10th International Conference of the Practice and Theory of Automated Timetabling
PATAT 2014, 26-29 August 2014, York, United Kingdom

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

Michele Battistutta · Andrea Schaerf · Tommaso Urli

Abstract We propose a single-stage Simulated Annealing procedure for the Examination Timetabling problem (as formulated in the 2nd International Timetabling Competition). Over our approach, we perform a statistically principled experimental analysis, in order to understand the effect of parameters and to devise a feature-based parameter tuning strategy. The outcome of this work (which is still ongoing) is that this rather straightforward search method, if properly tuned, is able to compete with all the state-of-the-art specialized solvers.

Keywords Examination Timetabling · Simulated Annealing · Parameter Tuning

1 Introduction

We consider the Examination Timetabling problem in the version used in the 2nd International Timetabling Competition (ITC2007 [16], Track 1). For this problem (see [15] for the detailed formulation), we propose a single-stage Simulated Annealing (SA) procedure, along the lines of our previous work on the other two tracks of ITC2007 [2, 3, 6].

We perform an experimental analysis of our solver on the 12 public 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

M. Battistutta, A. Schaerf, and T. Urli
DIEGM, University of Udine
Via delle Scienze 206
E-mail: {michele.battistutta,schaerf,tommaso.urli}@uniud.it

a regression model, a linear function that correlates the value of the most important parameters to the features of the instances.

The outcome of this work (which is still ongoing) 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 a few instances.

2 Search method

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

Search Space: The search space is composed by all the assignments of exams to periods and rooms. States that violate hard constraints (e.g., precedences 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 by the union of two basic moves: *1)* Reschedule a single exam to a new period and/or new room *2)* Swap both period and room of two exams. The random selection of the candidate neighbor is performed in two steps: First select 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, perform a uniform selection of the specific move within the corresponding neighborhood.

Stop Criterion: The stop 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 employ a Simulated Annealing method that uses a cutoff-based non-geometric cooling scheme[11], 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 number of iterations (n_S) has been expired or *b)* the allotted number of *accepted* moves (n_A) have been performed.

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

The 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 number of neighbors sampled at each temperature (n_S). More precisely, we pass to it the ratio $tr = t_0/t_{min}$ between the initial and the final one.

$$n_S = it_{max} \Big/ \left(\frac{-\log(tr)}{\log \alpha} \right) \quad (1)$$

Actually, the final temperature t_{min} is different for each run due to cut-offs, but we consider the one that is reached in case of a standard cut-off-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) instead of absolute values prevents from including meaningless configurations in the analysis, such as those in which the final temperature is greater than the initial one.

Summarizing, the search procedure relies on the following six parameters:

- starting temperature (t_0),
- temperature range (tr),
- ratio between neighbors accepted and neighbors sampled at each temperature (ρ),
- cooling rate (α),
- hard constraints weight (w_H),
- swap rate (sr)

that have to be tuned as explained in the following section.

3 Experimental analysis

In order to tune the parameters of our method, we have carried out a statistically principled experimental analysis over the 12 available instances from the ITC2007 competition.

3.1 Preliminary tuning

For the tuning phase, without resorting to any "premature commitment" [10], we have assigned meaningful ranges to the six parameters described in Section 2, and we have sampled 100 alternative configurations from the *Hammersley point set* [9] based upon such ranges. The Hammersley point set *is scalable*, both with respect to the number of sampled points, and to the dimensions of the sampling space. Moreover, the sampled points exhibit *low discrepancy*, i.e., they are space-filling, despite being random-like, and are thus particularly indicated for parameter tuning applications. The sampled configurations were then tuned through an automated *F-Race(RSD)* [4] with confidence 95 (p-value < 0.05). The considered parameters are summarized in Table 1, together with the values identified by the tuning.

All the experiments were generated and executed automatically using the tool JSON2RUN [18] on an Ubuntu Linux 13.04 machine with 16 Intel® Xeon®

Parameter	Symbol	Tuned value
Starting temperature	t_0	900
Temperature range	tr	1000
Cooling rate	α	0.99
Neighbors sampled / neighbors accepted ratio	ρ	0.08
Hard constraints weight	w_H	30
Swap rate	sr	0.8
Final temperature	t_{min}	0.9 (derived)
Neighbors sampled at each temperature	n_S	727467 (derived)
Neighbors accepted at each temperature	n_A	58197 (derived)

Table 1 Parameters identified by automatic tuning through *F-Race(RSD)*.

CPU E5-2660 (2.20 GHz) physical cores, hyper-threaded to 32 virtual cores. A single virtual core has been dedicated to each experiment.

3.2 Feature-based tuning

The results attained by our tuned algorithm were good on some instances, but not on all of them. We therefore decided to investigate the origin of this effect, by looking closely at the instances that failed. The outcome of the analysis was that the overall best parameter configuration (see Table 1) was actually sub-optimal for the failing instances, and yielded violations of the hard constraints. A subsequent *F-Race(RSD)* limited to those instances, revealed that, while for most parameters the winning values found in the preliminary race were also good for the failing instances, t_0 and w_H had to be tuned differently. Since *F-Race(RSD)* is based on ranks, and not directly on the obtained costs, the failing instances were only seen as instances with low statistical significance.

We thus ran an exploratory set of experiments on all the instances, and with 100 repetitions for each experiment, by varying t_0 and w_H together. The experiments revealed that, because of the geometric-like temperature update scheme, which is based on α and ρ, the time spent at high temperature is very short, and thus setting a high value for t_0 is always a reasonably safe choice. We thus set $t_0 = 1000$, and ran another set of experiments to study the effect of w_H, which appeared to be much more relevant.

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 1 for one specific instance (no. 4).

As it is visible 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 neighboring 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, the goal of a good tuning would be to find the parameter configuration that yields the best cost related to soft constraints, while also minimizing

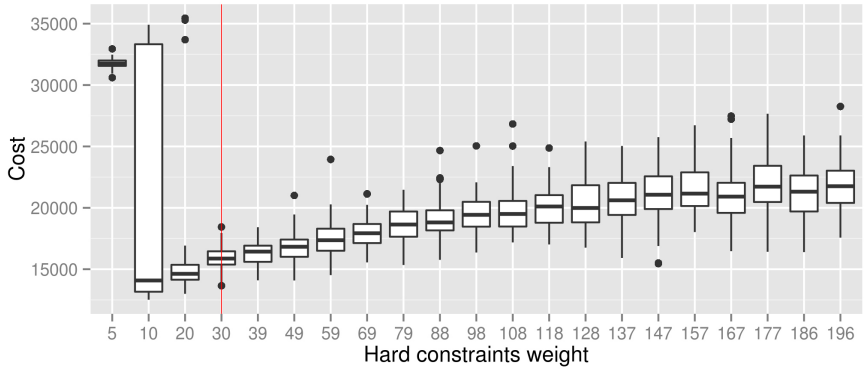

Fig. 1 Correlation between w_H and cost distribution (in logarithmic scale) on instance 3.

the number of violations of the hard constraints. However, since the ideal w_H can differ significantly from instance to instance, we decided to drop the choice of single-point tuning, i.e., one parameter configuration for all instances, and explored a way to compute the ideal w_H "on-the-fly", based on the features of the instance at hand.

3.2.1 Per-instance tuning

First, we observed that a robust strategy to reduce the number of hard constraint violations without increasing too much the cost related to soft constraints, consisted in choosing, for each instance, the value of w_H that minimized the 95^{th} percentile, of the cost distributions (highlighted by the vertical red line in Figure 1).

3.2.2 Feature-based parameter regression

Once an ideal value for w_H was identified for each instance, we looked at the instance features to see whether it was possible predict this value dynamically based on them.

We have considered the set of features described in [14], and a number of additional ones. The features that turned out more useful in our case are summarized in Table 2.

Given the feature values, we built a linear regression model in `R` [17] based on the 100 repetitions of the experiments with varying w_H. The model selection procedure works as follows. First, a model without any variable is generated, this model generates a constant w_H for all the instances and, as expected, has a bad approximation of the ideal w_H. Then, we add one feature at a time, always choosing the one that, if added to the model, minimizes the *Akaike information criterion* (AIC) [12], which is known to have good performances for prediction. The procedure stops when adding one more feature would not lead to a model with a better approximation.

I	Exams	Students	Periods	Rooms	Two in a row	Two in a day	Period spread	Frontload (Exams)	Frontload (Periods)
1	607	7891	54	7	7	5	5	100	30
2	870	12743	40	49	15	5	1	250	30
3	934	16439	36	48	15	10	4	200	20
4	273	5045	21	1	9	5	2	50	10
5	1018	9253	42	3	40	15	5	250	30
6	242	7909	16	8	20	5	20	25	30
7	1096	14676	80	15	25	5	10	250	30
8	598	7718	80	8	150	0	15	250	30
9	169	655	25	3	25	10	5	100	10
10	214	1577	32	48	50	0	20	100	10
11	934	16349	26	40	10	50	4	400	20
12	78	1653	12	50	35	10	5	25	5

Table 2 Instance features for the ITC2007 benchmark instances.

Component	**Symbol**	**Coefficient**	**(Cumulative)** R^2
Intercept	—	18.356988	—
Period Spread Penalty	**PS**	2.311492	0.26
Frontload Periods	$\mathbf{Fl_P}$	-1.7743	0.509
Periods	**P**	1.322	0.824
Two In a Day	**TiD**	1.005	0.894
Students	**S**	-0.0027	0.943
Rooms	**R**	0.292	0.975

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

Table 3 shows, in order, the features added to the model, with their coefficient and cumulative coefficient of correlation (R^2) which measures its prediction quality. Therefore, the linear model corresponds to computing the following formula

$$\begin{aligned}\mathbf{w_H} = 18.356988 + 2.311492 \times \mathbf{PS} - 1.7743 \times \mathbf{Fl_P} \\ +1.322 \times \mathbf{P} + 1.005 \times \mathbf{TiD} - 0.0027 \times \mathbf{S} + 0.292 \times \mathbf{R},\end{aligned}$$

which can thus be used to obtain a per-instance tuning of the w_H, which should ideally generalize also to instances outside of the training set.

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 the next section.

4 Comparison of results

Table 4 reports the comparison with the 5 finalists of ITC2007. Specifically, we add our solver as a further competitor, and rerun the competition adjudication by applying the ranking procedure on 10 runs for each of the 6 solvers. According to Table 4 our solver has the lowest sum of ranks (8.58), and thus would have won the competition if submitted at that time.

Table 5 shows our average results (for 100 runs), in comparison with subsequent results available from the literature. We include in Table 5 only those

I	Müller	Gogos	Atsuna *et al*	De Smet	Pillay	Us
1	15.5	25.5	45.5	35.5	55.5	5.5
2	14.8	35.5	48.8	25.5	52.2	6.2
3	28.7	20.5	34.9	53	36.1	9.8
4	30.1	31.95	38.55	48.5	28.4	5.5
5	15.3	35.5	45.5	25.5	55.5	5.7
6	20.1	28.95	35.6	45.9	46.95	5.5
7	15.4	35.5	46.5	25.5	54.5	5.6
8	14.7	25.5	35.5	55.5	45.5	6.3
8	15	31.4	43.6	33.5	53.4	6.1
10	35.9	54.5	21.8	9.5	40.9	20.4
11	41.4	22.5	45	45	16.5	12.6
12	24.8	50	9.7	50	34.7	13.8
avg	22.64	33.11	37.58	37.74	43.34	**8.58**

Table 4 Comparison with the competition finalists.

I	McCollum *et al* [14]		Bykov & Petrovic [5]		Hamilton-Bryce [8]		Alzaqebah [1]		Us	
	$\bar{f}$	F%	$\bar{f}$	F%	$\bar{f}$	F%	$\bar{f}$	F%	$\bar{f}$	F%
1	4799	100	4008	100	5469	100	5517	100	**4004**	100
2	425	100	404	100	450	100	538	100	**399**	100
3	9251	100	**8012**	100	10444	100	10325	100	9033	98
4	15821	100	**13312**	100	20241	100	16589	100	15132	100
5	3072	100	**2582**	100	3185	100	3632	100	2876	100
6	25935	100	**25448**	100	26150	100	26275	100	25912	100
7	4185	100	3893	100	4568	100	4592	100	**3747**	100
8	7599	100	**6944**	100	8081	100	8328	100	7711	100
9	1071	100	**949**	100	1061	100	—	—	994	100
10	14552	100	**12985**	100	15294	100	—	—	14956	96
11	29358	100	**25194**	100	44820	100	—	—	28773	89
12	5699	100	**5181**	100	5464	100	—	—	5648	98

Table 5 Comparison of available results.

results that are compliant with ITC2007 rules, in terms of timeout. The column F% reports the percentage of feasible solutions obtained. Average costs are computed on feasible solutions only.

Table 5 shows that our results are outperformed by the ones of Bykov and Petrovic [5] in 9 out of 12 instances, and they are superior on the 3 remaining ones. With respect to all the other researchers, our results are globally superior.

We acknowledge that our solver does not find feasible solutions more often than the other solvers. This is inherent to the choice of using a single-stage approach that does not solve the feasibility in advance, but rather tries to optimize the objective function and to satisfy the hard constraints all at once.

5 Conclusions

Our solver turned out to have complementary characteristics with respect to previous research. Specifically, it is able to improve the costs on the easier instances (in terms of hard constraints), but it is not always able to find a feasible solution for the hard ones.

Admittedly, the results of Bykov and Petrovic [5], that rely on the Kempe-chain neighborhood, are superior. Nevertheless, we consider these results, which are still preliminary, quite encouraging for further improvements.

Given that w_H turned out to be the most critical parameter, current work involves the use of a *strategic oscillation* approach [7] 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 was the scarcity of instances to use for training the linear regression model. In order for the model to attain better generalization properties, i.e., to work well on instances outside of the training set, the size of the training set should ideally be much larger. To deal with this aspect, we are working on an instance generator, in the spirit of the one developed by Lopes and Smith-Miles [13], that will be able to create realistic instances with diverse features.

References

1. 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.
2. Ruggero Bellio, Sara Ceschia, Luca Di Gaspero, Andrea Schaerf, and Tommaso Urli. A simulated annealing approach to the curriculum-based course timetabling problem. In *Proc. of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-13)*, 2013.
3. 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.
4. Mauro Birattari, Z. Yuan, P. Balaprakash, and Thomas Stützle. *F-Race and iterated F-race: An overview.* Springer, Berlin, 2010.
5. 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.
6. 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.
7. Fred Glover and Manuel Laguna. *Tabu search.* Kluwer Academic Publishers, 1997.
8. 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.
9. John Michael Hammersley, David Christopher Handscomb, and George Weiss. Monte Carlo methods. *Physics today*, 18:55, 1965.

10. Holger H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.
11. 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.
12. Roger Koenker. *Quantile regression*. Cambridge University Press, Cambridge, 2005.
13. Leo Lopes and Kate Smith-Miles. Pitfalls in instance generation for Udine timetabling. In *Learning and Intelligent Optimization (LION4)*, pages 299–302. Springer, 2010.
14. 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.
15. 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.
16. 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.
17. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.
18. Tommaso Urli. json2run: a tool for experiment design & analysis. *CoRR*, abs/1305.1112, 2013.