ENIGMAWatch: ProofWatch Meets ENIGMA

Zarathustra Goertzel, Jan Jakubův, and Josef Urban

Czech Technical University in Prague

TABLEAUX 2019

Learning From Mizar Proofs

We work with

- Mizar Mathematical Library (MML) contains 1148 articles:
  - including *Bolzano-Weierstrass* and *Gödel's completeness theorem*
- An interactive theorem proving system
- Evaluate on 5000 (out of 57897) Mizar theorems and top-level lemmas

Learning From Mizar Proofs

- MML contains **De Morgan's laws** in Boolean algebra, and the related inequalities in Heyting algebras

```
theorem :: WAYBEL_1:85
 for H being non empty lower-bounded RelStr st H is Heyting holds
 for a, b being Element of H holds 'not' (a "/\" b) >= ('not' a) "\/" ('not' b)
```

$$\text{for } a, b \in H,\ \neg(a \wedge b) \geq \neg a \vee \neg b$$

```
theorem Th36: :: YELLOW_5:36
 for L being non empty Boolean RelStr
 for a, b being Element of L holds
 ( 'not' (a "\/" b) = ('not' a) "/\" ('not' b) & 'not' (a "/\" b) = ('not' a) "\/" ('not' b) )
```

$$\text{for } a, b \in L,\ \neg(a \vee b) = \neg a \wedge \neg b$$

$$\neg(a \wedge b) = \neg a \vee \neg b$$

Learning From Mizar Proofs

We work with

- Mizar Mathematical Library (MML)
  - An Interactive Theorem Proving system
  - Evaluate on 5000 (out of 57897) Mizar theorems and top-level lemmas
- E prover
  - Saturation based automated theorem prover (ATP)
  - Can be a hammer for interactive theorem proving (ITP)
  - Uses Mizar in first order formula (FOF) form
  - We learn from E proof clauses in conjunctive normal form (CNF)

Results Overview

On our 5000 problem benchmark, E proves:

• *Baseline strategy*:	1140
• *ProofWatch (symbolic learning)*:	1356 (+19%)
• *ENIGMA (statistical learning)*:	1557 (+37%)
• *ENIGMAWatch (combined learning)*:	1694 (+49%)

Outline of talk

- Brief overview of E prover.

- ENIGMA (Efficient learNing-based Inference Guiding Machine)
- ProofWatch: Dynamic Watchlist Guidance
- ENIGMAWatch: ProofWatch → Enigma

- Experiments + Results
- Conclusion

E Prover (a Saturation-based ATP)

- **Goal: Prove conjecture from premises.**
- **E has two sets of clauses:**
  - *Processed* clauses P (initially empty)
  - *Unprocessed* clauses U (Negated Conjecture and Premises)
- **Given Clause Loop:**
  - Select '*given clause*' g to add to P
  - Apply *inference rules* to g and all clauses in P
  - Process new clauses. Add non-trivial and non-redundant ones to U.
- **Proof search succeeds when empty clause is inferred.**
- **Proof consists of some of the given clauses.**

Given Clause Loop in E

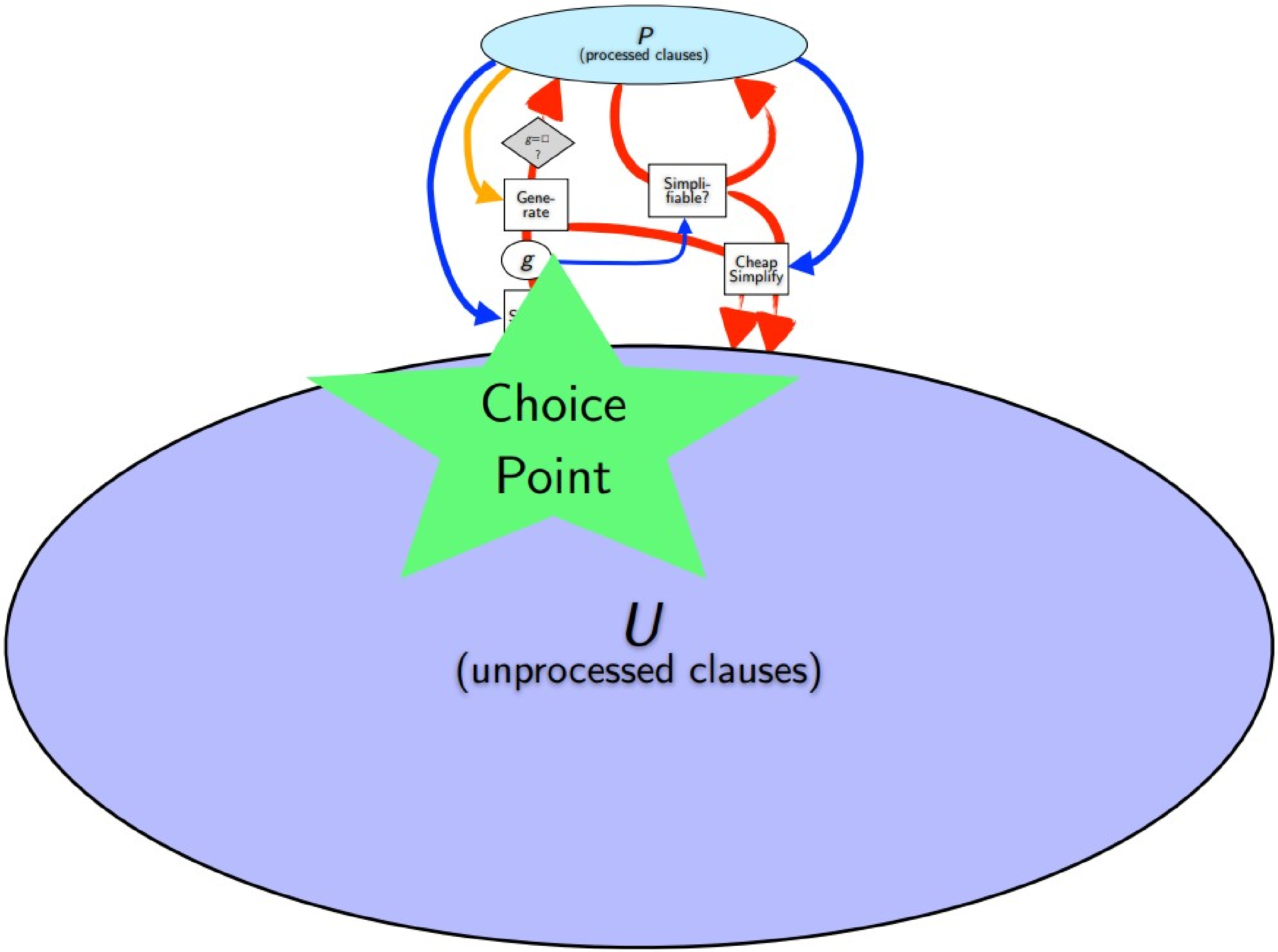


Image thanks to Stephan Schulz

E Strategies

- Consist of *Clause Evaluation Functions*:
  - *Priority functions*: partition clauses into priority queues.
    - e.g., *PreferUnit*, *ConstPrio*
  - **Weight functions**: order clauses in queues based on a score.
    - e.g.: **Clauseweight**, **FIFOWeight**
- Weighted by frequency of use, for example:

-H(2***Clauseweight**(*PreferWatchlist*,20,9999,4)
,4***FIFOWeight**(*PreferUnit*))

Learning Given Clause Selection

ENIGMA

- Statistical Learning
- Learns from given-clauses
- *Positive* and *Negative*
- Maps clauses to vectors
- Weight function
- No proof state
- Ranks all clauses

ProofWatch

- Symbolic Learning
- Learns from given-clauses
- *Positive only* (proof clauses)
- Uses clauses as is
- Priority function
- Yes proof state
- Only ranks some clauses

ENIGMA

- Use statistical machine learner to select given clauses

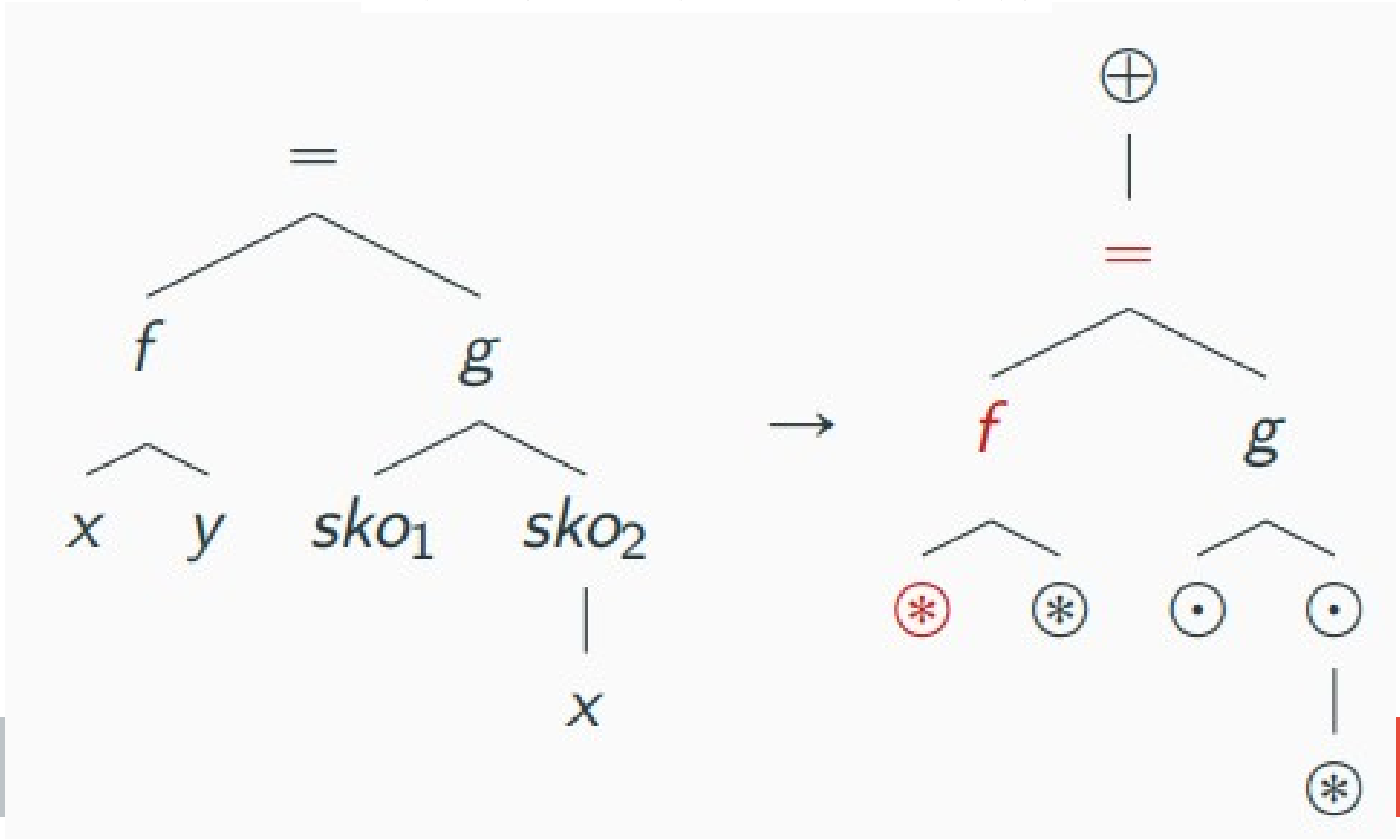
- Input:
  - Positive examples + conjecture features
  - Negative examples + conjecture features

- Output:
  - (Fast) model to predict whether (clause, conjecture) pairs are *positive* or *negative*.

Clauses → Vectors

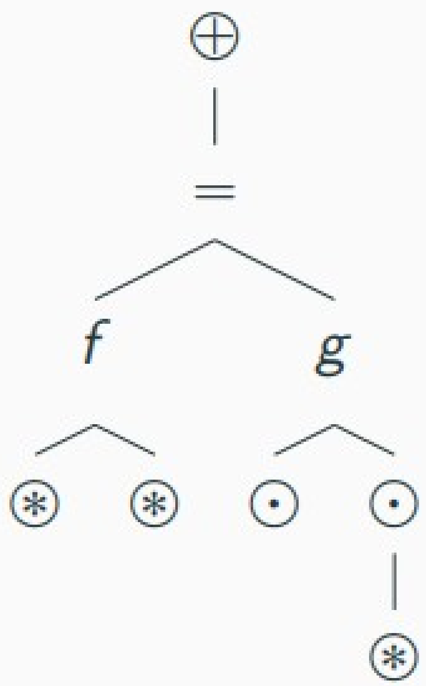
- Treat clauses as trees. Abstract vars and skolem symbols
- *Vertical Features* are descending paths of length 3

For example: $f(x, y) = g(\mathrm{sko}_1, \mathrm{sko}_2(x))$



Clauses → Vectors

- Enumerate features → $\mathbb{R}^{|\text{Features}|}$
- Count features in a clause for its vector



#	feature	count
1	$(\oplus,=,a)$	0
⋮	⋮	⋮
11	$(\oplus,=,f)$	1
12	$(\oplus,=,g)$	1
13	$(=,f,\circledast)$	2
14	$(=,g,\odot)$	2
15	$(g,\odot,\circledast)$	1
⋮	⋮	⋮

Feature Types

- **Vertical** :- top-down tree-walks
- **Horizontal** :- cuts of term tree
- **Symbol** :- occurrence/depth statistics
- **Length** :- clause length, #pos/neg literals
- . . .

Feature Vector Hashing

- Feature vectors on MML exceed 1,000,000
- So we reduce the size to 32,768 (2^{15})
- by adapting a string hash function from SDBM project

ENIGMA

- Train statistical learner to select given-clauses

- **Enumerate feature map *π*: feature → R**
- Input:
  - Positive examples + conjecture features
  - Negative examples + conjecture features

- Output:
  - Model M to predict whether (clause, conjecture) pairs are *positive* or *negative*.

ENIGMA Weight Function

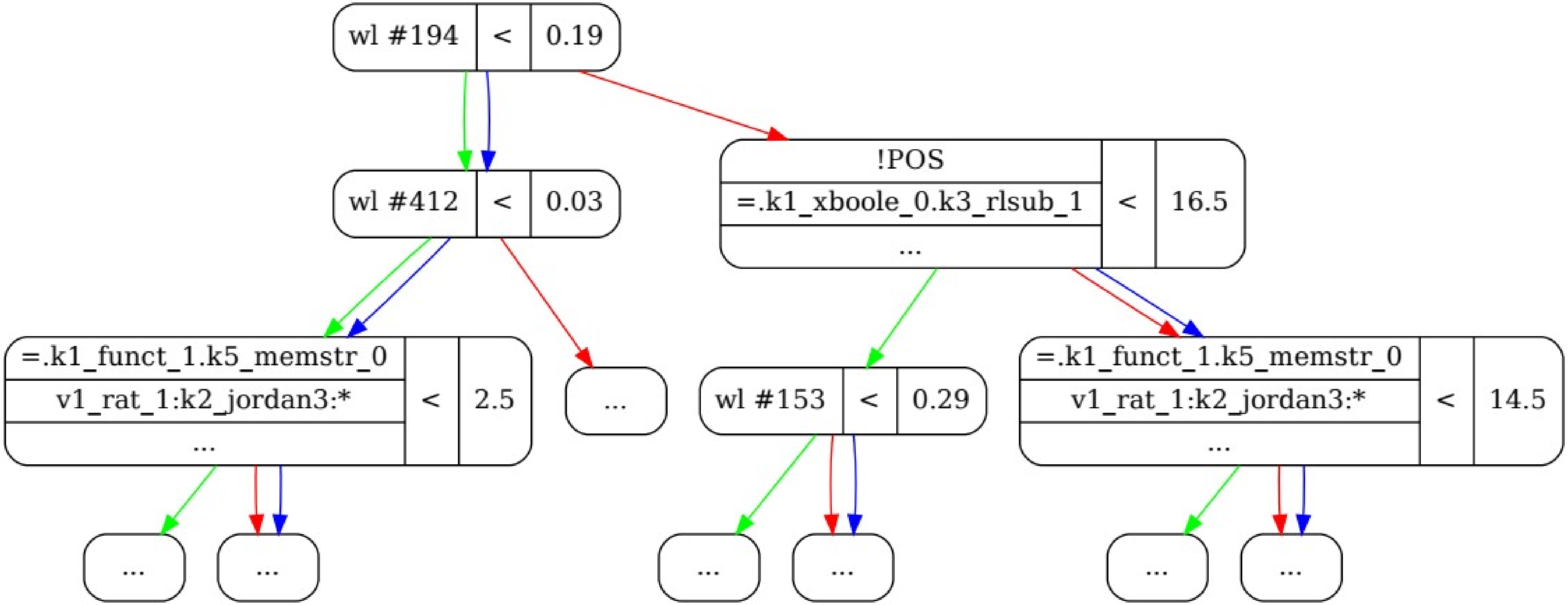
- Feature vector $\varphi = (\varphi_C, \varphi_G)$
  - $\varphi_C = \pi(\text{clause})$
  - $\varphi_G = \pi(\text{conjecture})$
- *weight*(C) = 1 if $M(\varphi) > 0.5$ else 10
- It would be good to include the proof-state in φ.

ENIGMA’s Machine Learner

We currently use **XGBoost**, a gradient boosted tree algorithm that

- learns k decision trees to classify data
- sums the k trees’ decisions to determine the ensemble estimate
- maintains a histogram of the features to choose splitting points for creating trees

XGBoost Example Tree



Watchlists

- A ***watchlist*** is a set of clauses loaded into the ATP.
- Logical subsumption is used to check the watchlist.
- For example:
  - Let W $= brother(zar, Y) \vee \neg uncle(zar)$
  - Let C $= brother(X, Y)$
  - Then C $\sqsubseteq$ W (with X = zar)
  - We say clause C matches the watchlist if it subsumes a clause on the watchlist.

Brief Watchlist History

1. Hint list used by Bob Veroff (96)
   - In Prover9 and Otter (ATPs).
   - Has proven extensions of AIM conjecture (Abelian Inner Mapping) in loop theory.
   - Enabled very long proofs (1000+ steps)
2. E's watchlist mechanism implemented by Stephan Schulz.
   - Uses a priority function: *PreferWatchlist*
   - All clauses that match a watchlist are selected first.
   - Works with any E weight function.

ProofWatch (static)

- Uses E's watchlist feature.
- Loads proof clauses onto watchlist:
  - Positive examples only.
- Used via *PreferWatchlist*.
- All *matched clauses* given **the same** priority.

ProofWatch (dynamic)

- Extends E's watchlist feature to multiple watchlists.
- Loads *k* proofs onto *k* watchlists.
- Counts matches to each watchlist during proof-search
  - *progress*(W)
- Assumption: completion ratio (*progress*(W_i)/|W_i|) approximates relevance of W_i's proof to conjecture.

$$relevance(C) = \max_{W \in \{W_i : C \sqsubseteq W_i\}} \left(\frac{progress(W)}{|W|} \right)$$

ProofWatch (dynamic)

- Loads *k* proofs onto *k* watchlists.
- Counts matches to each watchlist during proof-search
  - *progress*(W)
- Assumption: completion ratio (*progress*(W_i)/|W_i|) approximates relevance of W_i's proof to conjecture.

$$relevance(C) = \max_{W \in \{W_i : C \sqsubseteq W_i\}} \left(\frac{progress(W)}{|W|} \right)$$

- **Boosts priority** as a function of relevance.
- Used with *PreferWatchlistRelevant.*

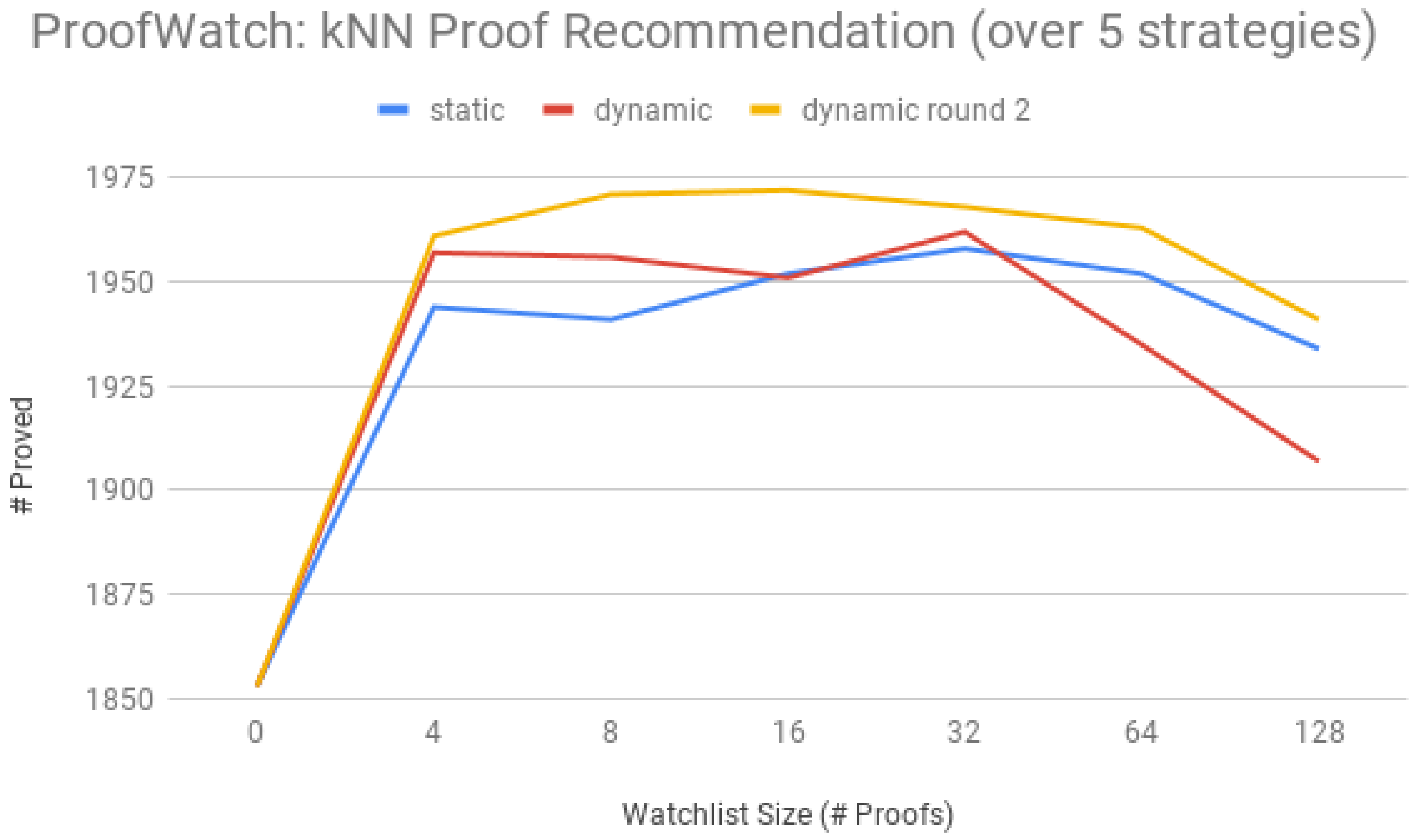
Watchlist Curation

In the ProofWatch paper we

1. Used E proofs from the conjecture’s Mizar article.

2. Used Enigma features with k-NN (k nearest neighbors) to recommend similar proofs.

ProofWatch Results



Proof Vector

A snapshot of the proof-vector for YELLOW 5:36 with 32 k-NN recommended proofs:

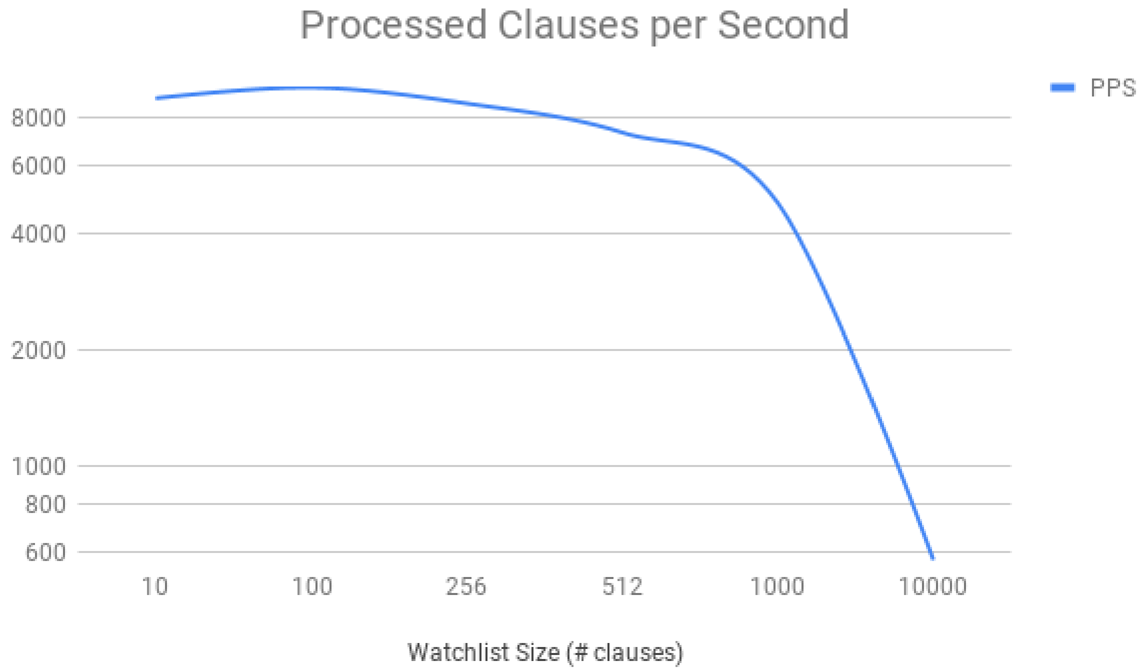
0	0.438	42/96	1	0.727	56/77	2	0.865	45/52	3	0.360	9/25
4	0.750	51/68	5	0.259	7/27	6	0.805	62/77	7	0.302	73/242
8	0.652	15/23	9	0.286	8/28	10	0.259	7/27	11	0.338	24/71
12	0.680	17/25	13	0.509	27/53	14	0.357	10/28	15	0.568	25/44
16	0.703	52/74	17	0.029	8/272	18	0.379	33/87	19	0.424	14/33
20	0.471	16/34	21	0.323	20/62	22	0.333	7/21	23	0.520	26/50
24	0.524	22/42	25	0.523	45/86	26	0.462	6/13	27	0.370	20/54
28	0.411	30/73	29	0.364	20/55	30	0.571	16/28	31	0.357	10/28

Proof Number

Completion Ratio

Multi-index Subsumption

- 32 proofs is pretty small, right?
- E crawled to a halt with more than 4000 clauses or 128 proofs on the watchlist



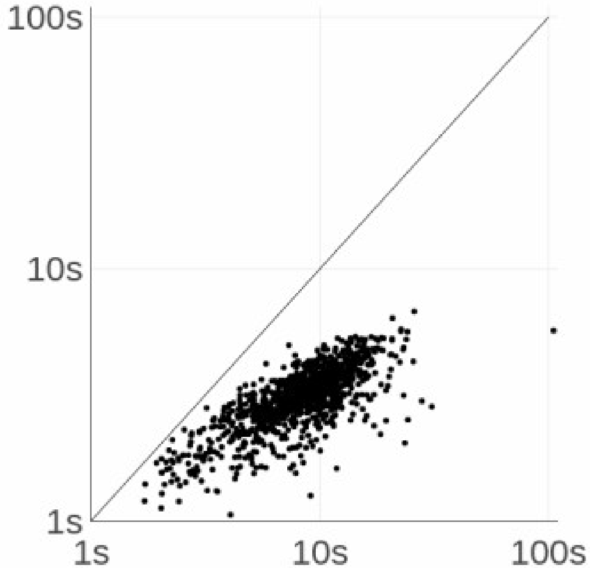
Multi-index Subsumption

- Define *code*(C) = {top-level predicate symbols)
  - *code*("$P(a) \vee \neg P(b) \vee P(f(x))$") = $\{+P, -P\}$

- *Given clauses C and D*, $C \sqsubseteq D$ *implies* $\mathrm{code}(C) \subseteq \mathrm{code}(D)$.

- Create an index for each clause code in the watchlist.
- Given clause C, check subsumption in each index whose code contains *code*(C).

Multi-index Subsumption



	runtime (left graph ←)		
	single	multi	speedup
avg	9.23s	3.16s	2.9×
best	105.3s	5.7s	18.5×
worst	2.26s	2.09s	1.08×
	subsumptions (right →)		
	single	multi	reduction
avg	2328k	52k	44.1×
best	3059	1	3059×
worst	709k	367k	1.9×

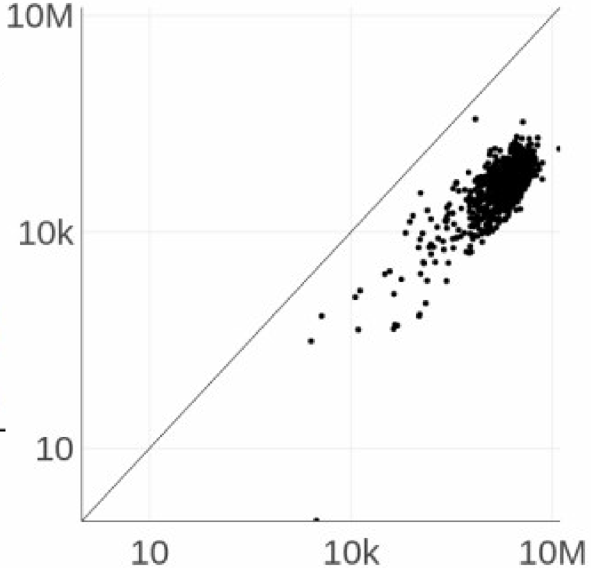


Table 2. Evaluation of multi-indices subsumption indexing.

ENIGMAWatch

Idea: ProofWatch's proof-vector can capture some proof-state information. Give this to ENIGMA.

- Feature vector $\varphi = (\varphi_C, \varphi_G, \varphi_\pi)$
  - $\varphi_C = \pi(\text{clause})$
  - $\varphi_G = \pi(\text{conjecture})$
  - φ_π = proof-vector of completion ratios

Challenge: ENIGMA needs uniform vector space for features to learn over "big data".

Mizar Mathematical Library (MML)

- 57,897 Mizar theorems and top-level lemmas
- Premises already selected

- Previously ENIGMAWatch was tested on the MPTP Challenge Benchmark:
  - The 252 Mizar lemmas used to prove Bolzano-Weierstrass theorem.

Proof Vector Construction

We want:

- Proofs that will be useful over the whole MML

Proof Vector Construction

We want:

- Proofs that will be useful over the whole MML

Step 1:

- Run E with 14,882 proofs loaded as watchlists
- For each Conjecture’s proof search,
  - For each given-clause,
    - For each watchlist proof,
      - How many proof-clauses were subsumed at the time **g** was selected?
    - **The proof-vectors of completion ratios**: $\varphi\Pi_g$

Proof Vector Construction

Step 1:

- Run E with 14,882 proofs loaded as watchlists
- For each Conjecture's proof search,
  - For each given-clause,
    - Proof-vectors: $\varphi\Pi_g$ (over the 15k proofs)

Step 2:

- Sum over given-clauses to obtain mean proof-vectors

$$\varphi\Pi_{\overline{C}} = \frac{1}{\#g}\sum_{g}\varphi\Pi_g$$

Proof Vector Construction

Step 2: $\varphi_{\Pi_{\overline{C}}} = \frac{1}{\#g} \sum_{g} \varphi_{\Pi_g}$

- Sum over given-clauses to obtain mean proof-vectors

Step 3:

- Choose the “best” 512 watchlists based on $\varphi_{\Pi_{\overline{C}}}$

Proof Vector Construction

Step 3:

- Choose the “best” 512 watchlists based on $\varphi_{\Pi_{\overline{C}}} = \frac{1}{\#g} \sum_{g} \varphi_{\Pi_g}$

Methods: Stack $\varphi_{\Pi_{\overline{C}}}$ into matrix M

- **Mean**: mean proof-vector across rows, i.e., $\max_{W_i}(\frac{1}{\#C} \sum_{C} \varphi_{\Pi_{\overline{C}}})$
- **Var**: compute variance of each watchlist W_i over conjectures
- **Corr**: find least correlated proofs W_i by computing Pearson correlation matrix of M^T
- **Rand**: randomly select 512 watchlists to use

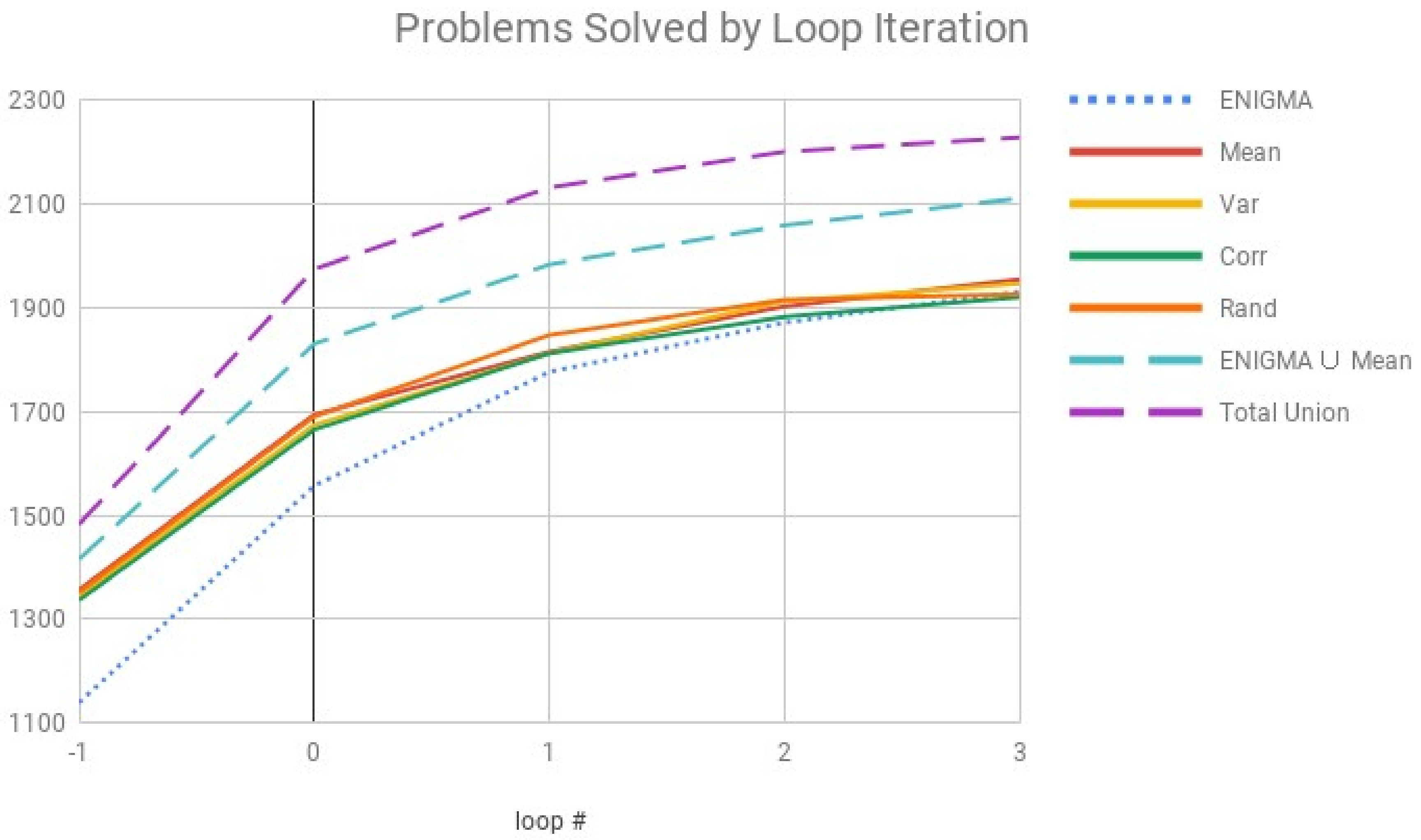
Experiments

- **Baseline** is the strongest ProofWatch strategy so far.

- The time limit is 60 seconds
- With a 30,000 generated clause limit.
  - Which **Baseline** does can do in 10 seconds.
  - *Abstract time*

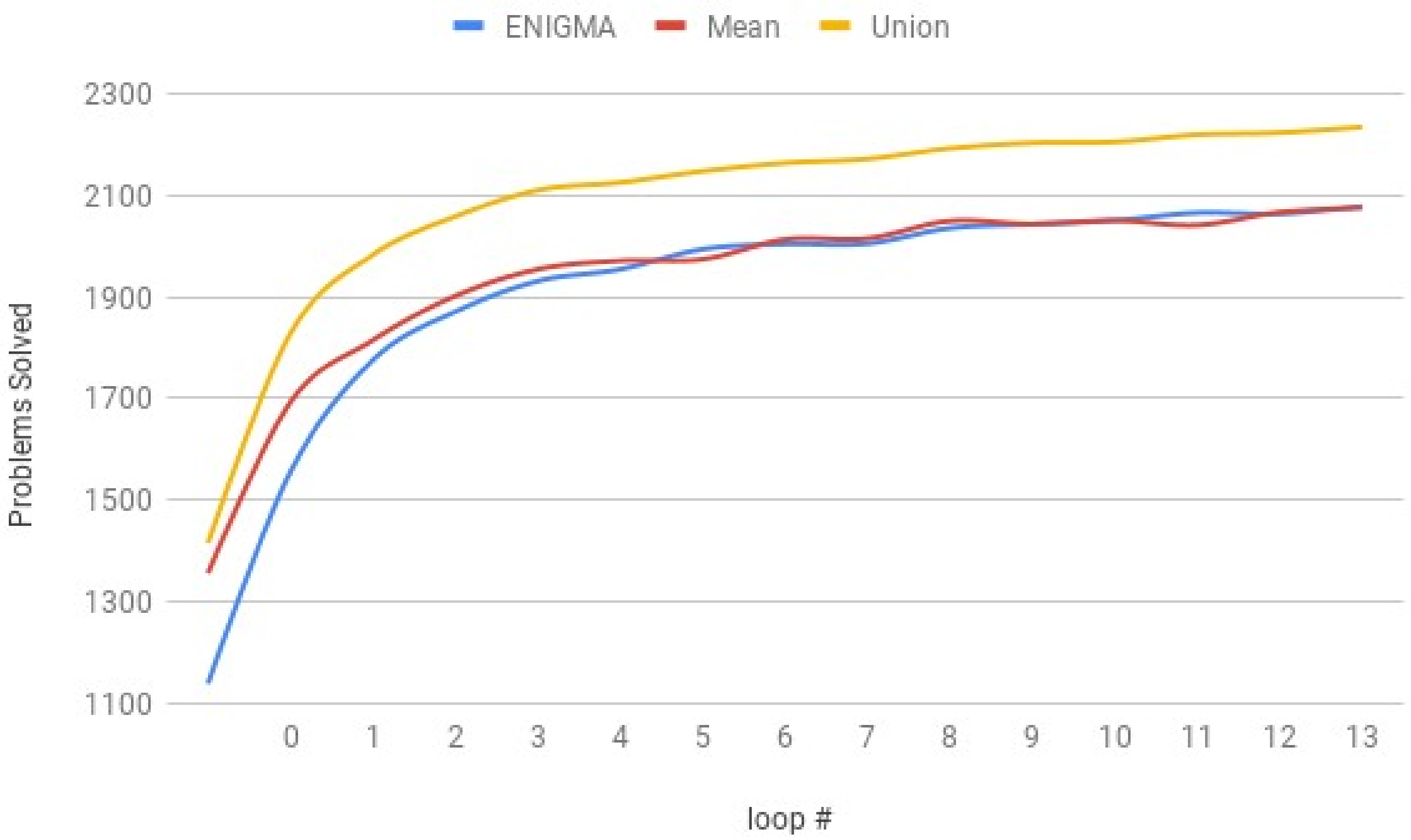
- Training and tests are done on 5000 problems from Mizar

Results



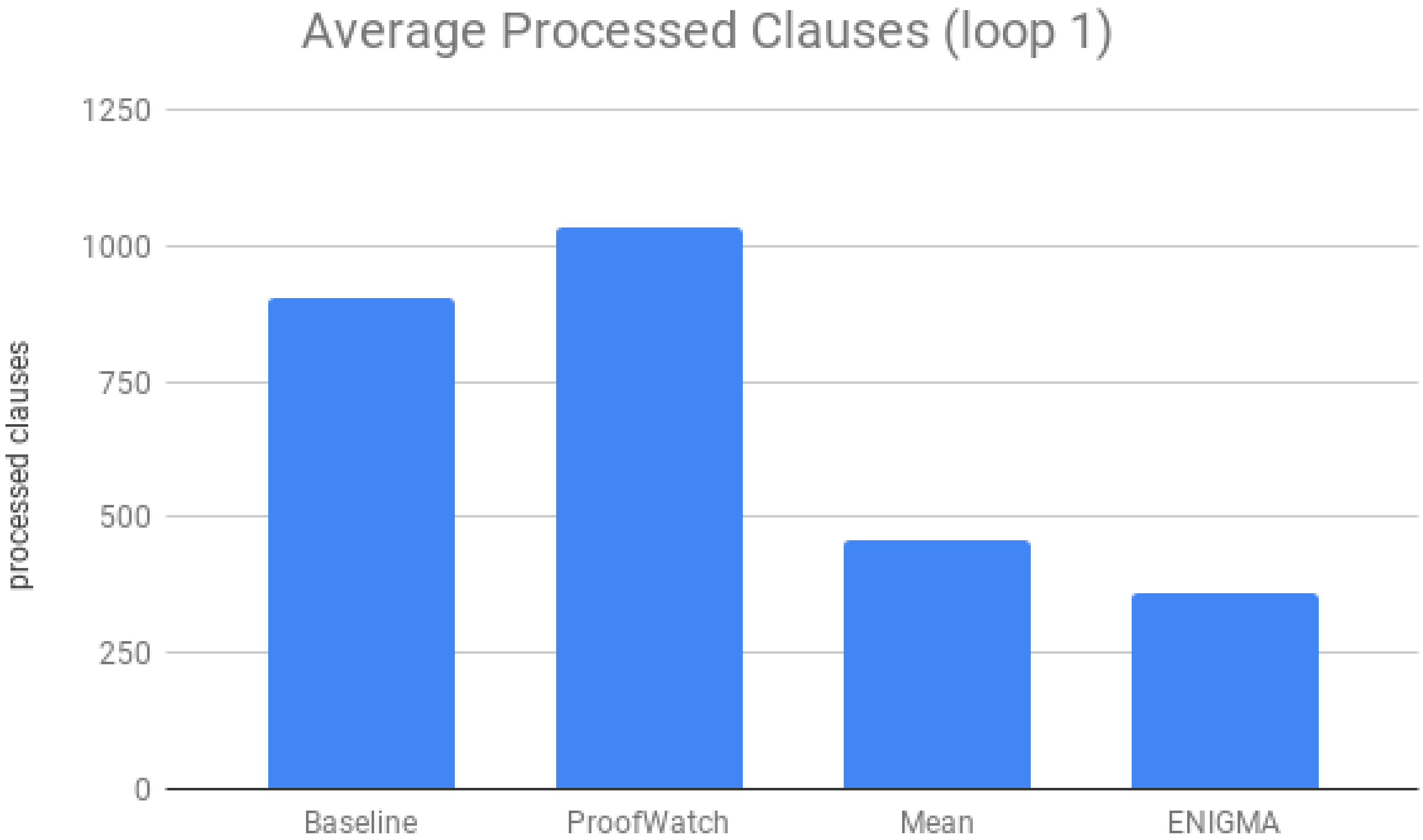
Problems Solved by Loop Iteration
2300
2100
1900
1700
1500
1300
1100
-1
0
1
2
3
loop #
ENIGMA
Mean
Var
Corr
Rand
ENIGMA ∪ Mean
Total Union

Results



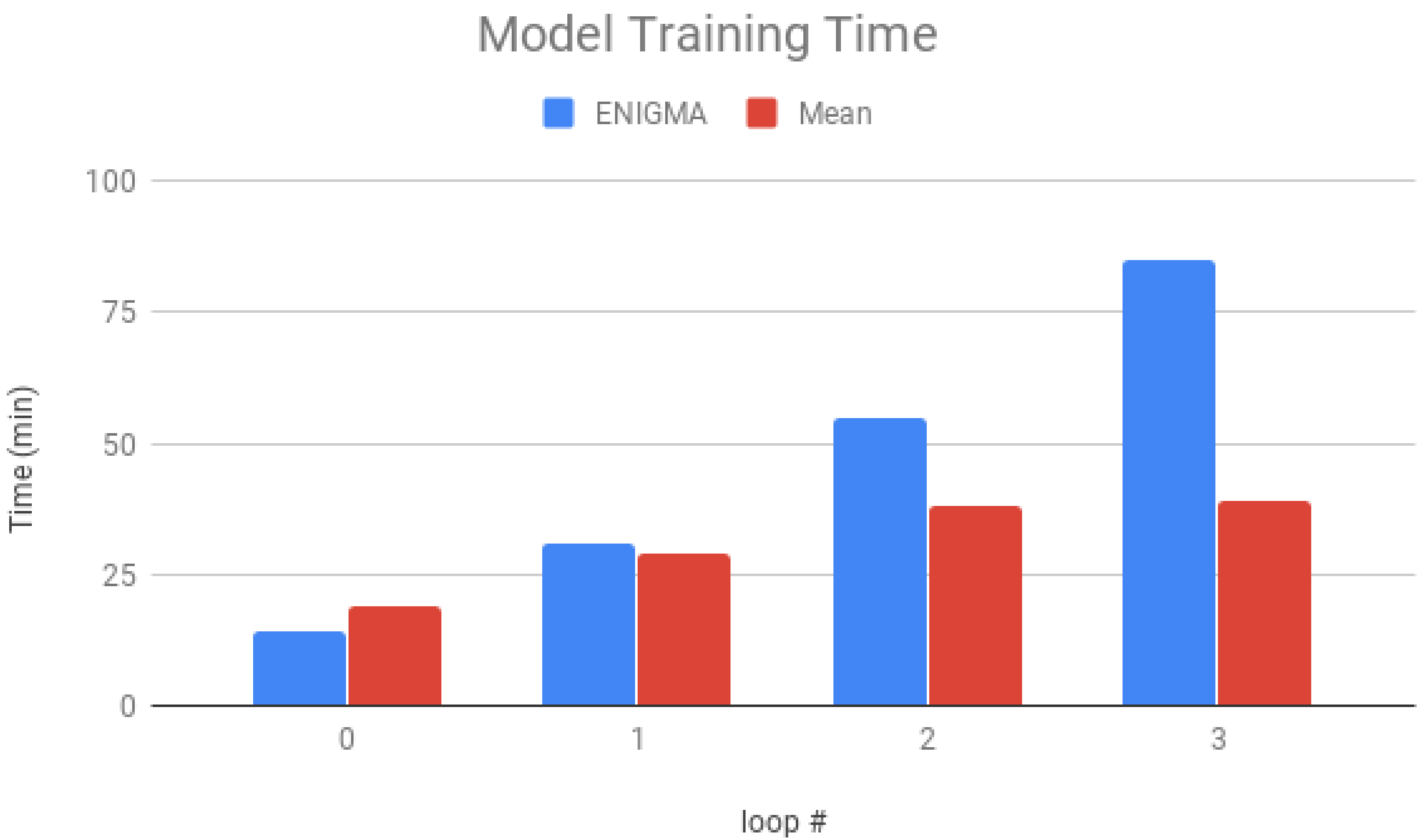
ENIGMA
Mean
Union
2300
2100
1900
1700
1500
1300
1100
Problems Solved
0
1
2
3
4
5
6
7
8
9
10
11
12
13
loop #

Results



Average Processed Clauses (loop 1)
1250
1000
750
500
250
0
processed clauses
Baseline
ProofWatch
Mean
ENIGMA

Results



Model Training Time
ENIGMA
Mean
100
75
50
25
0
Time (min)
0
1
2
3
loop #

Conclusion

- Feature Hashing and Multi-Index Subsumption allow ENIGMA and ProofWatch to be run on full MML with large watchlists.
- ENIGMAWatch:
  - Proves more problems than ENIGMA in early loops
  - Trains faster
  - Provides complementarity with ENIGMA (good for scheduling)
- Good paradigm of merging symbolic and statistical machine learning.