Transformer-based Feature Learning for Algorithm Selection in Combinatorial Optimisation

Alessio Pellegrino
Dept. of Computer Science and Engineering, University of Bologna, Italy

Özgür Akgün
School of Computer Science, University of St Andrews, Scotland

Nguyen Dang
School of Computer Science, University of St Andrews, Scotland

Zeynep Kiziltan
Dept. of Computer Science and Engineering, University of Bologna, Italy

Ian Miguel
School of Computer Science, University of St Andrews, Scotland

1 Introduction

It is well known that no single algorithm excels across all problems or even all instances of a single problem [7]. This led to automated Algorithm Selection (AS), which aims to automatically choose the best algorithm(s) from a *portfolio* with complementary strengths. AS has proven effective in domains such as Boolean Satisfiability (SAT) [9] and Constraint Programming (CP) [3]. We define an algorithm as a *model* (the problem description) and the solver that processes it. Since problems can be modelled in many ways, high-level languages like MiniZinc [5] and Essence [4] allow model specification without low-level detail. Toolchains such as MiniZinc, Conjure [1], and Savile Row [6] then translate those models into lower-level models. AS often uses Machine Learning (ML) to predict the best algorithm(s) based on instance features. Good features must capture relevant aspects of both the problem and the performance of modelling-solver combinations. A well-known feature set is FZN2FEAT [2], which extracts 95 features from FlatZinc [5]: a low-level representation that reflects specific modelling decisions. Rather than relying on low-level translations, we propose using a transformer encoder [8] to learn features directly from high-level problem descriptions. Our approach has three main advantages: (1) features are learned automatically from text, (2) high-level representations may contain more useful information for AS, and (3) the learned features are cheaper to extract. We evaluate this approach using the Essence pipeline across three case studies: Car Sequencing, Covering Array, and Social Golfers.

2 Methodology and key findings

Our approach involves two main steps: (i) learning features from the high-level Essence instance descriptions using a neural network, and (ii) using these features to select the best algorithm.

Transformer-based Neural Network for Feature Learning
We employ a Neural Network incorporating a BERT transformer encoder to process the textual input of problem instances. We developed two models that share a common *core* architecture consisting of: the Transformer Encoder, a Feature Elaborator (two linear layers with corresponding activations), and an Output Layer. The key difference between the two models lies in their output:
B-NN Model, designed to predict the single best algorithm for a given instance. The final activation function is SoftMax, generating a probability distribution over the algorithms in the portfolio.
C-NN Model, aims to learn the *competitiveness* of algorithms in the portfolio (i.e., an algorithm that solves an instance close to the best-performing algorithm's time). The final activation function is the

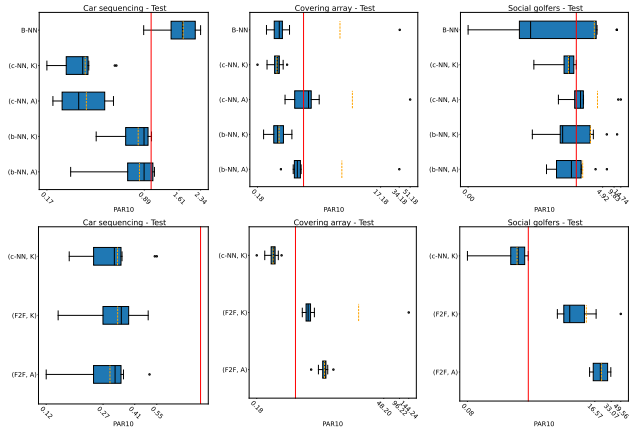

Figure 1 Normalised PAR10 score on the test set comparing the neural features (top) and c-NN against FZN2FEAT (bottom). We define the normalised score as $(p(AS) - p(VBS))/(p(SBS) - p(VBS))$, where $p(AS)$, $p(VBS)$ and $p(SBS)$ are the PAR10 scores of an AS approach. VBS' score is 0 while SBS' score is 1.

Sigmoid, which yields a competitiveness probability for each algorithm.

Algorithm Selection Using Learnt Features

We investigate two strategies for performing AS using the learnt features: We can use the B-NN model directly, where we select the algorithm with the highest probability (**Fully Neural Approach**). We can extract the features from the NN models ('b-NN' from B-NN and 'c-NN' from C-NN) and use them with an external AS algorithm, Autofolio (A) or K-means clustering (K) (**Hybrid Approach**). The extracted features combine the NN's output with the Tanh output from the Feature Elaborator.

2.1 Experimental Evaluation

Experiments were conducted using 10-fold cross-validation, with performance measured by the PAR10 score normalised against SBS and VBS. Our approaches are named as: (features, AS) to have an easy way to identify each combination. The only exception is the fully neural approach named `B-NN` from the model used. In our findings, the hybrid approaches consistently outperformed `B-NN`. The `(c-NN, K)` approach, in particular, yields the best overall performance across the three problems. This suggests that separating feature learning from the AS decision can be more effective. To further prove our findings, we compared our `(c-NN, K)` approach with the same AS trained on FZN2FEAT features. The features learnt by our C-NN model generally outperformed FZN2FEAT features. Furthermore, FZN2FEAT extraction sometimes failed due to memory limitations, a problem not encountered with our method. The feature extraction process for c-NN features is significantly faster and more consistent with marginal impact on the final PAR10 score than for FZN2FEAT features, which often took several seconds with up to minutes on some instances. However, it's important to note that the feature extraction process of our features requires a GPU.

3 Conclusions

We successfully demonstrate that transformer-based models can learn effective instance features directly from high-level descriptions of combinatorial optimisation problems. The proposed hybrid approach, `(c-NN, K)`, showed strong performance, exceeding traditional features, while also incurring significantly lower feature extraction costs. This automated feature learning technique offers a viable path towards more adaptive and powerful algorithm selection systems.

References

1 Ö. Akgün, A. M Frisch, I. P Gent, C. Jefferson, I. Miguel, and P. Nightingale. Conjure: Automatic generation of constraint models from problem specifications. *AIJ*, 310:103751, 2022.

2 R. Amadini, M. Gabbrielli, and J. Mauro. An enhanced features extractor for a portfolio of constraint solvers. In *Proceedings of the 29th annual ACM symposium on applied computing*, pages 1357–1359, 2014.

3 N. Dang, Özgür Akgün, J. Espasa, I. Miguel, and P. Nightingale. A Framework for Generating Informative Benchmark Instances. In Christine Solnon, editor, *CP 2022*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

4 A. M. Frisch, W. Harvey, C. Jefferson, B. Martínez-Hernández, and I. Miguel. Essence: A constraint language for specifying combinatorial problems. *Constraints*, 13(3):268–306, 2008.

5 N. Nethercote, P. J Stuckey, R. Becket, S. Brand, G. J Duck, and G. Tack. Minizinc: Towards a standard cp modelling language. In *CP*, pages 529–543. Springer, 2007.

6 P. Nightingale, Ö. Akgün, I. P Gent, C. Jefferson, I. Miguel, and Patrick Spracklen. Automatically improving constraint models in savile row. *AIJ*, 251:35–61, 2017.

7 J. R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

8 A Vaswani. Attention is all you need. *NeurIPS*, 2017.

9 L. Xu, F. Hutter, Holger H Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *JAIR*, 32:565–606, 2008.