# IndiCon: Selecting SAT Encodings for Individual Pseudo-Boolean and Linear Integer Constraints

Based on work in PhD thesis

**Felix Ulrich-Oltean**, Peter Nightingale, James Walker

October 2024

# Outline

Constraint Programming + Boolean Satisfiability

Learning to Select Encodings

Results and Observations

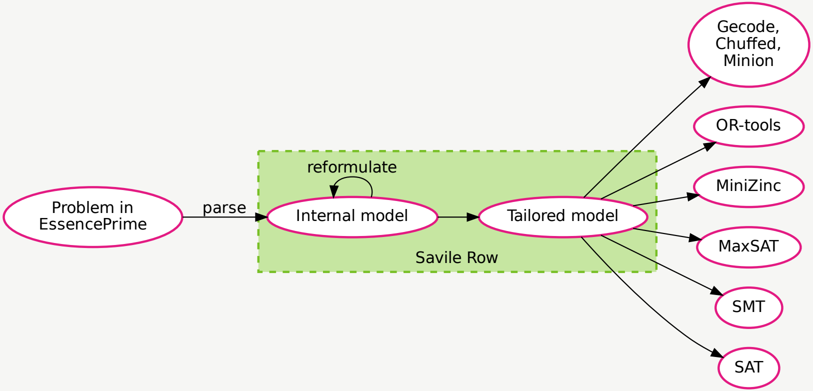
# Constraint Programming + Boolean Satisfiability

# Nurse Scheduling, an Example CSP

```
given n_nurses, n_days, n_sh_types : int
given covers : matrix indexed by [int(1..n_days*n_sh_types)] of int
given prefes : matrix indexed by [int(1..n_nurses*n_days*n_sh_types)] of int
given ub : int

find alloc : matrix indexed by [int(1..n_nurses*n_days)] of int (1..n_sh_types) such that

$ enough nurses are allocated per shift
forAll d : int(1..n_days).
  forAll st : int(1..n_sh_types).
    sum([alloc[(n-1)*n_days+d]=st | n : int(1..n_nurses)]) >=covers[(d-1)*n_sh_types+st],

$ each nurse is allocated to 5 shifts
forAll n : int(1..n_nurses).
  sum([alloc[(n-1)*n_days+d]!=n_sh_types | d : int(1..n_days) ]) = 5,

$ penalise violation of nurses' preferences
(sum n : int(1..n_nurses).
   sum d : int(1..n_days).
     sum st : int(1..n_sh_types).
       (alloc[(n-1)*n_days + d]=st) * prefes[(n-1)*n_days*n_sh_types + (d-1)*n_sh_types + st]) <= ub
```

An EssencePrime model for the nurse scheduling problem. Constraint models represent problems in terms of decision variables and rules limiting their allowed values.

# Solving CSP with Savile Row




Using Savile Row to reformulate and solve CSPs using various back-end solvers

# Why SAT? Effective Back-end Solver

| | MiniZinc Challenge 2024 | | | XCSP Comp. 2024 |
|---|---|---|---|---|
| Rank | Fixed | Free | Parallel | Main CSP |
| 1 | OR-tools CP-SAT | OR-tools CP-SAT | OR-tools CP-SAT | Picat |
| 2 | Choco CP-SAT | PicatSAT | PicatSAT | CPMpy-ortools |
| 3 | SICStus | iZplus | Choco CP | Fun-sCOP (cadical) |

Constraint solving competition results from `https://www.minizinc.org/challenge.html` and `https://www.xcsp.org/competitions/`

# Encoding CSP to SAT

To use a SAT solver, the CSP has to be encoded as Boolean formula, usually in conjunctive normal form (CNF)

- SAT variables and clauses for each integer decision variable
- clauses (and potentially extra variables) for constraints

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

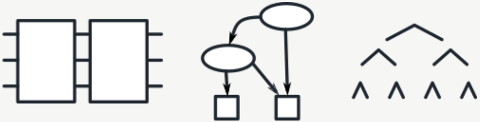
The beginning of CNF output from Savile Row for a simple knapsack problem

# Learning to Select Encodings

# Encoding Pseudo-Boolean and Linear Integer Constraints

Savile Row has 9 encodings for PBs (and therefore LIs)

- *GSWC* models a circuit which sequentially adds the weights
- *MDD* use multi-valued decision diagrams
- *Tree, GGT, GGTh, RGGT* are based on the totalizer tree-based approach
- *GGPW, GLPW* are based on sorting and bit arithmetic
- *GMTO* uses mixed-radix arithmetic



# LeaSE-PI

Our previous work LeaSE-PI (CP2022, *Constraints*)

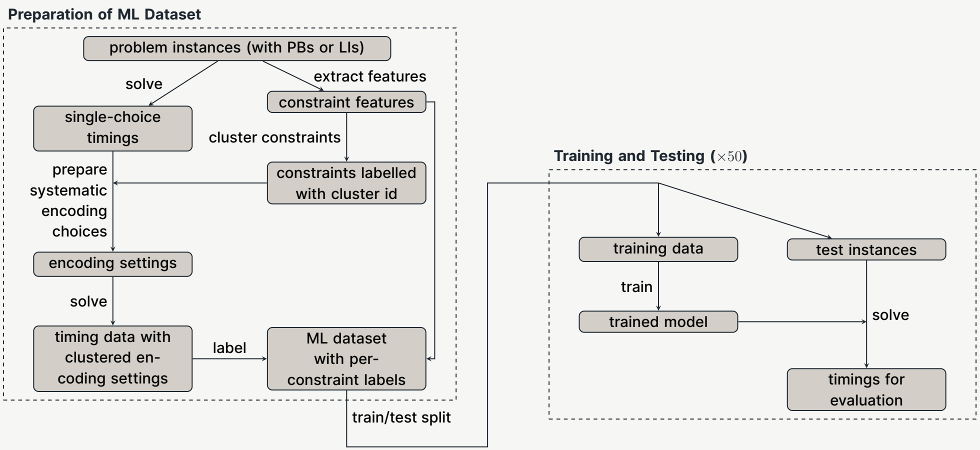
- learns to select encodings per problem instance for PB and LI constraints
- can train/test on known problem classes but also performs well on unseen problem classes
- selects PB/LI constraints together, first reducing the options to a smaller portfolio (81 down to 6)

# LeaSE-IndiCon

In this work, we learn to select potentially different encodings for each individual constraint in a problem instance. Why?

- Intuitively, there may be constraints of the same type but with very different characteristics within a single problem instance.
- Could overall performance be better if ML is allowed to select at this more fine-grained level?

# An Overview of IndiCon

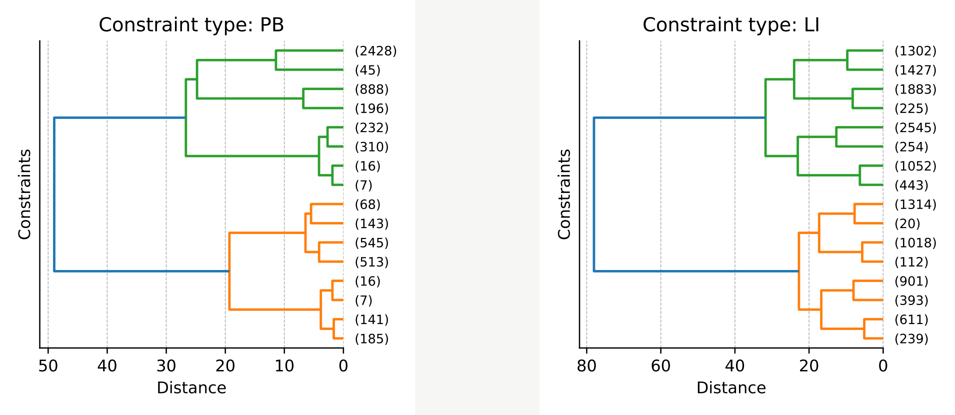



A summary of the steps involved in IndiCon

# Individual Constraint Features for PB and LI

| | |
|---|---|
| n | Number of terms |
| wsum | Sum of coefficients |
| q0, q2, q4, iqr | Mininum, median, maximum, IQR of coefficients |
| skew | Coefficients’ quartile skew |
| sepw | Number of distinct coefficient values |
| sepwr | Ratio of distinct coefficient values to number of coefficients |
| is_equality | Is it an equality constraint? |
| k | Right-hand side $k$ of the constraint |
| amogs | Number of At-Most-One groups (AMOGs) |
| amog_size_mn | Mean size of AMOGs |
| amog_size_mn_r2n | Mean AMOG size $\div$ number of terms |
| amog_maxw_med | Median size of the maximum coefficient across AMOGs |
| amog_maxw_mn | Mean size of the maximum coefficient across AMOGs |
| amog_maxw_mn2k | The ratio of amog_maxw_mn : k |
| amog_maxw_sum | Sum of the maximum coefficients in each AMOG |
| amogs_maxw_skew | Skew of the maximum coefficient in AMOGs |
| amog_maxw_sum_k_prod | amog_maxw_sum $\times k$ |

# Clustering Constraints




Dendrograms showing agglomerative clustering by constraint features. The x-axis shows the Euclidean distance between clusters. On the y-axis labels indicate the number of data points in a branch.

# Results and Observations

# Problem Corpus

A selection of the problem classes in the corpus, with the number of instances ($n$) and the mean number of PB and LI constraints ($\bar{c}$) per instance

| | | $\bar{c}$ | | | | $\bar{c}$ | |
|---|---|---|---|---|---|---|---|
| Problem | $n$ | PB | LI | Problem | $n$ | PB | LI |
| killerSudoku2 | 50 | 2473 | 194 | efpa | 20 | 244 | 0 |
| nurse-sched | 50 | 207 | 0 | handball7 | 20 | 894 | 1809 |
| carSequencing | 49 | 1024 | 0 | mrcpsp-pb | 20 | 100 | 62 |
| knights | 44 | 255 | 505 | n-queens | 20 | 1859 | 0 |
| langford | 39 | 231 | 0 | bibd | 19 | 537 | 0 |
| opd | 33 | 36 | 103 | molnars | 17 | 0 | 6 |
| knapsack | 24 | 1 | 1 | briansBrain | 16 | 0 | 1 |
| sonet2 | 24 | 10 | 1 | life | 16 | 0 | 786 |
| immigration | 23 | 0 | 1 | n-queens2 | 16 | 361 | 0 |
| bibd-implied | 22 | 651 | 0 | bpmp | 14 | 21 | 0 |

# Runtimes

IndiCon performance for the best 3 setups for PB and LI constraints, ordered from best to worst performing. Each setup is tested over 50 train/test splits. Performance is measured using PAR10 and shown as a multiple of the Virtual Best* time.

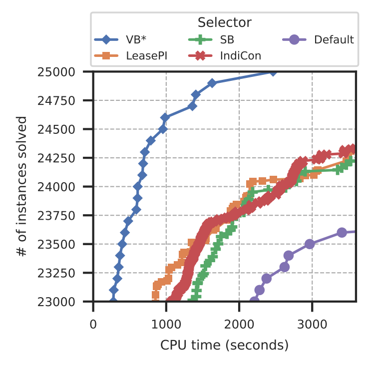
IndiCon for PB

| Setup | | Runtime |
|---|---|---|
| Clusters | Classifier | PAR10 VB* |
| 1 | RF | 5.57 |
| 1 | DT | 5.69 |
| 5 | GB | 8.10 |
| *Single Best* | | *11.58* |

IndiCon for LI

| Setup | | Runtime |
|---|---|---|
| Clusters | Classifier | PAR10 VB* |
| *Single Best* | | *4.53* |
| 6 | RF | 6.44 |
| 1 | RF | 6.70 |
| 6 | GB | 11.12 |

# Results When Setting Both PB and LI




Instances solved in given CPU time by single-choice virtual best (VB*), single best (SB), default encoding (Def), best LeaSE-PI and IndiCon setups

- IndiCon for 250 instances in corpus with both PBs and LIs
- Random sample ($\times 100$) of test runs from LeaSE-PI and IndiCon
- IndiCon slightly better on harder instances (around 3000 s)

# Advantages and Challenges

On the plus side:

- More flexible and potentially better performing (for PBs in our case) than one choice per instance
- IndiCon more than matches state of the art performance on unseen problem classes when setting PB and LI together
- IndiCon scales well; any type of constraint could be addressed
- Simple and explainable ML models competitive (for PB)

Challenges:

- LI selection underperforms single best
- Range of SAT encodings also exist for other constraint types, feature calculation could be challenging for some, e.g. AMO

# Thank you

**Any Questions?**

Do chat afterwards or get in touch:

- felix.ulrich-oltean@york.ac.uk
- felixvuo.github.io