Automatic Feature Learning for Essence: A Case Study on Car Sequencing

Alessio Pellegrino,[1] Özgür Akgün,[2] Nguyen Dang,[2]
Zeynep Kiziltan,[1] Ian Miguel [2]

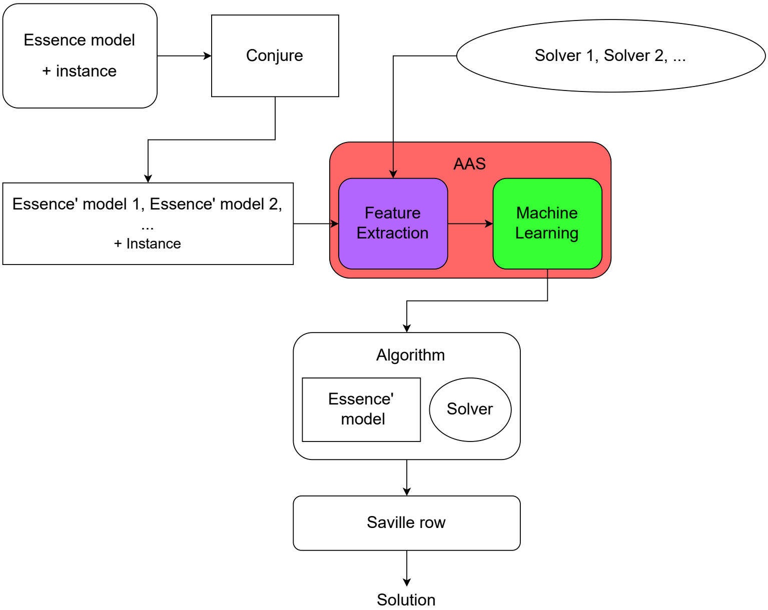
[1]Dept. of Computer Science and Engineering, University of Bologna, Italy
[2]School of Computer Science, University of St Andrews, Scotland

ModRef 2024

Introduction

- Essence is a high-level language designed to abstract problem modelling using a blend of natural language and discrete mathematics. This abstraction addresses the challenging nature of problem modelling.
- Conjure is a software that, given an Essence model, produces a portfolio of Essence' models.
- Hard to choose the best Essence' model and solver for an instance.
- Automatic Algorithm Selection (AAS) aims to select the best algorithm (Essence' model + solver) for a problem instance from a portfolio of algorithms.
- Our contribution is in automatic feature extraction for AAS starting from Essence instances.
- We focus on a specific combinatorial problem: Car sequencing.

Full Picture

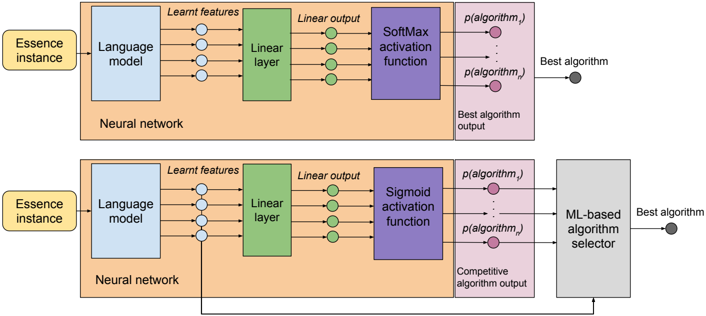


Feature Generation for AAS

- State-of-the-art
  - Feature extraction processes require running the instance for a few seconds.
  - Only concerns the low-level representation.
  - Example: fzn2feat (ACM SAC, 2014).
- Our proposal
  - Extract the features from a high-level representation.
    - Faster (no need to execute nor translate the instance).
    - Allows to exploit high-level properties.
  - Automatic learning of feature directly catered to the task.

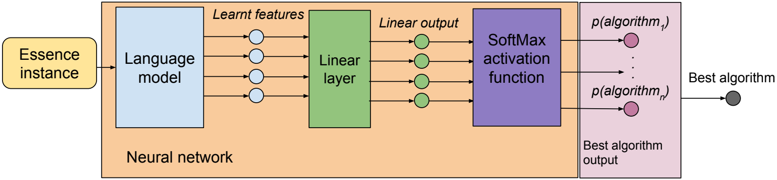
How?

- Use a NN to learn features.
  - Language Model: a BERT-like architecture for problem representation.
- Use the learnt features to predict the algorithm to use.



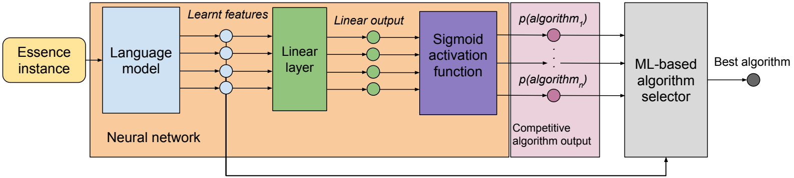
- Two approaches
  - Entirely NN-based.
  - Hybrid of NN and ML-based algorithm selector.

Entirely NN-based Approach



- The model takes as input the text of the instance.
- Trained to predict the best algorithm for an instance.
- A language model produces features for a linear layer which outputs a probability for each algorithm.
- The highest the probability, the more likely is the algorithm to be the best-performing one.
  - All probabilities sum to one.

Hybrid Approach

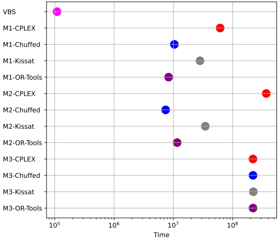


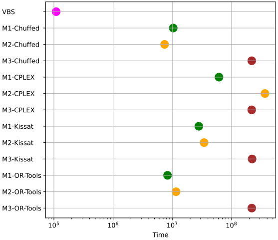
- The model takes as input the text of the instance.
- The network is trained to predict "competitive" algorithms.
  - Competitive = takes less than 10 seconds or less than twice the time of the best algorithm .
- A language model produces features for a linear layer which outputs a probability for each algorithm
  - Probabilities are independent.
- After the network is trained, we extract the features by combining the output of the language model and the output of the linear layer.

Car Sequencing Problem

- Involves scheduling cars in an assembly line.
- Optional features must be evenly distributed to avoid overloading any station.
- Ensure station capacities are not exceeded.

- 10,214 total instances generated using AutoIG[a]
- Three Essence' models.
- Four solvers: Cplex, Kissat, Or-tools, Chuffed.





[a] https://arxiv.org/abs/2205.14753

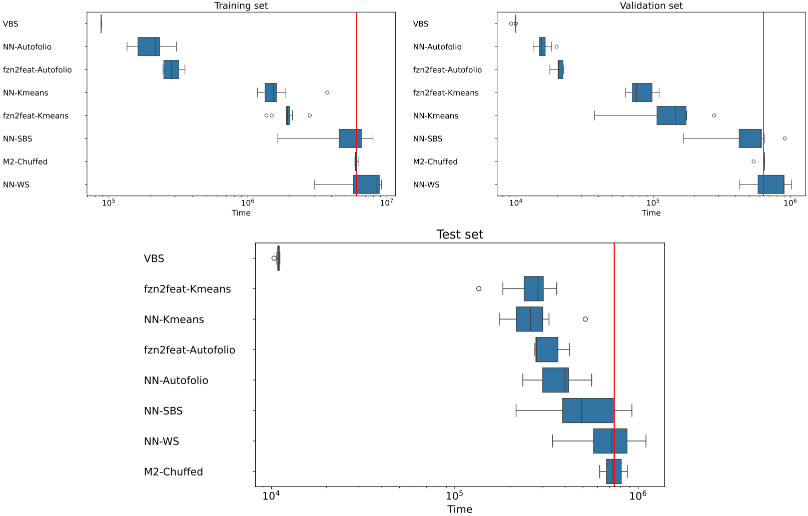
Experimental Evaluation

- Single best (SBS): best algorithm in the portfolio.
- Virtual best (VB): best time combining all algorithms in the portfolio.
- Our goal: Beat SBS, get as close as possible to VB.
- Compared approaches:
  - Entirely NN-based.
  - Autofolio (state-of-the-art AAS, JAIR 2015) with fzn2feat or hybrid features.
  - Kmeans with a given solver per cluster with fzn2feat or hybrid features.
  - Static ordering with competitive hybrid output.

Results - Entirely NN-based Approach

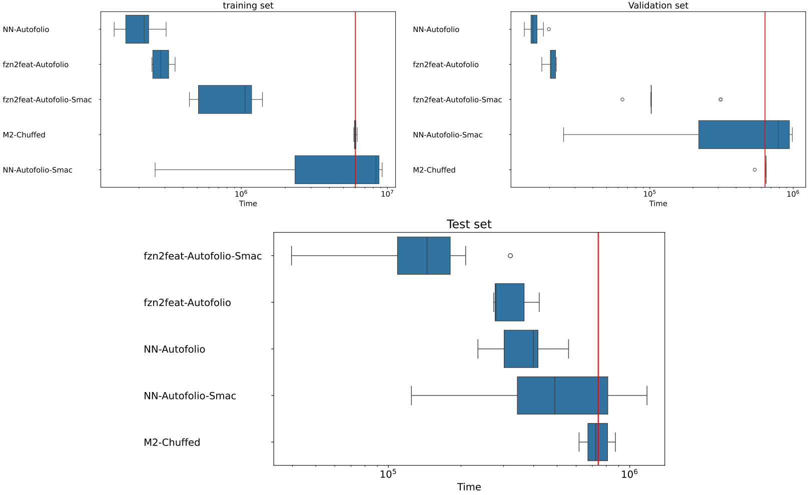
- Worse than SBS.
- Why?
  - Error rate (how much a wrong prediction is worth) cannot be changed from one algorithm to the next.
  - For the NN, choosing an algorithm which is 10% slower than the best yields the same error as choosing an algorithm 3 times slower.
  - Updating NN weights (backpropagation) relies on the error rate.
  - Sample weights: each weight modifies the error generated by a wrong prediction.
  - But sample weights make the training of a NN unstable.

Results - Hybrid Approach



- PAR10 scores of different approaches across 10 folds. The mean score of SBS is shown with the red line.

Results - Autofolio



- PAR10 scores of Autofolio (tuned with SMAC or not) across 10 folds. The mean score of SBS is shown with the red line.

Feature Extraction Cost

	Median	Mean	Max	Min
fzn2feat	6.71	5.38	33.68	0.80
NN	0.02	0.02	0.38	0.02

- Statistics to compute a feature vector in seconds across all the instances.

Summary of Results

Good News

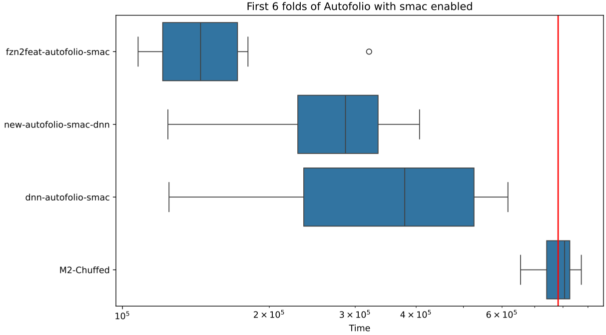
- It works.
- It is fast.

Bad News

- Results have high variability.
- Even though with Kmeans results are slightly better than Fzn2Feat, with Autofolio they are a lot worse.

Further Improvements

- A possible approach is to reimagine the NN-based approach and take advantage of the high number of features.
- We also could keep the current mixed approach, reduce the number of features and fix their representation (e.g. put them into a 0 - 1 range).



- Some early results using only 100 features ("new") instead of 783.

Future Work

- Expand the research on more models.
- Improve performances.
- Try to have a single NN to produce features for all Essence models.

Thank You

Questions?