A Statistical Analysis of the Features of a Hybrid Local Search Algorithm For Course Timetabling Problems

Ruggero Bellio[1] and Luca Di Gaspero[2], Andrea Schaerf[2]

[1] Dipartimento di Scienze Statistiche – Università di Udine
via Treppo 18, I-33100, Udine, Italy
bellio@dss.uniud.it

[2] Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica
Università di Udine
via delle Scienze 208, I-33100, Udine, Italy
l.digaspero@uniud.it, schaerf@uniud.it

Abstract

In this work we study a hybrid local search algorithm for the solution of timetabling problems, and we undertake a systematic statistical study of the relative influence of the relevant features on the performances of the algorithm. In particular, we apply statistical methods for the design and analysis of experiments. This work is still ongoing, and its ultimate objective is to develop a procedure for obtaining the best combination of parameters for the algorithm for a given instance and predicting them for the unseen ones.

1 Introduction

In the last years the research in metaheuristic seems to have reached a certain level of maturity. Indeed, we witness an evolution from articles proposing new metaheuristics or the handcrafted application of existing ones, to papers addressing engineering methodologies for applying those methods and evaluating their behavior.

With this respect, one fundamental issue and a recent trend of research concerns the analysis of this kind of algorithm (see, e.g., [7, 12]). Indeed, many metaheuristics are stochastic in nature and need a careful investigation by means of statistically sound techniques in order to characterize their behavior.

In this paper we look through this lens at the features of a hybrid local search algorithm, based on a complex combination of simulated annealing and dynamic tabu search. The study focuses on a basic timetabling problem, namely the course timetabling problem formulation used for the International Timetabling Competition (ITC-2007) as track 3 [3], named Curriculum-Based Course Timetabling (CB-CTT). The instances upon which the algorithm is experimented are also the official ones of the competition.

Proceedings of the 15th International RCRA workshop (RCRA 2008):
Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion
Udine, Italy, 12–13 December 2008

The remainder of the paper is organized as follows: In Section 2 we provide the definition of the problem and in Section 3 we illustrate the basic local search model for solving it and the two basic components of the hybrid algorithm. The outcomes of the statistical analyses we performed are reported in Section 4. Finally in Section 5 we summarize what we have done and point out the directions for future work.

2 Problem definition

The basic features of CB-CTT are presented in the ITC-2007 web site and in a companion technical report [3]; however, in order to make this paper self-contained, we present here the full problem definition. The problem consists of the following basic entities:

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

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

Rooms. Each *room* has a *capacity*, expressed in terms of number of available seats, and a *location* expressed as an integer value representing a separate building. Some rooms may be not suitable for some courses (because they miss some equipment).

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

The solution of the problem is an assignment of a period (day and timeslot) and a room to all lectures of each course that satisfies a set of constraints. The constraints are split in two categories:

- *Hard constraints*: these are constraints that must be always satisfied in any feasible solution of the problem.
- *Soft constraints (or objectives)*: they are preferential features that a solution should have. Differently from the hard ones, soft constraints can be violated at the price of worsening the quality of the solution.

In the following we detail these two categories of constraints for the problem taken into consideration.

2.1 Hard constraints

(H1) Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods. A violation occurs if a lecture is not scheduled or two lectures are in the same period.

(H2) Conflicts: Lectures of courses in the same curriculum *or taught by the same teacher* must be all scheduled in different periods. Two conflicting lectures in the same period represent one violation. Three conflicting lectures count as 3 violations: one for each pair.

(H3) RoomOccupancy: Two lectures cannot take place in the same room in the same period. Two lectures in the same room at the same period represent one violation. Any extra lecture in the same period and room counts as one more violation.

(H4) Availability: If the teacher of the course is not available to teach that course at a given period, then no lecture of the course can be scheduled at that period. Each lecture in a period unavailable for that course is one violation.

2.2 Soft Constraints

(S1) RoomCapacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures.

(S2) MinWorkingDays: The lectures of each course must be spread into the given minimum number of days. *Each day below the minimum counts as 1 violation.*

(S3) IsolatedLectures: Lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day. *Each isolated lecture in a curriculum counts as 1 violation.*

(S4) RoomStability: All lectures of a course should be given in the same room. *Each distinct room used for the lectures of a course, but the first, counts as 1 violation.*

3 Local Search for CB-CTT

In order to design a local search algorithm for the problem we have to specify some basic local search entities, namely the *search space*, the *neighborhood relation* and the *cost function*.

The search space we consider is composed of all the assignment for which the hard constraints (1) and (4) hold. States for which the hard constraints (2) and (3) do not hold are allowed, but are penalized within the cost function.

As for the neighborhood relation, we employ two neighborhoods. For the Move neighborhood, given a reference solution, we consider as neighbors all the solutions in which a lecture is moved to a different room and/or to a different timeslot. Only the moves that lead to an *empty* room/timeslot pair are allowed. The Swap neighborhood considers as neighbors of a given solution those solutions that have the room/time assignments of a pair of lectures exchanged. The two neighborhoods are combined by means of the set-union operator.

The cost function is a weighted sum of the violations of the aforementioned hard constraints and the violations of the soft constraints. In order to give precedence to feasibility over the objectives, hard constraints are assigned the weight w_H, which is a value greater than the maximum value of soft constraints violations.

In the following we describe in some detail the metaheuristics employed in this study, namely Simulated Annealing and Dynamic Tabu Search, and we illustrate the high-level strategy for combining them.

3.1 Simulated Annealing

Simulated Annealing (SA) [6] is a metaheuristic whose name comes after an analogy with a simulated controlled cooling of a collection of hot vibrating atoms. The idea is based on accepting non-improving moves with probability that decreases with time.

The process starts by creating a random initial solution and randomly generating at each iteration a neighbor of the current solution. The new solution is accepted and becomes the current one if either it is an improving one or with probability $e^{-\Delta f/T}$, where T is a value called the *temperature*. The temperature T is initially set to an appropriately high value T_0. After a fixed number of iterations, the temperature is decreased by the *cooling rate* γ, so that at each cooling step n, $T_n = \gamma \times T_{n-1}$. The procedure stops when the temperature reaches a *low-temperature region*, that is when no solution that increases the cost function is accepted anymore. In this case we say that the system is *frozen*.

The control parameters of the procedure are summarized in Table 1.

Parameter name	**Description**
T_0	Starting temperature
T_{min}	Ending temperature
γ	Cooling rate in the geometric scheme $T_n = \gamma \times T_{n-1}$
n	Number of neighbors sampled at each temperature level

Table 1: Control parameters of the Simulated Annealing metaheuristic

3.2 Dynamic Tabu Search

Tabu Search [4] is a meta-heuristic method in which a fundamental role is played by keeping track of features of previously visited solutions.

The basic mechanism of Tabu Search is quite simple: at each iteration a subset of the neighborhood of the current solution is explored and the neighbor that gives the minimum value of the cost function becomes the new current solution independently of the fact that its value is better or worse than the current solution.

The neighborhood subset is induced by the so-called *tabu list*, i.e., a list of moves that are forbidden to be performed. The tabu list comprises the last moves (to prevent cycling), and it is run as a queue; that is, whenever a new move is accepted as the new current solution, the oldest one is discarded.

In our implementation we adopt the *robust* TS mechanism for managing the tabu list, that is the size of the tabu list is variable and each performed move remains in the list for a number of iterations randomly selected in the range $[k-\delta, k+\delta]$, where k and δ are parameters of the method. Moreover, at each iteration the *full* neighborhood is traversed and all non-tabu neighbors are evaluated. A move is considered as tabu if a move in the list involves the same lecture. The stop criterion we adopt is based on the number of iterations since the last improvement.

Our TS algorithm is dynamic (DTS) in the sense that it changes continuously the shape of the cost function in an adaptive way, thus causing the search trajectory to pass through infeasible states and visit states that have a different structure than the previously visited ones. Namely, the weight of each hard component is let to vary according to the so-called *shifting penalty* mechanism: if for a number k of consecutive iterations all constraints of that component are satisfied (resp. not satisfied), then the weight is divided (multiplied) by a factor α. Besides α, we also consider the minimum (w_{min}), the maximum (w_{max}), and the initial weight (w_{start}) of the hard constraints.

In order to reduce the number of free control parameters, we set some of the parameters values according to the results of a preliminary screening experiment (see Sect. 4.1) whose aim is to eliminate the non-influential parameters. The control parameters for DTS and the values of the unim-

Parameter name	**Parameter value**	**Description**
k	—	Central value of the tabu list range
δ	5	Width of the tabu list range
α	—	Modification factor of the dynamic adaptive modification
w_{start}	1.0	Starting weight
w_{min}	0.0005	Minimum weight
w_{max}	1.0	Maximum weight
$\max_{ii}$	—	Maximum number of iterations without improvement

Table 2: Control parameters of the Dynamic Tabu Search metaheuristic, the dash indicates free parameters

portant ones are summarized in Table 2.

3.3 Combination of metaheuristics

In this work we study a high-level search control strategy that hybridize the basic local search components. This idea is an instantiation of Hoos and Stützle's *Generalized Local Search Machines* (GLSM) [5, Chapter 3], which is a formal framework for describing search control by clearly separating it from the search components. In this framework, the basic search components are represented as states (i.e., nodes) of a Finite State Machine, whereas the transitions (i.e., edges) correspond to conditions for modeling the search control.

In Figure 1 we describe the high-level control strategy for combining SA and DTS. The notation employed in the figure is quite similar to the one used by UML state diagrams. The search starts from an initial random state built by the *IS* component; when this component has finished it unconditionally passes its solution to DTS, which searches until the number of iterations without improvement has reached the maximum value allowed (i.e., its stopping condition becomes true). Whenever this happens, DTS passes its best solution to SA, which runs until its temperature is higher than the minimum value T_{min}. Then SA returns its best solution to DTS, which starts again its search. The whole strategy is stopped from either DTS or SA when an overall timeout τ has expired.

4 Statistical Analyses

The statistical analyses of a timetabling algorithm requires a sequence of steps. Broadly speaking, they include preliminary screening, selection of a suitable experimental design, data analysis and final validation. The use of

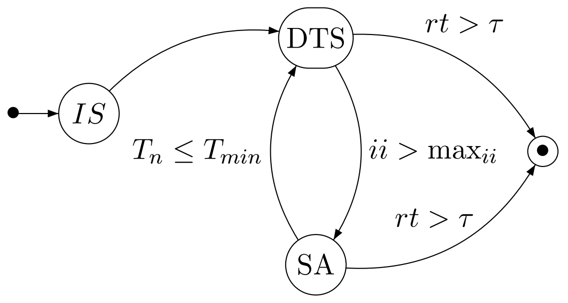

Figure 1: A schematic illustration of the high-level strategy for combining SA and DTS

a statistical software is largely advisable, and our choice is given by the R statistical software [9]. The single steps of the analysis are given as follows.

4.1 Preliminary screening

At first, a screening experiment was run to eliminate unimportant algorithm parameters. In detail we executed SA and DTS with several "extreme" settings of each parameter (keeping the other parameters at fixed "reasonable" values) and we analyzed whether this has any impact in terms of the produced solutions. We eliminate from the study those parameters that seem not to be sensitive to the values assigned.

After this step, we identified the most crucial parameters for minimizing the total cost. Studying only few parameters is clearly a simplification, but this allowed us to achieve a better understanding of the problem under investigation. However, we kept in mind more general extensions, searching a methodology which could be useful in more challenging settings.

4.2 Experimental design

The experimental design should be targeted to the goal of the analysis. In order to analyse the role of algorithm parameters on the resulting performances, it is useful to follow the principle of Response Surface Methodology (RSM). The idea is to approximate the scaled cost (output) as a quadratic function of the algorithm parameters (inputs). More precisely, if the algorithm has h parameters under study, $x = x_1, \ldots, x_h$, each coded in the range $[-1, 1]$, the meta-model for the scaled cost is given by

$$g(x, \beta) = \beta_0 + \sum_{i=1}^{h} \beta_i x_i + \sum_{i=h+1}^{2h} \beta_i x_i^2 + \sum_{i=1}^{h-1} \sum_{j>i}^{h} \beta_{i,j} x_i x_j \,, \tag{1}$$

where the coefficients β are unknown parameters to be estimated from the data.

Designs commonly used in RSM, such as Central Composite Designs (CCD) are excellent for detecting interactions and quadratic terms, but do not provide information about all portions of the experimental region. In many cases, it might be preferable to make use of space-filling designs, which spread points evenly throughout the experimental region [11]. In particular, we adopt the modern solution given by Nearly Orthogonal and Space-filling Latin Hypercubes, introduced by [2]. NOLH are space-filling designs which provide a nearly orthogonal design matrix when used in a first-order regression model. Their main property is that they are particular suitable also for large h, reducing drastically the number of input combinations (n) required. For example, $n = 33$ suffices for NOLH with $2 \leq h \leq 6$, while for $h = 5$ a typical CCD already requires $n = 3^5 = 243$. For each instance, we then ran a NOLH with 65 design points, and 10 replications for each design points. Despite the suggestion that random seed should be taken as a blocking factor [10], preliminary experimentation suggested this was not strongly required for the algorithm under study.

4.3 Data analysis

Statistical analyses have to be performed by considering the results of experiments carried out for several different instances (20 in our case). In order to pool together the results obtained from different instances, a sensible strategy is to use a response variable which is a scaled version of the total cost, suitably normalized using the best available result for each instance. Namely, if z denotes the total cost and z^* the best result, we took as response variable y defined as

$$y = \left[c_0 + \Phi^{-1} \left(1 - \frac{z^* + c_1}{z + c_2} \right) \right] \frac{1}{s}\,.$$

Here c_0, c_1 and c_2 are suitable constants ensuring that the resulting y is always positive, $\Phi(\cdot)$ is the standard normal distribution function, and s is an instance-specific estimate of scale. The monotonic increasing transformation defining the scaled cost y has the effect of obtaining results which are comparable across the different instances. Moreover, for the scaled cost the assumptions required by the statistical methods suitable for our framework are to a good extent satisfied.

The experiments are still ongoing at the time of writing, and only some partial results are already available. Some features are readily detectable, such as the presence of a large amount of heterogeneity between instances and the strong effect of algorithm parameters on the resulting performances. Cooling rate appears to be by far the most important tuning parameter. In the following, we illustrate the methodology that will be used for the analysis of the experimental data.

First of all, our aim is to pool all the instances together, which is sensible since it can be argued that there is a common structure. The methodological tool for pooling together the results of different instances is given by random coefficient models, an extension of the usual mixed effects ANOVA model already proposed by Bang-Jensen et al. (SLS 2007; see also [8]). Mixed models provide smoothed estimates of instance-specific models coefficients, by taking advantage of the common structure assumed for all the data. The details are as follows. If i is the index for the instance ($i = 1, \ldots, I$), j is the index for the design point ($j = 1, \ldots, J$), and k is the index for the different random seed ($k = 1, \ldots, K$), the model is

$$y_{ijk} = g(x_{ij}, \beta_i) + u_k + \varepsilon_{ijk} \,.$$

Here $g(x_{ij}, \beta_i)$ is the the function (1) used for modelling the mean scaled cost as a function of algorithm parameter and instance-specific coefficients β_i. The further step is to assume that $\beta = \beta + b_i$, where β is a fixed effect and b_i are random deviations, often (but not necessarily) assumed to be normally distributed. Furthermore, $u_k \sim N(0, \sigma_u^2)$, $\varepsilon_{ijk} \sim N(0, \sigma_e^2)$. All the random terms are independent of one another. When the random seed effect is dropped, as the various runs are not blocked on seed, then $\sigma_u^2 = 0$.

The model can be fitted as illustrated in [8]. The estimates obtained in such way should be preferable to those obtained in instance-specific analyses. Hopefully, for most of the instances the fitted approximating surface should display a minimum point within the acceptable operating region for the algorithm parameters.

4.4 Final validation

The final step will be devoted to verify that the results obtained by means of the RSM correspond to an algorithm combinations that actually is more advantageous than alternatives ones. Methodologically, however, rather than considering a single point, it is more correct to consider confidence regions for the optimal point. The size of the confidence region provides an effective method for summarising how informative is the RSM for the problem at hand. Furthermore, the fitted model can be used to study the relation between the optimal tuning of algorithm parameters and instance features. This can be of some importance for predicting the performance of the algorithm for unseen instances.

5 Conclusions and future work

This work is still ongoing and we fully accomplished only two of the four phases described in the previous section. At present, therefore, our main contribution is of methodological nature. We still are waiting for the full data to be analyzed to shed light on the relevant features of the algorithm studied. Nevertheless, preliminary results on a simplified version of the algorithm [1] (namely employing only the DTS metaheuristic) seem to be promising to this respect.

Our short-term roadmap is quite straightforward: we plan to finish the computational experiments and the data analysis in a few weeks. As a mid-term goal, instead, we plan to test the proposed methodology also on other problems. This will allow to assess some general validity of the procedure or, instead, will prompt for modifications.

References

[1] Ruggero Bellio, Luca Di Gaspero, and Andrea Schaerf. A statistical analysis of the features of a dynamic tabu search algorithm for course timetabling problems. In Edmund K. Burke and Michel Gendreau, editors, *Proceedings of the 7th International Conference on Pratice and Theory of Automated Timetabling (PATAT-2008)*, 2008. Electronic proceedings. Available at `http://w1.cirrelt.ca/~patat2008/PATAT_7_PROCEEDINGS/Papers/Schaerf-HC1a.pdf`.

[2] Thomas M. Cioppa and Thomas W. Lucas. Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49:45–55, 2007.

[3] Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1, School of Electronics, Electrical Engineering and Computer Science, Queens University, Belfast (UK), August 2007. ITC-2007 site: `http://www.cs.qub.ac.uk/itc2007/`.

[4] Fred Glover and Manuel Laguna. *Tabu search.* Kluwer Academic Publishers, 1997.

[5] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search – Foundations and Applications.* Morgan Kaufmann Publishers, San Francisco, CA (USA), 2005.

[6] S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[7] Luís Paquete, Marco Chiarandini, and Dario Basso, editors. *Proceedings of the Workshop on Empirical Methods for the Analysis of Algorithms, EMAA 2006*, Reykjavik, Iceland, September 9 2006.

[8] José C. Pinheiro and Douglas M. Bates. *Mixed-Effects Models in S and S-Plus*. Springer, 2000.

[9] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

[10] Enda Ridge and Daniel Kudenko. Screening the parameters affecting heuristic performance. In Hod Lipson, editor, *GECCO*, page 180. ACM, 2007.

[11] Thomas P. Ryan. *Modern Experimental Design*. John Wiley & Sons, 2007.

[12] Thomas Stützle, Mauro Birattari, and Holger H. Hoos, editors. *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. SLS 2007*, volume 4638 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Germany, 2007.