artificially generated) from the *validation instances* (the ITC2007 ones), a widely known best practice for cross-validation, so as to avoid possible *overtuning* phenomena that compromise the fairness of the results. The new artificial instances and the costs of our solutions for all instances (ITC2007 and artificial ones) are available at `http://bitbucket.org/satt/examtt-instances`, along with the Python scripts that validate both instances and solution costs.

The paper is organized as follows. In Section 2, we recall the problem formulation. In Section 3, we discuss the related work. In Section 4, we introduce our search procedure. In Section 5, we illustrate our experimental analysis and we compare our results with the ones in the literature. Finally, in Section 6 we draw our conclusions and discuss future work.

2 Problem formulation

The formulation used in this paper is the one proposed for ITC2007. Therefore, we refer to McCollum *et al* [26] for all the details. However, in order to make the paper self-contained, we provide an informal description of the problem and its constraints. The problem consists of the following entities:

Periods. The examination session is divided in a number of periods over a specified time horizon. Each period has a specific length (in minutes), belongs to a day, and can have a penalty for scheduling exams in it.

Rooms. Each *room* has a *capacity*, i.e., a number of seats available for students. In addition, like periods, rooms might be undesired, in the sense that there is a penalty for their use.

Exams. Each exam has to be scheduled in a period and a room. For each exam, a length of execution is given. In addition, an exam might require to be scheduled in a dedicated room. For each exam, the set of students enrolled for it is also given.

Precedence rules. For some pairs of exams a precedence rule is specified. In detail, a precedence rule states that one exam must be scheduled after, at the same time, or at a different time with respect to the other one.

The search space of the problem is defined by the following set of hard constraints, which must be satisfied by every feasible solution:

Conflicts. A student cannot take two exams in the same period.

Room capacity. The capacity of the room cannot be exceeded in any period.

Period length. Each exam must take place in a period at least as long as the exam itself.

Precedence. Precedence rules between exams must be satisfied.

Room exclusivity. Exams which require a dedicated room must not share a room with another exam in the same period.

The objective function is built upon of the following set of soft constraints:

Two exams in a row. A student should not take two exams in consecutive periods in the same day.

Two exams in a day. A student should not take two exams in the same day.

Spread. A student should not take two exams within a fixed number of periods.

Mixed duration. Exams in the same room should have the same duration.

Frontload. Large exams should be scheduled towards the beginning of the session, i.e., in the earlier periods.

Undesired period. An exam should not be scheduled in an undesired period.

Undesired room. An exam should not be scheduled in an undesired room.

The weights of the soft constraints may vary from case to case, and are included in the input file of each instance.

Regarding the instances, the ITC2007 organizers made available 12 instances upon which the results were evaluated. In detail, four instances were published at the beginning of the competition, 4 other two weeks before the deadline, and 4 were hidden to the participants, and used only by the organizers for the validation.

3 Related work

Examination timetabling has been considered in the scientific literature since the 1960s and many formulations have been proposed (see [8,9,36] for surveys of the early work). Until 2007, however, only one formulation had received considerable attention in the research community, and it is the one proposed by Carter *et al* [10]. The main reason for the success of Carter's formulation is the *Toronto benchmark*, which is a data set of 13 real-world instances, collected from universities all around the world (mainly in Canada). An overview of the results achieved for this data set in the literature is available in [33]. Carter's formulation has been criticized to be an oversimplification of the real-world problem (see, e.g., [25]). For this reason, some variants have been proposed in the literature, and consequently the benchmark data set has been complemented with new (artificial) data [28]. These variants, however, up to our knowledge have received limited attention in the community.

Thanks to ITC2007, a new real-world version of the examination timetabling problem [27] has been defined and popularized in the research community. This formulation, on the one hand, includes many practical constraints, on the other hand it is still simple enough to be accepted as a standard by a reasonably large number of researchers. Competition participants have contributed to the literature with their search methods, which comprise scatter search [16], hybrid local search [29], and hyper-heuristics [32].

Subsequent work, still using the competition formulation and data set, have been published recently. The solution techniques range from bee colony optimization [2,3], to hill climbing variants [7], to great deluge local search [17, 24], to GRASP [15], to other hyper-heuristics [6,31,35]. Comparison with the results of the cited papers are provided in Section 5.4, limited to those that report average values obtained following the ITC2007 rules.

The use of Simulated Annealing for timetabling problems can be traced back to the early works of Abramson [1] and Dowsland [13]. Subsequently, Thompson and Dowsland [38,39] have solved a variant of examination timetabling using various versions of Simulated Annealing. Our research group has worked recently on the application of Simulated Annealing to other timetabling problems (see [**?**,11]). In these other problems, however, the behavior of the solver turned out to be quite different, and therefore different tuning methodologies have been applied. In particular, the problem of the present paper is the only one in which the weight applied to hard constraints (namely w_H, see Section 5) is particularly relevant, so that the tuning methodology has been designed ad-hoc for this case. The other substantial difference is in the statistical analysis used to select the parameters. In our previous work, instances are classified depending on their features and the parameters are defined for each group, while here the parameters are calculated by a formula for each instance.

Also Mascia *et al* [23] implemented a features-based tuning for the definition of the parameters for their Robust Tabu Search for the Quadratic Assignment Problem. The main different with our approach is that they use a

fixed set of three features, whereas we start from a larger set and identify the most relevant ones. They compare the results of the resulting solver with a *reactive* technique that modifies the parameters online during the search. The surprising outcome of their experimental analysis is that the feature-based tuning obtains results superior to the reactive techniques.

4 Search method

Our search method is based on local search. To this regard, we use the following characteristics:

Search Space. The search space is composed of all assignments of exams to periods and rooms. States that violate hard constraints (e.g., precedence and conflicts) are included in the search space, and they are penalized in the cost function with a high weight (called w_H).

Neighborhood Relation. The neighborhood relation is composed of the union of two basic moves:

- Reschedule: move a single exam to a new period and/or new room;
- Swap: exchange both period and room of two exams.

The random selection of the candidate neighbor is performed in two steps. First, we decide the neighborhood (Reschedule or Swap) according to a non-uniform distribution that selects Swap with probability sr and Reschedule with probability $1-sr$, where sr stands for "swap rate" and is a parameter of the method. Second, we draw the specific move within the designated neighborhood using a uniform distribution.

Stopping Criterion. The stopping criterion is based on the total number of iterations, so as to have approximately a constant running time independently of the other parameters of the method.

Initial Solution. The initial solution is totally random, and is obtained by assigning a random period and a random room to each exam.

We develop a Simulated Annealing method that uses a cutoff-based, non-geometric cooling scheme [19], that speeds up the search in the initial phase. Specifically, the temperature is decreased (multiplying it by the cooling rate α) when the first of the following two conditions holds: *a)* the allotted budget of iterations at the same temperature (n_S) reaches zero, or *b)* the allotted budget of *accepted* moves (n_A) reaches zero.

The stopping condition of the method is based on the total number of iterations it_{max}, rather than on the final temperature. For the sake of comparability, it_{max} is set to a fixed value such that the expected running time is the one prescribed by the ITC2007 benchmarking tool (324s on our machine). The resulting value is $it_{max} = 5 \times 10^8$.

The desired final temperature t_{min} is passed to the solver and it is used along with it_{max} and the cooling rate α to compute the budget of iterations (neighbors sampled, n_S) at each temperature. More precisely, we pass to it the

ratio $tr = t_0/t_{min}$ between the initial and the final temperature. The formula we use to compute n_S is the following

$$n_S = -\frac{it_{max}}{\log_\alpha tr}.$$

In fact, the final temperature t_{min} is different for each run due to cutoffs, but we consider the one that is reached in case of a standard cutoff-free execution. Similarly, instead of using directly the parameter n_A, we replace it with its ratio $\rho = n_S/n_A$ with the neighbors sampled n_S. The use of the ratios (tr and ρ), rather than absolute values, allows us to generate only meaningful parameter configurations, avoiding situations in which, for instance, the number of neighbors accepted at a given temperature is higher than the number of neighbors sampled.

The parameters of the search procedure are summarized in Table 1, together with their symbols. These parameters are tuned as explained in the following section.

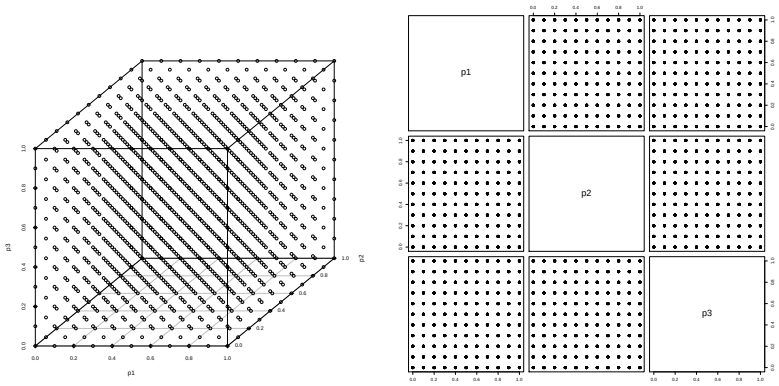
5 Experimental analysis

In this section, we present the experimental analysis we have carried out in order to assess the quality of the solver, and also to tune the parameters of our method and to understand their impact on the solution quality.

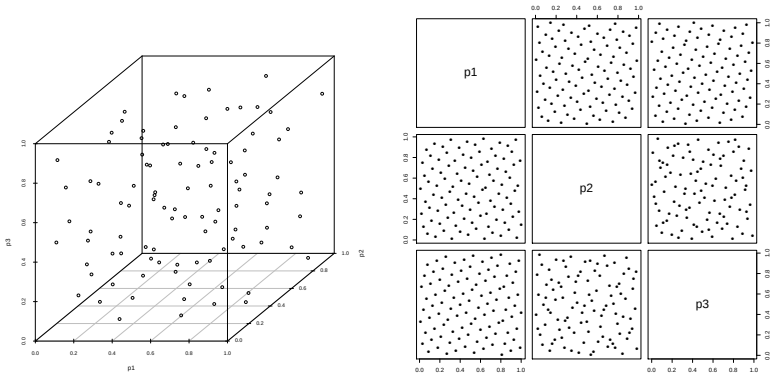
5.1 Preliminary tuning

The first step of our experimental analysis is a parameter tuning phase based on *F-Race* [5]. F-Race takes a set of parameter configurations, and iteratively tests them against an increasingly large set of instances, drawn uniformly at random from a pool. At each step, a Friedman rank sum test or a Wilcoxon rank sum tests are used to sort out the configurations that are significantly worse than the others in a statistical sense, if any. The procedure stops when either a single configuration is left in the race or a predefined maximum number of iterations is executed.

In our tuning phase, the initial set of parameter configurations is generated from the *Hammersley point set* [18], a low-discrepancy deterministic sequence of n points in $\mathbb{R}^k$, where n and k are controllable parameters of the sequence. Points in the Hammersley sequence possess two properties that make them particularly attractive for parameter tuning. Firstly, they tend to be space-filling, i.e., they represent the space rather well, especially for small values of k (e.g., $k < 10$). Secondly, they are scalable both with respect to the number of generated configurations, n, and to the dimensions of the parameter space, k. This feature allows to control the cost of the experimental analysis, a clear advantage over full-factorial designs, where the number of configurations to test grows exponentially in the number of parameters.



(a) Full-factorial design (1000 points)



(b) Hammersley-based design (100 points)

Fig. 1: Comparison between full-factorial and Hammersley-based design.

For the sake of comparison, Figure 1 shows the 3D view and the pair-wise projections of a full-factorial design (with 1000 points) and an Hammersley-based design (with 100 points) on a 3-dimensional space. From the pair-wise projections, it is easy to see that the Hammersley-based design achieves a comparable coverage of the search space with a fraction of the configurations generated by the full-factorial design.

In our setup, we considered $n = 100$ initial configurations, generated from the Hammersley point set defined by the $k = 6$ parameters described in Section 4. An F-Race was run between the generated configurations, with confidence level 95%. For the sake of fairness, we use only the first 8 ITC2007 instances, so as to be in the same conditions of the participants of ITC2007, with whom we compare in Section 5.4. The F-Race stopped after 45 iterations with one winning configuration, which is reported in Table 1, and hereinafter called the *baseline*.

Parameter	**Sym.**	**Range**	***Baseline***
Starting temp.	t_0	$[400, 1000]$	700
Temp. range	tr	$[500, 2000]$	1390.62
Cooling rate	α	$[0.98, 0.99]$	0.99
Neigh. sampled / Neigh. accepted	ρ	$[0.05, 0.2]$	0.14
Hard constraints weight	w_H	$[20, 100]$	21
Swap rate	sr	$[0.3, 0.8]$	0.70
Final temp. (derived)	t_{min}	$[0.2, 2]$	0.50
Neigh. sampled at each temp. (derived)	n_S	$[329733, 1625420]$	764141
Neigh. accepted at each temp. (derived)	n_A	$[16486, 325084]$	106980

Table 1: Parameters identified by automatic tuning through F-Race.

The solver is written in C++, using the framework EasyLocal++ [12, (v. 3)], and compiled using GNU C/C++ (v. 4.9). All the experiments are generated and executed automatically using json2run [40] on an Ubuntu Linux 13.04 machine with 16 Intel® Xeon® CPU E5-2660 (2.20 GHz) physical cores, hyper-threaded to 32 virtual cores. Each run was single-threaded and run on a single virtual core.

5.2 Feature-based tuning

To assess the quality of the preliminary tuning, we tested the baseline against the 12 real-world instances. This validation revealed that, while the algorithm performed well in most of the instances, in four of them, namely instances 4, 6, 11, and 12, the winning configuration produced violations of the hard constraints. The reason for this kind of failure can be found in the underlying assumption behind approaches such as F-Race, i.e., that it is possible to find a single parameter configuration that works well for all the test instances. We call these approaches *single-point tuning*, to differentiate them from the *feature-based tuning* approach proposed in this work.

A subsequent F-Race limited to the failing instances suggested that, while for most parameters the winning value found in the preliminary race is also good for the failing instances, t_0 and w_H must be tuned differently.

We thus ran an exploratory set of experiments on all the instances, with 100 repetitions for each experiment, by varying t_0 and w_H together. These experiments revealed that, because of the temperature update scheme, which is based on α and ρ, the time spent at high temperature is very short, and thus setting a relatively high value for t_0 is always a safe choice. We thus set t_0 as in Table 1, and run another set of experiments to study the effect of w_H, which appears to be much more significant. This second set of experiments revealed a recurrent correlation between the value of w_H and the distributions of costs, which is depicted in Figure 2 for one specific instance (no. 4). For the values reported in the y-axis, the cost is recomputed with w_H set to 1000.

As is appreciable from the plot, when approaching low values of w_H, the cost increases very steeply because of the increase in the hard constraint violations. This is expected, as a low w_H makes it more likely to choose a neigh-

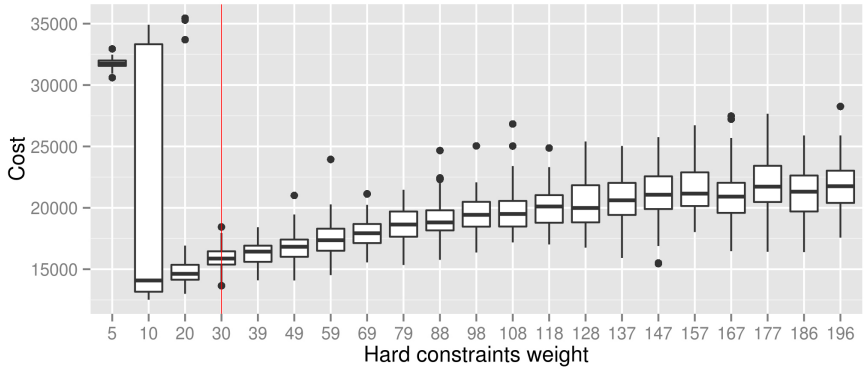


Fig. 2: Correlation between w_H and cost distribution on instance 4.

boring solution with hard constraint violations. On the other hand, as w_H gets farther from the "danger zone", the number of solutions with hard constraint violations decreases, but the costs related to soft constraints increases, because the search procedure is not able to exploit the possibility to cross the feasibility boundary.

Ideally, in this situation, the goal of a good tuning strategy would be to find the parameter configuration that yields the best cost related to soft constraints, while also minimizing the probability of incurring in violations of the hard constraints. Since the ideal w_H differs significantly from instance to instance, such a strategy is not possible in the framework of *single-point* tuning.

Therefore, we decide to explore the *feature-based tuning* strategy that aims at computing the ideal w_H "on-the-fly", based on the features of the instance at hand. The general idea of feature-based tuning is to use regression analysis to learn a predictive formula that, when fed with the feature vector of an instance, outputs the ideal value of w_H. The following sections describe the procedure we used to obtain such a model.

5.2.1 Per-instance tuning

The first step of our feature-based tuning approach is to identify the *ideal* setting of the parameters (in our case w_H) on a per-instance basis. This information is needed for training the regression model.

In order to do this, we need to define what *ideal* means in this context. Since we are ultimately trying to minimize the cost, it makes sense to seek the value of w_H which yields the lower cost value. However, since our approach is stochastic, each setting of w_H produces a distribution of costs, rather than a single one. We observe that a robust strategy to reduce the number of hard constraint violations without increasing too much the cost related to soft constraints, consists in choosing, for each instance, the value of w_H that minimized

I	**E**	**S**	**P**	**R**	$\mathbf{P}_{HC}$	$\mathbf{R}_{HC}$	$\mathbf{FL}_p$	**CD**	**ExR**	**SxE**	**S/Cap**	**PC**
1	607	7,883	54	7	12	0	30	0.05	1.61	53.34	0.75	15.93
2	870	12,484	40	49	8	2	30	0.01	0.44	42.96	0.23	26.45
3	934	16,365	36	48	82	15	20	0.03	0.54	65.47	0.33	34.11
4	273	4,421	21	1	20	0	10	0.15	13.00	79.63	0.86	19.05
5	1,018	8,719	42	3	27	0	30	0.01	8.08	33.59	0.34	72.62
6	242	7,909	16	8	22	0	30	0.06	1.89	76.31	0.56	35.00
7	1,096	13,795	80	15	28	0	30	0.02	0.91	41.51	0.22	43.63
8	598	7,718	80	8	20	1	100	0.05	0.93	52.46	0.43	177.00
9	169	624	25	3	10	0	10	0.08	2.25	14.98	0.60	32.80
10	214	1,415	32	48	58	0	10	0.05	0.14	36.70	0.13	89.38
11	934	16,365	26	40	82	15	20	0.03	0.90	65.47	0.48	51.08
12	78	1,653	12	50	9	7	5	0.18	0.13	47.24	0.20	36.67

Table 2: Instance features for the ITC2007 benchmark instances used in formulas (1) and (2). Symbol definition: **E** (Exams), **S** (Students), **P** (Periods), **R** (Rooms), $\mathbf{P}_{HC}$ (Periods Hard Constraints), $\mathbf{R}_{HC}$ (Rooms Hard Constraints), $\mathbf{FL}_p$ (Frontload(Periods), **CD** (Conflict Density), **ExR** (Exams Per Room), **SxE** (Students Per Exam), **S/Cap** (Students Capacity Ratio), **PC** (Period Conflict)

the 95^{th} percentile of the cost distributions, highlighted by the vertical (red) line in Figure 2.

5.2.2 Feature-based parameter regression

Once an ideal value for w_H has been identified for each instance, we consider the measurable features of the instance, and identify a correlation between them and the ideal value of w_H. We consider the set of features described in [24] and a number of additional ones. Overall, the set of features considered in our analysis are summarized in Table 2. Note that the feature $\mathbf{P_{HC}}$ is reported incorrectly in previous papers (see, e.g., [24] and [17]), because of some redundancy in the input files. Specifically, in some instances, the "same period" precedence rules are written twice (with the exams swapped).

In addition, we use only the first 8 instances of ITC2007, in order to stick to the competition rules, according to which the last 4 instances were hidden to the participants, and thus not usable in the training set. From the feature values, we built a linear regression model in `R` [34] based on 100 repetitions of the experiments with varying w_H. In our model, the dependent variable is w_H, while the control variables are the values of the instance features.

The *model selection* procedure works as follows. First, a model without any control variable is generated. This model generates a constant w_H for all the instances and, as expected, has a rather bad approximation of the ideal w_H. Note that this first model is similar to single-point tuning, in that it outputs the same value of w_H for all the instances, trying to get it as close as possible to the ideal w_H for all the instances. Then, we repeatedly use the *Akaike information criterion* (AIC) [21] to decide which other feature to include in the model. The AIC considers the current model and all the models

Component	Symbol	Coefficient	(Cumulative) R^2
Intercept	–	23.45040	–
Exams	**E**	-0.01924	0.44
Students Capacity Ratio	**S/Cap**	29.10456	0.63
Conflict Density	**CD**	-191.44258	0.91
Period Conflict	**PC**	-0.05139	0.94
Rooms	**R**	-0.11727	0.98

Table 3: Coefficients and correlation of the linear predictor for w_H.

obtained by adding any of the remaining features to the current model, and provides a measure of the obtained increase in accuracy by adding any of the variables. The lower the AIC, the more accurate the model will be. We proceed by greedily adding the next suggested feature, until adding any more features does not improve the model anymore. Table 3 shows, in order, the features added to the model, with their coefficient and cumulative coefficient of correlation (R^2) which is a measure of the accuracy of its prediction.

The obtained linear regression model corresponds to computing the following formula

$$\begin{aligned} \mathbf{w_H} &= 23.45040 \\ &\quad - 0.01924 \times \mathbf{E} \\ &\quad + 29.10456 \times \mathbf{S/Cap} \\ &\quad - 191.44258 \times \mathbf{CD} \\ &\quad - 0.05139 \times \mathbf{PC} \\ &\quad - 0.11727 \times \mathbf{R}, \end{aligned} \tag{1}$$

which can be used as a per-instance tuning of the w_H. Since this formula computes the w_H solely based on the value of the instance features, if the instances used for the training are representative, this tuning strategy should also generalize to unseen instances.

We have validated the effectiveness of this tuning for our approach over the 12 instances of the ITC2007 competition. The results of the comparison are described in Section 5.4.

5.3 Feature-based tuning on artificial instances

The availability of such a small set of instances has initially forced us to use instances 1 to 8 as both the training set and part of the validation set. Because of this, the features-based model we generate can be possibly affected by the peculiarity of each single instance (overtuning), rather than being general enough to be useful for dealing with unseen instances.

The correct procedure dictates to use disjoint sets of instances to define the model for predicting w_H, and for testing the results obtained using it.

However, the ITC2007 instances are, as of today, the only available ones. To overcome this problem, we decide to develop an instance generator, in the

spirit of the one developed by Lopes and Smith-Miles [22], to have availability of an arbitrary number of instances with different sizes and feature profiles.

5.3.1 Instance generator

The generator takes as input the number of exams, which roughly represents the problem size. Then, it randomly selects the conflict density, the number of period and room constraints, and the weights of the soft constraints. All these random values are selected within the ranges of the corresponding values in the ITC2007 instances.

In addition, for each of the period and room constraints, the elements involved (exam plus room or period) are selected at random. Finally, the students are assigned randomly to different exams so that the number of students for each exam follows a log-normal distribution, and the percentage of conflicts per exam is the one randomly selected in advance.

5.3.2 Instance selection with Principal Component Analysis

Clearly, nothing guarantees that generated instances are representative of real-world situations, therefore we need a method to select the most suitable ones. First, instances that can be easily proven to be infeasible are discarded straight away. This is done by using an exact solver, written in MiniZinc [30], that checks only the feasibility of the assignment of exams to periods (which can be done rather efficiently). In addition, instances that are too similar to each other are discarded. To this aim, we employ a Principal Component Analysis (PCA) [20] to compare their structure. The PCA, starting from the features, generates a number of linearly uncorrelated variables (the principal components) trying to retain as much information as possible from the sample, thus operating a dimensionality reduction strategy on the feature space (granted that the number of principal components is smaller than the number of features). As a result of this transformation one can expect the first principal component (PC_1) to have the most variance, and to retain the most information.

We perform the PCA based on the 12 instances of ITC2007, and consider only the first two principal components, PC_1 and PC_2. This allows us to have a bi-dimensional graphical representation of the set of instances. We then compute the principal component values of the generated instances. Generated instances are kept in the pool only if their Euclidean distance (in the PCA plane) from all the others is above a given threshold (set to 0.8). After sorting out the artificial instances that were too similar, we retained a set of 50 instances.

Figure 3 shows the principal components of the original ITC2007 instances (blue triangles) and of the generated instances (pink diamonds). Figure 3 reveals also that the generator has been able to reproduce the feature combinations of most of the ITC2007 instances, and only two of them, namely instances 8 and 11, are still relatively far from the area filled by the generated ones.

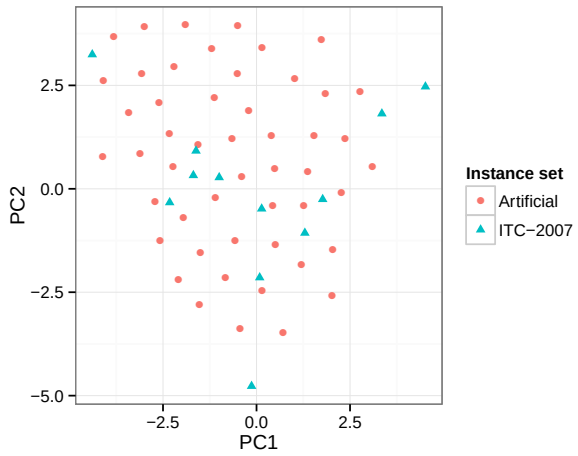


Fig. 3: Representation of the instances using the two principal components.

5.3.3 Parameter regression

Using the same procedure described in Section 5.2.2, but based *exclusively* on the generated instances, we trained a new regression model (Table 4) that fixes w_H using the features of Table 2 and the following formula:

$$\begin{aligned}\mathbf{w_H} &= 16.73100 \\ &+ 102.30000 \times \mathbf{CD} \\ &- 0.48330 \times \mathbf{P} \\ &- 0.17740 \times \mathbf{SxE} \\ &- 1.23900 \times \mathbf{ExR} \\ &- 0.00076 \times \mathbf{S} \\ &+ 0.11590 \times \mathbf{P_{HC}} \\ &+ 0.66660 \times \mathbf{FL_P} \\ &+ 0.78080 \times \mathbf{R_{HC}} \\ &+ 32.46000 \times \mathbf{S/Cap} \\ &+ 0.10100 \times \mathbf{PC}. \end{aligned} \tag{2}$$

This formula has been used to compute our results on the ITC2007 instances.

5.4 Comparison of results

Table 5 reports the comparison with the 5 finalists of ITC2007. Specifically, we add our solver as a further competitor, and re-run the competition adjudication by applying the ranking procedure on 10 runs for each of the 6 solvers. For our

Component	Symbol	Coefficient	(Cumulative) R^2
Intercept	–	16.73100	–
Conflict Density	**CD**	102.30000	0.25
Periods	**P**	-0.48330	0.42
Students Per Exam	**SxE**	-0.17740	0.56
Exams Per Room	**ExR**	-1.23900	0.67
Students	**S**	-0.00076	0.76
Periods Hard Constraints	$\mathbf{P_{HC}}$	0.11590	0.82
Frontload(Periods)	$\mathbf{FL_P}$	0.66660	0.86
Rooms Hard Constraints	$\mathbf{R_{HC}}$	0.78080	0.90
Students Capacity Ratio	**S/Cap**	32.46000	0.93
Period Conflict	**PC**	0.10100	0.95

Table 4: Coefficients and correlation of the linear predictor for w_H.

I	**Müller**	**Gogos**	**Atsuna *et al***	**De Smet**	**Pillay**	**Us**	Feasible	Avg. Cost
1	15.5	25.5	45.5	35.5	55.5	5.5	10	3900.3
2	14.25	35.5	48.8	25.5	52.2	6.75	10	400.5
3	29.5	21.5	35.9	53.5	37.1	5.5	10	8145.4
4	30.1	31.95	38.55	48.5	28.4	5.5	10	13868.8
5	15.5	35.5	45.5	25.5	55.5	5.5	10	2804.1
6	20.1	28.95	35.6	45.9	46.95	5.5	10	25929.5
7	15.5	35.5	46.5	25.5	54.5	5.5	10	3643.3
8	15.1	25.5	35.5	55.5	45.5	5.9	10	7680.3
9	15.4	31.4	43.6	33.5	53.4	5.7	10	997.6
10	31.1	52	18.5	7.5	35.9	38	4	13353.3
11	42.5	24.5	46	46	18.5	5.5	10	25204
12	17.9	46.5	7.4	46.5	27.35	37.35	4	5745.6
avg	21.87	32.86	37.28	37.41	42.57	**11.02**		

Table 5: Comparison with the competition finalists.

solver, Table 5 reports also the number of feasible solutions and the average cost among the feasible ones.

According to Table 5 our solver has the lowest sum of ranks (11.02), and thus would have won the competition, if submitted at that time. For this comparison we use formula (1) rather than formula (2), given that also the other participants tuned their solver on the same 8 instances.

Table 6 shows the comparison with subsequent results available from the literature. We include in Table 6 only those results that are compliant with ITC2007 rules, in terms of timeout. Given that this is a general open comparison and not a competition, as customary, is it presented in terms of averages, rather than on ranks. For this comparison, we used the model in formula (2), which was trained without using any ITC2007 instance. The last two columns report our average results and their standard deviation (for 100 runs).

In this comparison, our results are outperformed by the ones of Bykov and Petrovic [7] in 10 out of 12 instances, and they are superior on the 2 remaining ones. With respect to all the other researchers, our results are globally superior.

I	McC[24]	Byk[7]	H-B[17]	Alz[2]	Alz[3]	Gog[15]	Us	
1	4799	4008	5469	5517	5227.81	5032	**3926.96**	(75.74)
2	425	**404**	450	538	457.55	**404**	407.72	(29.29)
3	9251	**8012**	10444	10325	10421.64	9484	8849.46	(242.75)
4	15821	**13312**	20241	16589	16108.27	19607	15617.82	(1345.68)
5	3072	**2582**	3185	3632	3443.72	3158	2849	(169.03)
6	25935	**25448**	26150	26275	26247.27	26310	26081.35	(240.1135)
7	4185	3893	4568	4592	4415.00	4352	**3661.64**	(211.02)
8	7599	**6944**	8081	8328	8225.81	8098	7729.46	(98.15)
9	1071	**949**	1061	—	—	—	991.57	(28.39)
10	14552	**12985**	15294	—	—	—	13999.56	(250.62)
11	29358	**25194**	44820	—	—	—	27781.5	(843.75)
12	5699	**5181**	5464	—	—	—	5550.2	(268.37)

Table 6: Comparison of available results.

6 Conclusions

We have proposed a Simulated Annealing approach to the Examination Timetabling problem in the standard formulation proposed for ITC2007.

Our solver turned out to have complementary characteristics with respect to previous research. In fact, it is able to improve the costs quite significantly on a few instances, but on the other instances it is inferior to the solver of Bykov and Petrovic [7], that relies on the Kempe-chain neighborhood. Our results show that a principled tuning can help overcoming the intrinsic limitation of Simulated Annealing of being very sensitive to the scale of the cost function, and in particular to w_H.

Given that w_H turned out to be the most critical parameter, we should mention that currently ongoing work involves the use of a *strategic oscillation* approach [14, Cap. 4] that adaptively changes the value of w_H, depending on the current number of violations. This seems to be a promising approach to improve our performances on the harder instances.

Regarding the experimental analysis, one of the main obstacles to our study has been the scarcity of instances to use for training the linear regression model. To deal with this aspect, we have developed an instance generator able to create realistic instances with diverse features.

For the future, we want to explore the use of new neighborhood relations, including Kempe-chains, and new metaheuristics, to which we plan to apply to same feature-based tuning procedure. Finally, we plan to improve the quality of the generator by creating a more structured enrollment matrix, that takes care of the different curricula of the students. In fact, the most relevant limit of the current generator is that students are randomly assigned to exams, ignoring possible clustering effects present in the real life problem.

Acknowledgements We thank Eugenio Macor for developing the instance generator.

References

1. D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37(1):98–113, 1991.
2. M Alzaqebah and S Abdullah. An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. *Journal of Scheduling*, pages 1–14, 2013.
3. M. Alzaqebah and S. Abdullah. Hybrid bee colony optimization for examination timetabling problems. *Computers and Operations Research*, 54:142–154, 2015.
4. Ruggero Bellio, Luca Di Gaspero, and Andrea Schaerf. Design and statistical analysis of a hybrid local search algorithm for course timetabling. *Journal of Scheduling*, 15(1):49–61, 2012.
5. Mauro Birattari, Z. Yuan, P. Balaprakash, and Thomas Stützle. F-race and iterated F-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, Berlin, 2010.
6. Edmund K Burke, Rong Qu, and Amr Soghier. Adaptive selection of heuristics for improving exam timetables. *Annals of Operations Research*, 218(1):129–145, 2014.
7. Yuri Bykov and Sanja Petrovic. 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)*, pages 691–693, 2013.
8. M. W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193–202, 1986.
9. M. W. Carter and G. Laporte. Recent developments in pratical examination timetabling. In E. K. Burke and P. Ross, editors, *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling (ICPTAT-95)*, volume 1153 of *Lecture Notes in Computer Science*, pages 3–21, Berlin-Heidelberg, 1996. Springer-Verlag.
10. M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 74:373–383, 1996.
11. Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, 39:1615–1624, 2012.
12. Luca Di Gaspero and Andrea Schaerf. EasyLocal++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.
13. K. A. Dowsland. A timetabling problem in which clashes are inevitable. *Journal of the Operational Research Society*, 41(10):907–918, 1990.
14. Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.
15. Christos Gogos, Panayiotis Alefragis, and Efthymios Housos. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 194(1):203–221, 2012.
16. Christos Gogos, George Goulas, Panayiotis Alefragis, Vasilios Kolonias, and Efthymios Housos. Distributed scatter search for the examination timetabling problem. In *Proc. of the 8th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2010)*, pages 211–223, 2010.
17. R Hamilton-Bryce, P McMullan, and B McCollum. Directing selection within an extended great deluge optimization algorithm. In *Proc. of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-13)*, pages 499–508, 2013.
18. John Michael Hammersley, David Christopher Handscomb, and George Weiss. Monte Carlo methods. *Physics today*, 18:55, 1965.
19. D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.
20. Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.
21. Roger Koenker. *Quantile regression*. Cambridge University Press, Cambridge, 2005.
22. Leo Lopes and Kate Smith-Miles. Pitfalls in instance generation for Udine timetabling. In *Learning and Intelligent Optimization (LION4)*, pages 299–302. Springer, 2010.

23. Franco Mascia, Paola Pellegrini, Mauro Birattari, and Thomas Stützle. An analysis of parameter adaptation in reactive tabu search. *International Transactions in Operational Research*, 21(1):127–152, 2014.
24. B McCollum, PJ McMullan, AJ Parkes, EK Burke, and S Abdullah. An extended great deluge approach to the examination timetabling problem. In *Proc. of the 4th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-09)*, pages 424–434, 2009.
25. Barry McCollum. A perspective on bridging the gap in university timetabling. In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), selected papers*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23, Berlin-Heidelberg, 2007. Springer-Verlag.
26. Barry McCollum, Paul McMullan, Edmund K. Burke, Andrew J. Parkes, and Rong Qu. The second international timetabling competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, Queen's University, Belfast (UK), September 2007.
27. Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, and Edmund K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.
28. Liam Merlot, Natashia Boland, Barry Hughes, and Peter Stuckey. A hybrid algorithm for the examination timetabling problem. In Edmund Burke and Patrick De Causmaecker, editors, *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers*, volume 2740 of *Lecture Notes in Computer Science*, pages 207–231, Berlin-Heidelberg, 2003. Springer-Verlag.
29. Tomáš Müller. ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1):429–446, 2009.
30. Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. MiniZinc: Towards A Standard CP Modelling Language. In *Principles and Practice of Constraint Programming CP 2007*, pages 529–543, 2007.
31. Ender Özcan, Anas Elhag, and Viral Shah. A study of hyper-heuristics for examination timetabling. In *Proc. of the 9th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2012)*, pages 410–414, 2012.
32. Nelishia Pillay. Evolving hyper-heuristics for a highly constrained examination timetabling problem. In *Proc. of the 8th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2010)*, pages 211–223, 2010.
33. R. Qu, E. Burke, B. McCollum, L. Merlot, and S.Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.
34. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.
35. Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 45(2):217–228, 2014.
36. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
37. K. A. Smith-Miles and L. Lopes. Generalising algorithm performance in instance space: A timetabling case study. In C. A. Coello Coello, editor, *Learning and Intelligent Optimization (LION 2011)*, number 6683 in Lecture Notes in Computer Science, pages 524–538. Springer, 2011.
38. Jonathan M Thompson and Kathryn A Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, 63(1):105–128, 1996.
39. Jonathan M Thompson and Kathryn A Dowsland. A robust simulated annealing based examination timetabling system. *Computers and Operations Research*, 25(7):637–648, 1998.
40. Tommaso Urli. json2run: a tool for experiment design & analysis. *CoRR*, abs/1305.1112, 2013.