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

UNIVERSITY of York

50 YEARS OF COMPUTER SCIENCE AT YORK

www.york.ac.uk/computerscience

Outline

Motivation

Encoding to SAT

Learning

Results and Findings

Conclusion

Motivation

Motivation

Why SAT?

Effective Back-end Solver

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	SICStus Prolog
Free	OR-Tools	PicatSAT	iZplus
Parallel	OR-Tools	PicatSAT	iZplus/Choco 4
Open	OR-Tools	PicatSAT	iZplus/Choco 4
Local Search	Yuck	OscaR/CBLS	

Figure Screenshot of MiniZinc2021 Challenge results from `https://www.minizinc.org/challenge.html`

Effective Back-end Solver

Rank	Main CSP (CPU)	Main COP (CPU)	Fast COP (CPU)	Mini track (CPU)
1st	PicatSAT	PicatSAT	AbsCon	NACRE (hybrid)
2nd	Fun-Scop (hybrid + CryptoMiniSAT)	choco (parallel)	PicatSAT	miniBTD
3rd	Fun-Scop (hybrid + Many Glucose)	AbsCon	choco (paralle)	cosoco

Figure Screenshot of XCSP Challenge 2019 results from `https://xcsp.org/competitions/`

“Free” Gains from an Improving Back-end

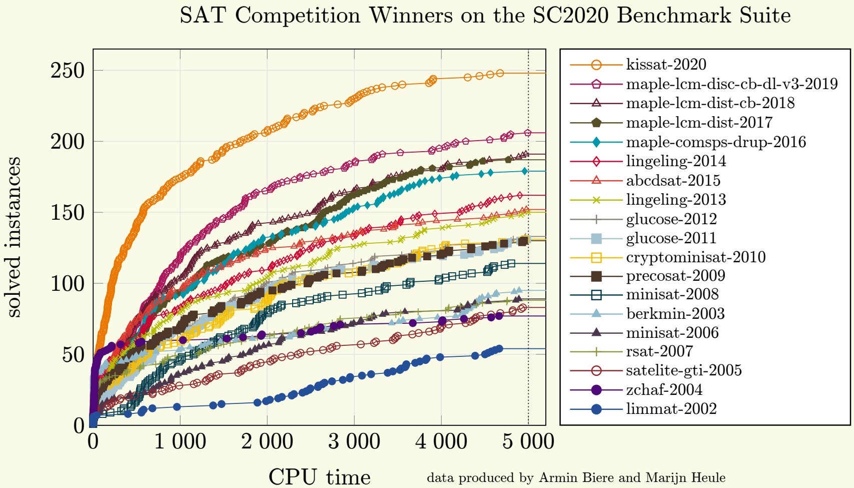


Figure Instances from the 2020 SAT Competition solved by historical winning solvers. Plot from `http://fmv.jku.at/kissat/`

Motivation

Portfolio approaches

Portfolios

- An expert's skill: right tool for the job
- Winner doesn't take it all, a complementary portfolio can perform better, especially when the input is varied
- SAT Competition banned portfolio-based solvers
- SunnyCP, Proteus, MeSAT, ...

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.

Encoding to SAT

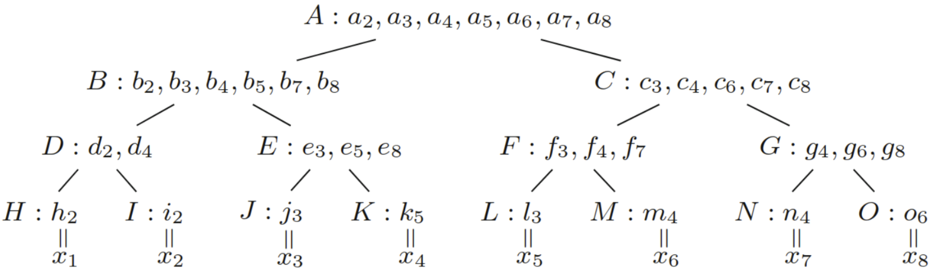
SAVILE ROW, Essence Prime, SAT

```
language ESSENCE' 1.0

given upbound : int
given mincard : int
given weights : matrix indexed by [int(1..n)] of int

letting VARS be domain int(1..n)

$ the boolean decision variables: which items to select
find chosen : matrix indexed by [VARS] of bool

$ the pseudo-Boolean sum constraint
such that upbound >= (sum i : VARS . chosen[i] * weights[i]),

$ one more constraint otherwise the solution is trivial
sum(chosen) >= mincard
```

```
p cnf 48 106
1 0
-11 12 0
-12 13 0
-14 15 0
-15 16 0
...
```

Figure Left: an Essence Prime model for a simple knapsack problem. Right: the beginning of the corresponding Boolean SAT formula as output by SAVILE ROW [Nightingale et al., 2017].

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]

What makes a good encoding?

	enc.	Q1	med	Q3	avg	t.o.	v.	cl.	g.t.
Set1	MDD	3.89	14.78	73	131	87	25	266	3.71
	GSWC	4.50	5.92	277	158	112	105	1076	10.01
	GGT	—	—	—	—	—	—	—	—
	GGPW	0.04	0.04	5.54	93	67	1.0	4.4	0.05
Set2	MDD	0.21	0.41	1.42	74	53	2.1	21	0.28
	GSWC	0.58	0.62	1.09	71	52	6.4	66	0.62
	GGT	2.42	8.83	53	132	95	1.9	120	1.53
	GGPW	0.02	0.03	3.36	89	65	0.6	2.5	0.03

Figure Extract from performance summary in [Bofill et al., 2019]

Learning

Learning

Experimental Setup

Overview

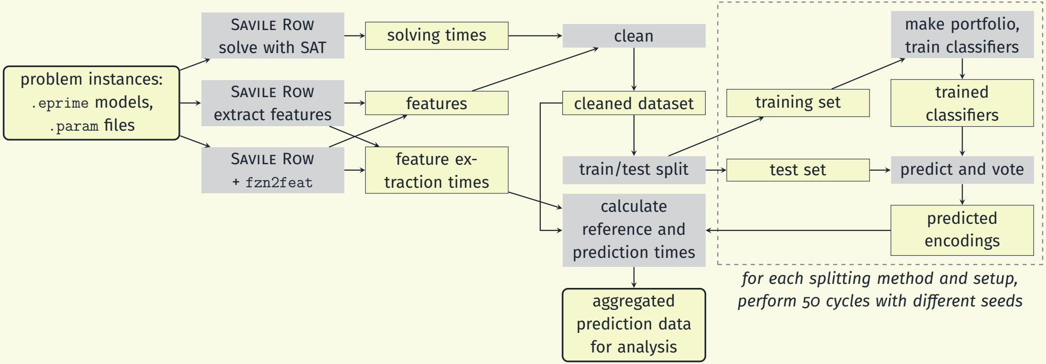


Figure An overview of the steps involved in our experimental investigation. The boxes with solid borders represent data; the grey boxes represent processes.

The corpus

Problem Class	#	PBs	LIs
killerSudoku2	50	1811.2	129.9
carSequencing	49	435.7	0.0
knights	44	170.5	336.9
langford	39	146.2	0.0
opd	38	21.7	74.8
knapsack	30	1.0	1.0
sonet2	24	10.0	1.0
immigration	23	0.0	1.0
bibd-implied	22	410.6	0.0
handball7	20	705.0	1206.0
mrcpsp-pb	20	90.0	45.7
n_queens	20	1593.0	0.0
efpa	20	156.6	0.0
bibd	19	338.7	0.0
n_queens2	16	309.0	0.0
briansBrain	16	0.0	1.0
life	16	0.0	438.9

Problem Class	#	PBs	LIs
molnars	15	0.0	4.0
bpmp	14	14.0	0.0
blackHole	11	202.2	0.0
pegSolitaireTable	8	59.9	0.0
pegSolitaireState	8	59.9	0.0
pegSolitaireAction	8	59.9	0.0
peaceArmyQueens1	7	0.0	1008.0
peaceArmyQueens3	6	0.0	4.0
golomb	6	59.2	38.7
quasiGrp5Idem	6	586.7	0.0
magicSquare	6	118.3	34.0
quasiGrp7	6	410.7	0.0
quasiGrp6	6	410.7	0.0
quasiGrp4NonIdem	4	1067.5	208.0
quasiGrp3NonIdem	4	1067.5	208.0
quasiGrp5NonIdem	4	389.0	0.0
quasiGrp4Idem	4	416.0	208.0

Problem Class	#	PBs	LIs
bacp	4	0.0	25.0
quasiGrp3Idem	4	458.0	208.0
waterBucket	4	0.0	46.0
discreteTomography	2	240.0	0.0
solitaire_battleship	2	72.0	16.0
plotting	1	1.0	28.0
nurse	1	27.0	42.0
grocery	1	0.0	2.0
farm_puzzle1	1	0.0	2.0
diet	1	0.0	6.0
sokoban	1	0.0	24.0
sonet	1	3.0	1.0
contrived	1	0.0	4.0
sportsScheduling	1	166.0	64.0
tickTackToe	1	6.0	14.0

Obtaining Timings

- SAVILE ROW has MDD, GSWC, GGPW, GGT + Tree encodings; we turn on AMO detection
- 5 choices for LI x 5 choices for PBs = 25 configurations
- each instance run with each configuration 5 times (to average out SAT solver randomness) and the median time taken
- timeouts set to 1 hour each for Savile Row and the SAT solver (Kissat)

Learning

Training

Classification

Not an "ordinary" classification task - not every misclassification is the same. We tried some things to address this:

- samples weighted according to "hardness"
- custom loss for hyperparameter tuning cross-validation
- pairwise training and voting inspired by [Lindauer et al., 2015]

Portfolio

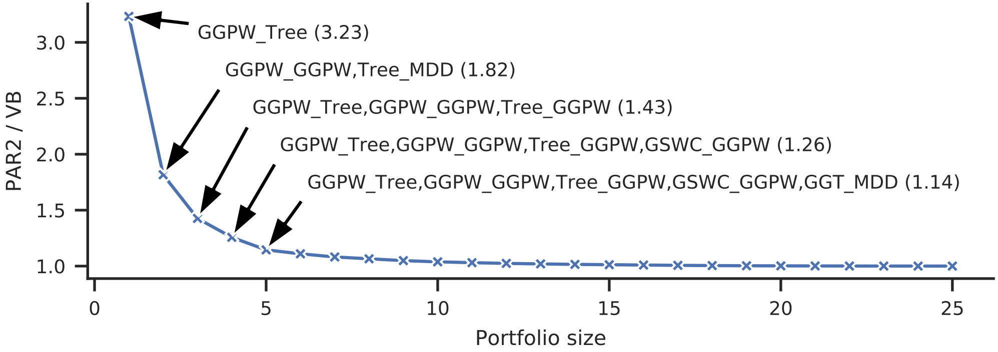


Figure The virtual best (VB) PAR2 run-time on our corpus for all portfolio sizes as a multiple of the overall VB

Features

f2f from fzn2feat [Amadini et al., 2014]: 95 generic CSP instance features relating to constraints, variables, and their domains. Extracted by outputting FlatZinc from Savile Row, then running `fzn2feat`

f2fsr an attempt to extract the same features from SAVILE ROW's internal model just before encoding to SAT

lipb new pb-related features

combi f2fsr and sumpb combined

Features relating to PB and LI constraints

Number of (PB or LI) constraints	No. of distinct values / no. of coefficients
Number of terms	Number of At-Most-One groups (AMOGs)
Sum of coefficients	Mean size of AMO group
Minimum coefficient	Mean AMOG size / number of terms
Maximum coefficient	Mean maximum coefficient size in AMOGs
Median coefficient	Skew of maximum coefficient in AMOGs
IQR of coefficients	Upper limit (k)
Coefficients' quartile skew	$k \times$ number of AMOGs
Number of distinct coefficient values	

Results and Findings

Results and Findings

Evaluating performance

PAR2 Performance for our ML Setups

Reference Times

Split	VB	SB	Def	VW
by instance	1.00	3.55	4.61	9.75
by class	1.00	5.06	4.53	9.49

Predicted Times

	Split by instance				Split by class			
Setup	f2f	f2fsr	lipb	combi	f2f	f2fsr	lipb	combi
pairwise combined	2.62	2.57	**2.41**	2.51	3.88	3.92	**3.75**	3.90
pairwise combined + sw	2.49	2.46	**2.28**	2.37	3.70	4.12	3.86	**3.52**
pairwise combined + cl	2.62	2.43	**2.36**	2.41	3.97	3.98	**3.58**	3.66
pairwise combined + sw + cl	2.45	2.37	**2.18**	2.23	4.24	3.66	3.56	**3.53**
single combined + sw + cl	2.43	2.43	**2.33**	2.36	4.23	4.43	3.89	**3.74**
pairwise separate + sw + cl	2.35	2.26	2.24	**2.18**	4.01	**3.90**	4.36	3.95

Table PAR2 runtimes including feature extraction, as a multple of the Virtual Best time

Distribution of Runtimes and Timeouts

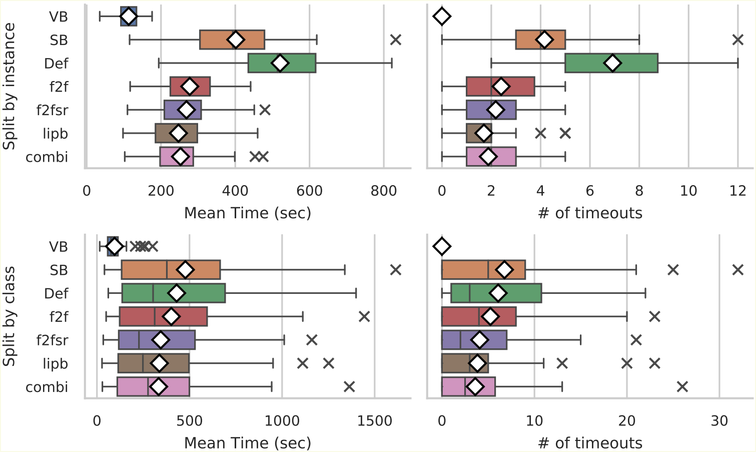


Figure Prediction performance using different featuresets against reference times. We show mean runtime (left) and number of timeouts (right) per test set, when using our preferred setup (*pairwise combined + sample weights + custom loss*).

Comparison with AUTOFOLIO

Reference Times

Split	VB	SB	Def	VW
by instance	1.00	10.14	18.60	41.41
by class	1.00	21.91	18.99	43.65

Predicted Times

	Split by instance				Split by class			
Setup	f2f	f2fsr	lipb	combi	f2f	f2fsr	lipb	combi
pairwise combined + sw + cl	5.68	5.95	**5.18**	5.41	14.39	14.58	13.75	**12.45**
AUTOFOLIO (1hr)	20.33	19.90	**19.28**	21.21	21.82	**20.01**	**20.01**	21.87
AUTOFOLIO (2hrs)	20.01	18.79	19.48	**18.33**	22.99	25.19	**17.17**	21.57

Table PAR10 runtimes including feature extraction, as a multiple of the Virtual Best time

Results and Findings

Feature importance

Permutation Feature Importance

The Permutation Feature Importance of feature *F* is the extra time it would take to run a test set based on encodings selected when column *F* has its values permuted randomly across all rows in the test set.

- PFI is calculated at prediction time, rather than at training time, as with impurity-based feature importance measures
- more appropriate at showing features which lead to predictions that generalise better
- BUT importance can still be masked by another closely related feature

Feature Importances

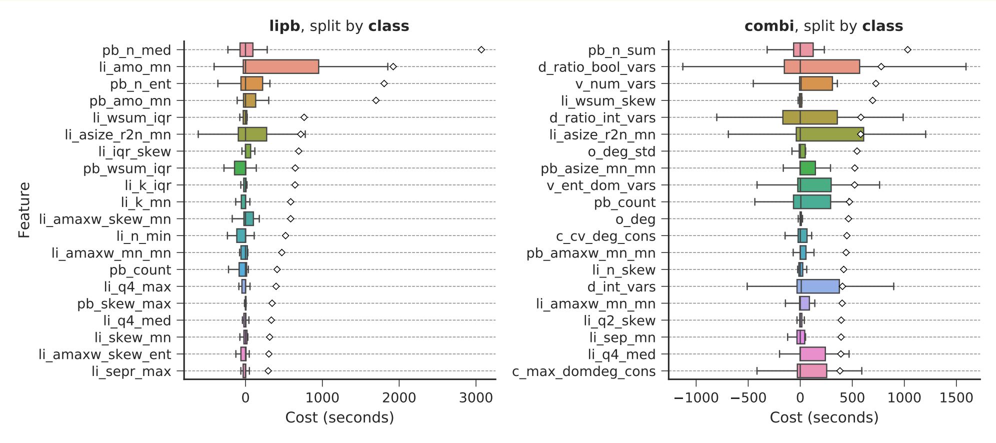


Figure Increase in PAR2 time for each permuted feature over 50 test sets. Top 20 features shown (by mean importance). Outliers are not shown. Features beginning `li_` or `pb_` are from *lipb*; the other feature names refer are generic instance features from *combi*.

Conclusion

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

References

Amadini, R., Gabbrielli, M., and Mauro, J. (2014).
An enhanced features extractor for a portfolio of constraint solvers.
In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 1357–1359, New York, NY, USA. Association for Computing Machinery.

Bofill, M., Coll, J., Suy, J., and Villaret, M. (2019).
SAT encodings of pseudo-boolean constraints with at-most-one relations.
In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 112–128. Springer.

Lindauer, M., Hoos, H. H., Hutter, F., and Schaub, T. (2015).
AutoFolio: An Automatically Configured Algorithm Selector.
Journal of Artificial Intelligence Research, 53:745–778.

Nightingale, P., Akgün, Ö., Gent, I. P., Jefferson, C., Miguel, I., and Spracklen, P. (2017).
Automatically improving constraint models in Savile Row.
Artificial Intelligence, 251:35–61.

Thank you, Questions

Thanks

- Grant EP/R513386/1 from the UK EPSRC
- Reviewers along the way
- ACP for helping me get here via the doctoral programme
- Audience

Any Questions?