Improving the Relevance of Artificial Instances for Curriculum-Based Course Timetabling through Feasibility Prediction

Thomas Feutrier, Nadarajen Veerapen, Marie-Eleonore Kessaci

Improving the Relevance of Artificial Instances for Curriculum-Based Course Timetabling through Feasibility Prediction

Thomas Feutrier
thomas.feutrier@univ-lille.fr
Univ. Lille, CNRS, Centrale Lille,
UMR 9189 - CRIStAL
Lille, France

Nadarajen Veerapen
nadarajen.veerapen@univ-lille.fr
Univ. Lille, CNRS, Centrale Lille,
UMR 9189 - CRIStAL
Lille, France

Marie-Éléonore Kessaci
mkessaci@univ-lille.fr
Univ. Lille, CNRS, Centrale Lille,
UMR 9189 - CRIStAL
Lille, France

ABSTRACT

Solvers for Curriculum-Based Course Timetabling were until recently difficult to configure and evaluate because of the limited number of benchmark instances. Recent work has proposed new real-world instances, as well as thousands of generated ones that can be used to train configurators and for machine learning applications. The less numerous real-world instances can then be used as a test set. To assess whether the generated instances exhibit sufficiently similar behavior to the real ones, we choose to consider a basic indicator: feasibility. We find that 38 % of the artificial instances are infeasible versus 6 % of real-world ones, and show that a feasibility prediction model trained on artificial instances performs extremely poorly on real-world ones. The objective of this paper is therefore to be able to predict which generated instances behave like the real-world instances in order to improve the quality of the training set. As a first step, we propose a selection procedure for the artificial training set that produces a feasibility prediction model that works as well as if it were trained on real-world instances. Then, we propose a pipeline to build a selection model that picks artificial instances that match the infeasibility behavior of the real-world ones.

CCS CONCEPTS

• **Computing methodologies** → **Discrete space search**; *Machine learning*.

KEYWORDS

University timetabling, feasibility prediction, real-world instances, synthetic instances

ACM Reference Format:
Thomas Feutrier, Nadarajen Veerapen, and Marie-Éléonore Kessaci. 2023. Improving the Relevance of Artificial Instances for Curriculum-Based Course Timetabling through Feasibility Prediction. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590690

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590690

1 INTRODUCTION

The goal of Automatic Algorithm Configuration (AAC) [10] is to find the best configuration of an algorithm that generalizes well to new data. To do this, training set instances are used to fit the algorithm configuration and evaluate its performance, while the test set instances are used to measure the generalization ability of the algorithm configuration. By using a separate test set, we can ensure that the algorithm's performance is not over-optimized for the training data, and that it can generalize well to new data. In addition to this, there need to be a sufficient number of instances in both sets, and the instances need to be reasonably diverse.

Finding real-world instances for combinatorial optimization problems can be difficult due to the complexity of the problems, the availability of data, and the diversity of the problems. Nevertheless, it is important to strive for realistic instances in order to evaluate and improve optimization algorithms and ensure that they can be applied effectively in practice. One way around the lack of real-world instances is to use generated, or artificial, ones. In particular, in the context of AAC, the available real-world instances can form the test set, while artificial instances form the training set. For this strategy to work effectively, the artificial instances need to behave as closely as possible to real-world instances in order to good configurations that work on real-world data.

In this paper we focus on Curriculum-Based Course Timetabling (CB-CTT), a variant of the University Timetabling, where a curriculum is a set of courses followed by all students that belong to it. It is an NP-hard problem that has been explored both in the academic literature [5–7, 9] and during scheduling competitions such as the International Timetabling Competition (ITC).

The initial widely available benchmark for CB-CTT, proposed for ITC 2007, contained only 21 (real-world) instances. Many publications and algorithms have been proposed that only consider this limited set of instances, including our own work [3, 4]. Recently, De Coster et al. [2] studied algorithm selection for the CB-CTT. To select the best solver, they compiled a long-awaited larger set of real-world instances and generated thousands of artificial ones based on characteristics derived from descriptive features of the real-world instances.

Our initial exploration of those new artificial instances revealed that they did not all seem to behave like the real-world instances when considering feasibility, even though the instance generator discarded instances deemed infeasible based on their descriptive characteristics. In particular, we noticed that the initial solution constructor of Müller [8], which won the ITC 2007, often failed

to produce a feasible solution within 5 minutes, whereas in the competition itself this was the budget for the whole solving process.

In light of this, this paper proposes the following contributions: 1) an analysis of the feasibility and distribution of features of the proposed instances; 2) a model for feasibility prediction based on instance features; 3) a pipeline for filtering out problematic artificial instances, i.e., keeping only ones that match the infeasibility behavior of the real-world instances.

2 CB-CTT

The Curriculum-Based Course Timetabling (CB-CTT) is a scheduling problem [11] that belongs to the family of University Timetabling problems. Its formulation and set of constraints was formalized for ITC 2007, a competition held for the PATAT 2007 conference.

With CB-CTT, students follow one curriculum. A curriculum is a package of courses. Courses are sets of lectures and are linked to only one referent teacher. The problem is scheduled over a limited number of days, divided into periods or timeslots. One period corresponds to the duration time of one lecture. In CB-CTT, a solution consists in scheduling lectures in timeslots and available rooms following 4 hard and 4 soft constraints. Hard constraints must be respected, for example all lectures must be scheduled. On the contrary soft constraints can be violated. A solver's objective is to minimize the number of violations. These soft constraints represent the wished for characteristics of the timetables. The objective function is an weighted sum of room capacity, minimum working days, curriculum compactness and room stability.

Instance Features. Nine instance features usually used to describe timetabling problems [1] are computed from the instance data : number of lectures, teachers, courses, curricula, rooms periods per day, days, timeslots and number of unavailable constraints (number of timeslots where teachers are unavailable). We also add a feature, called *space*, for the product of rooms and timeslots which represents the number of scheduling options for a lecture if we do not take into account hard constraints.

Real-world and Artificial Instances. There are 21 real-world instances that have been proposed for the International Timetabling Competition 2007. Among them, there are easily solvable ones and others that violate many soft constraints. These instances have been widely used in the literature since 2007 to evaluate the performance of solvers. In 2022, De Coster et al. [2] compiled a larger set of 82 real-world instances and presented a method to generate artificial instances based on 16 instance features and Principal Component Analysis (PCA). They use PCA because preliminary analyses have shown that the instance features are interrelated. Thus, if the number of hours of lessons increases, consequently the number of rooms also increases. Feature values are generated using a Gaussian Kernel on the data from the 82 real-world instances. The Gaussian Kernel is an estimator that takes the values of one variable and estimates the closest normal distribution to it. Obvious impossible instances are removed.

Feature distributions are similar between the two sets of instances but more spread out for the artificial instances. This behavior is explained by the properties of the Gaussian distribution estimator, which allows more extreme values. In Figure 1, the PCA

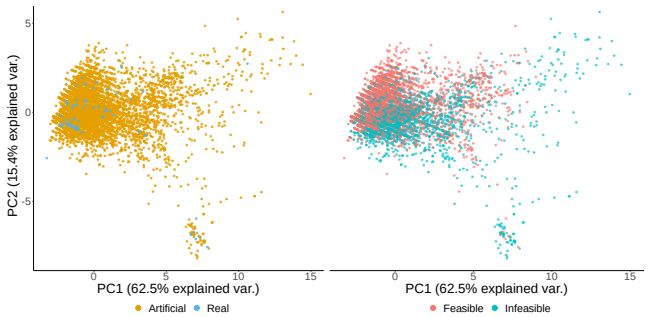

Figure 1: PCA Artifial vs Real and Feasible vs Impossible

on the left highlights that the instance generator succeeded in creating fairly similar problems, with the artificial instances filling in gaps between and around real-world ones.

In summary, we can see that regarding the distribution of features, the artificial instances are quite close to the real-world instances, except concerning the spread, which is more important. However, the PCA shows clearly the effects of the use of a Kernel Gaussian during the generation process [2]. Indeed, artificial data fill all the space created by real-world problems. That is why instances are generated in regions without real-world instances.

3 EXPERIMENTAL PROTOCOL

We use the constructor proposed by Müller [8] to generate initial solutions for each instance. It is very fast, taking less than 1 second for over 80 % of the instances. In order to have a diversity of solutions, the constructor is run 30 times with different random seeds on each of the 7024 problem instances. The time budget is set to 5 minutes and an instance is considered infeasible if no solution is produced within that time frame. Instance features are also extracted.

We use random forests as predictive models: they are easy to implement and have fairly robust default parameters. One drawback of this method, and many other models, is its difficulty to properly handle an unbalanced dataset. That corresponds to a dataset where one class largely outnumbers the other. That is why all models in this paper are trained on a balanced dataset obtained via undersampling, i.e., considering a smaller number of the over-represented class. The models are evaluated w.r.t. 3 metrics: specificity, sensitivity, and accuracy.

4 FEASIBILITY ANALYSIS

To better understand the behaviors that could explain the feasibility, it is essential to analyze CB-CTT instances. Furthermore, we oppose artificial and real-world sets in order to hypothesize on their future usefulness. Indeed, the aim of generating instances is to build complex methods trained on a large number of problems. If these instances are too different from real-world problems, then the methods will not be very useful.

Only 6% of the real-world instances are infeasible (5 out of 82). Therefore infeasibility is not only due to the generator. In contrast, 38% of the generated instances are infeasible (2572 out of 6942).

Figure 2 shows the distribution of the values of a subset of 4 representative instance features for real and generated instances.

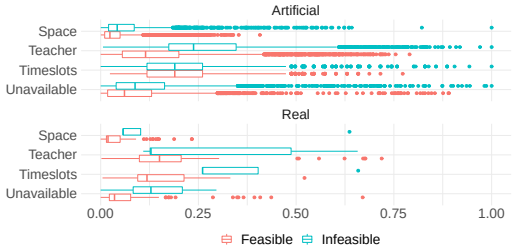

Figure 2: Subset of scaled instance feature distributions

Values are scaled by the MinMax standardization process. Infeasible instances seem to have higher values for all features, except for number of Days and Timeslots. These observations are confirmed by Wilcoxon tests with a threshold of 5%. The Wilcoxon results show that, except for Timeslots, distributions are all significantly different. This difference could be leveraged by a predictive model, although it is less pronounced on artificial instances.

On the right PCA visualization of Figure 1, we observe several feasible and infeasible instances overlapping in a cluster of points more in the center. However, many infeasible instances are located in the rightmost area, away from the cluster. That confirms the instance features can discriminate feasibility.

5 FEASIBILITY PREDICTION

Feasibility Models. Our goal is to create the most robust prediction model possible. Moreover, we want the model to be tested on real-world data because it will be used on future new real-world instances. So three models are created in total. All models are called FP^X for feasibility models trained on subset of instances of type X.

The first model, FP^R, is trained on 82 real-world instances. Due to the small amount of data it has, it is our reference for comparing the two other models and not a feasibility model *per se*. The second one is called FP^A, and is trained on the generated instances. FP^{AC} is the third model built. Contrary to FP^A, it is trained only on a selection of instances, as described below. The construction of $FP^R$and FP^{AC} is presented in Figure 3 (A, B).

One hypothesis for the difference in feasibility between artificial and real-world instances is that the generator added instances that were impossible because of the generation process and not due to their constraints or solving difficulty. Nevertheless, some generated instances are likely more similar to real instances, in terms of feasibility. Thus, an instance selection process is used and, we create the FP^{AC}model trained on those instances.

First, FP^R is used to predict the feasibility of the generated data set. Then, the instances which are well-predicted by FP^R are kept to train FP^{AC}. Our hypothesis is simple: if a generated instance is well-predicted by a model specialized for real-world instances, then its feasibility should be the same as for the real-world instances. This protocol selects 58% of the generated instances, i.e. 4446 problems.

Figure 4 shows no significant difference between the sets of selected and not selected artificial instances. However, there are slight shifts in the distribution between not selected artificial and real instances.

Table 1: Mean values for 3 metrics of the feasibility models on real-world instances.

Model	Accuracy		Sensitivity	Specificity
	Train	Test		
FP^R 3-Fold	0.81	0.78	0.67	0.9
FP^R LOOCV	0.75	1	1	1
FP^A	0.82	0.36	0.33	0.6
FP^{AC}	0.77	0.77	0.76	0.93

Model Evaluation. Three models have been proposed and now they are evaluated on their ability to predict the feasibility of real-world instances. FP^A and FP^{AC} are built and evaluated several times. Each time a different subset of the training set is used and is balanced to have the same number of impossible and feasible instances. The use of different training subsets increases robustness. Accuracy corresponds to the proportion of well-predicted real-world instances. FP^R is evaluated via 3-fold cross-validation. Given the few real instances, FP^R is also tested by a leave-one-out cross-validation (LOOCV).

Table 1, with the average results of each model, shows that FP^R is a useful model. As it is our reference model for prediction on real instances, it confirms our hypothesis and our analyses on the discriminating power of instance features. Accuracy is good, with 0.78 for 3-fold cross-validation and perfect for LOOCV. FP^A can predict the feasibility of a problem if it is artificial (train) but fails to predict real-world feasibility (test). That confirms a difference in the behavior or profile of these two sets despite the rigorous process used by De Coster et al. [2]. Indeed, the accuracy of this scenario is 0.36, which means that a random classification would be more efficient. This behavior is logical if we take into account the difference between the behaviors noted during the analyses. In contrast, FP^{AC}, built using only selected training instances, has a score of 0.77. This matches the performance of FP^R.

Prediction of Selected Artificial Instances. Let us now consider a model that avoids having to compute the feasibility of all generated instances and that depends on another model as was the case before. It would be enough to have an initial artificial data sample belonging to two classes: selected or not.

To build the dataset we use FP^R that tells which artificial instances to choose according to the prediction, Section 5. If an instance is well-predicted then it is selected. Following this process, we create a dataset of artificial instances. Then we create a balanced training set of selected and not selected that represents almost 66% of data. A random forest model, called *Selector*, is built using this training set. It is evaluated using 3-fold cross-validation, giving a sensibility of 0.73, a specificity of 0.86, and an accuracy of 0.82.

Pipeline Validation. We present a pipeline (Figure 5) that includes two models in sequence and evaluate the feasibility prediction of the whole pipeline. The objective of this pipeline is to show that we can use it as is in future work to create a filter that remains efficient despite two models in sequence. In order to obtain statistically robust results, the pipeline is executed 30 times.

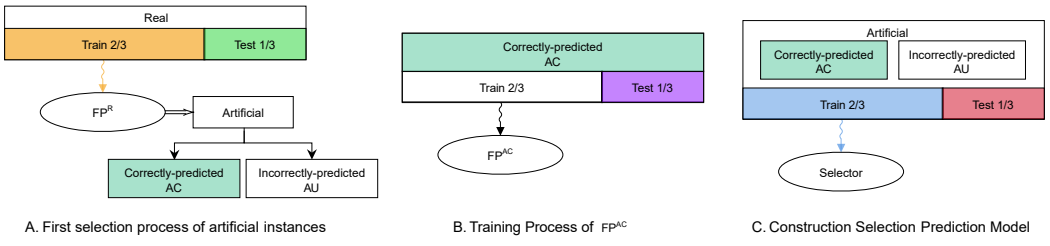

Figure 3: Processes to create optimal subset of artificial instances to predict feasibility

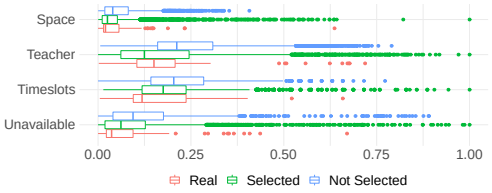

Figure 4: Subset of scaled instance feature distribution for real-world and selected or not selected artificial instances

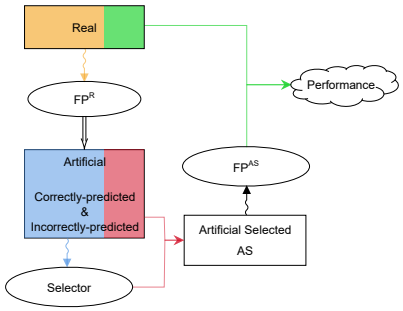

Figure 5: Pipeline

FP^R is trained on a balanced training set from the real-world set. Then FP^R is used to select the generated instances. Generated instances, whose feasibility has been well-predicted, are selected. *Selector* is trained using a balanced training set from these selected instances. Then it is tested on all remaining instances. The generated instances predicted as selected are put aside. They represent the training set of the final feasibility prediction model: FP^{AS}. Finally, we test this model on all real-world instances. The process is repeated 30 times. Each iteration contains two balanced training sets: for FP^R and *Selector*. So repeating iteration checks the robustness of the pipeline. Finally, the mean performance of the feasibility prediction using this pipeline is a mean sensitivity of 0.74, specificity of 0.80 and accuracy of 0.74.

6 CONCLUSION

We address the issue that the available artificial instances for CB-CTT do not necessarily behave like the benchmark real-world instances, especially when feasibility is considered. Indeed, we show that 38 % of the artificial instances are infeasible versus 6 % of real-world ones, and find that a feasibility prediction model trained on artificial instances performs extremely poorly on real-world ones. This difference in behavior is problematic on many levels, but especially when we need to use artificial instances as a training set for AAC, or more generally, for machine learning applications, since we want the training to have the same characteristics as the test composed of the far less numerous real-world instances.

We present an analysis of descriptive feature instances, and propose a selection procedure for the artificial training set that produces a feasibility prediction model that works as well as if it were trained on real-world instances. The resulting random forest model has good accuracy, about 0.77, the same as a model trained on real-world instances. We propose a pipeline to build a selection model that picks artificial instances that match the infeasibility behavior of the real-world ones, delivering an accuracy of 0.82.

The next step in this research is to use the selected instances to actually perform Automatic Algorithm Configuration. Initial results show that we can find better configurations for the state-of-the-art hybrid local search solver and that these configurations are actually algorithmically simpler than the default configuration.

REFERENCES

[1] Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. 2015. An overview of curriculum-based course timetabling. *TOP* 23, 2 (2015), 313–349.
[2] Arnaud De Coster, Nysret Musliu, Andrea Schaerf, Johannes Schoisswohl, and Kate Smith-Miles. 2022. Algorithm selection and instance space analysis for curriculum-based course timetabling. *Journal of Scheduling* 25, 1 (2022), 35–58.
[3] Thomas Feutrier, Marie-Eleonore Kessaci, and Nadarajen Veerapen. 2021. Investigating the landscape of a hybrid local search approach for a timetabling problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '21)*. ACM, New York, NY, USA, 1665–1673.
[4] Thomas Feutrier, Marie-Eleonore Kessaci, and Nadarajen Veerapen. 2022. Analysis of Search Landscape Samplers for Solver Performance Prediction on a University Timetabling Problem. In *Parallel Problem Solving from Nature – PPSN XVII* (2022) *(LNCS)*. Springer International Publishing, Cham, 548–561.
[5] Gerald Lach and Marco E. Lübbecke. 2012. Curriculum based course timetabling: new solutions to Udine benchmark instances. *Annals of Operations Research* 194, 1 (2012), 255–272.
[6] Rhydian Lewis. 2008. A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectrum* 30 (Jan. 2008), 167–190.
[7] Zhipeng Lü, Jin-Kao Hao, and Fred Glover. 2011. Neighborhood analysis: a case study on curriculum-based course timetabling. *J Heuristics* 17, 2 (2011), 97–118.
[8] T. Müller. 2005. *Constraint-based Timetabling Ph. D.* PhD Thesis. Prague.
[9] Ting Song, Mao Chen, Yulong Xu, Dong Wang, Xuekun Song, and Xiangyang Tang. 2021. Competition-guided multi-neighborhood local search algorithm for the university course timetabling problem. *Applied Soft Computing* 110 (2021), 107624.
[10] Thomas Stützle. 2016. Automatic Algorithm Configuration: Methods, Applications, and Perspectives. In *Proceedings of the 8th International Joint Conference on Computational Intelligence, IJCCI 2016, Volume 1*. SciTePress, 7.
[11] Anthony Wren. 1996. Scheduling, timetabling and rostering — A special relationship?. In *Practice and Theory of Automated Timetabling* (1996) *(LNCS)*. Springer, Berlin, Heidelberg, 46–75.