Selecting SAT Encodings for Pseudo-Boolean and Linear Integer Constraints

The 28th International Conference on Principles and Practice of Constraint Programming

Felix Ulrich-Oltean, Peter Nightingale, James Walker

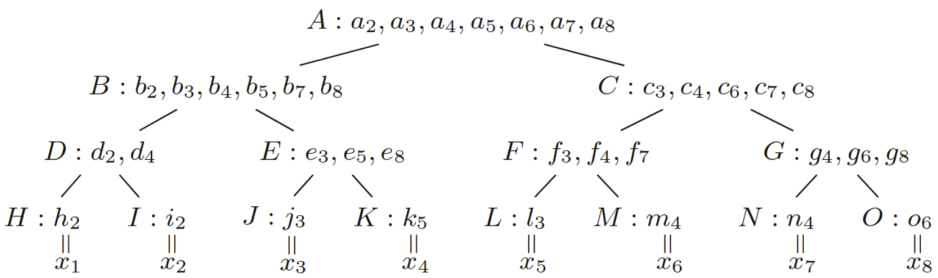
2-5 August 2022, Haifa

Project Aim

Can we use ideas from portfolio approaches to **learn to select** good SAT encodings *of constraints* for new CSP instances?

We focus on pseudo-Boolean / linear integer constraints in this work.

An Example Encoding Scheme



binary tree of $GT(2x_1+2x_2+3x_3+5x_4+3x_5+4x_6+4x_7+6x_8 \leq 7)$

$$\overline{l_{w_1}} \vee \overline{r_{w_2}} \vee o_{w_3} \quad l_{w_1} \in L.vars,\ r_{w_2} \in R.vars,\ w_3 = \min(w_1 + w_2, K + 1) \tag{9}$$

$$\overline{t_w} \vee o_w \quad t_w \in L.vars \cup R.vars \tag{10}$$

$$\overline{a_{k+1}} \tag{11}$$

Figure Diagrams and clauses for the “Generalized Totalizer” from [Bofill et al., 2019]

Overview

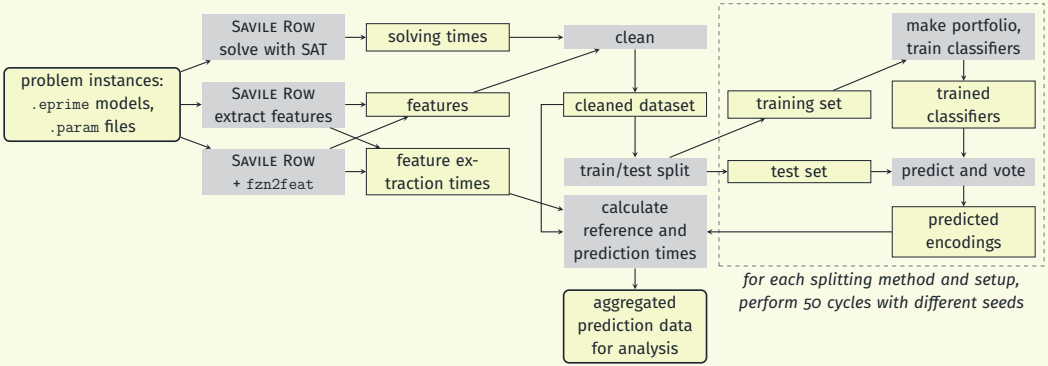


Figure An overview of the steps involved in our experimental investigation. The boxes with solid borders represent data; the grey boxes represent processes.

Distribution of Runtimes and Timeouts

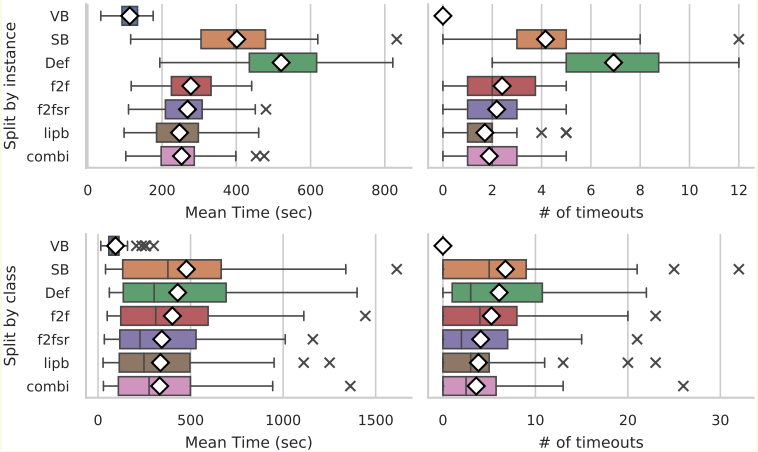


Figure Prediction performance using different featuresets against reference times. We show mean runtime (left) and number of timeouts (right) per test set, when using our preferred setup (*pairwise combined + sample weights + custom loss*).

Findings and Future

Findings

- possible to make good predictions even for unseen classes
- generic features worked well, but constraint-specific features were more useful and led to more robust predictions
- using pairwise classifiers, sample weighting and custom scoring can address the issue of near-miss classifications

Future

- more balanced and diverse corpus
- consider other constraint types
- learn to set different encodings for individual constraints within an instance