ProofWatch Meets Enigma: First Experiments

Zarathustra Goertzel, Jan Jakubův, and Josef Urban

Czech Technical University in Prague

IWIL'18

Outline of talk

- **Brief overview of saturation-based automated theorem provers (ATP)**
- **ENIGMA (Efficient learNing-based Inference Guiding Machine)**
- **ProofWatch: Dynamic Watchlist Guidance**
- **ENIGMAWatch: ProofWatch → Enigma**
- **MPTP Challenge: the dataset**
- **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 given clause.**

Given Clause Loop in E

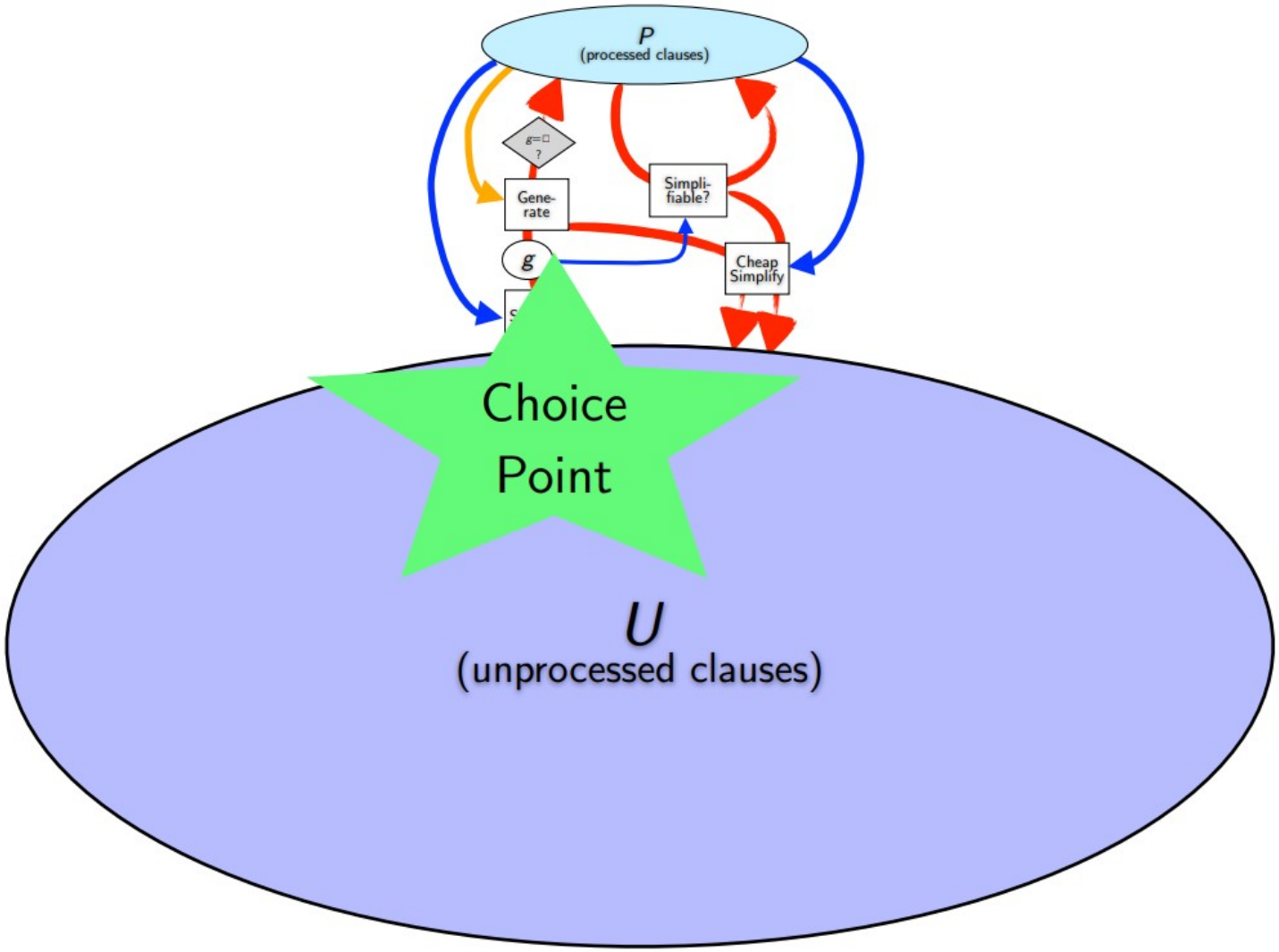


Image thanks to Stephan Schulz

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:

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

Approaches to Given Clause Selection

ENIGMA

- **Learns from given-clauses in E proof searches, i.e. positive and negative examples.**
- **Maps clauses to feature vectors.**
- **Uses *logistic regression* on clause and conjecture vectors.**
- **Weight function**
- **No proof state.**

ProofWatch

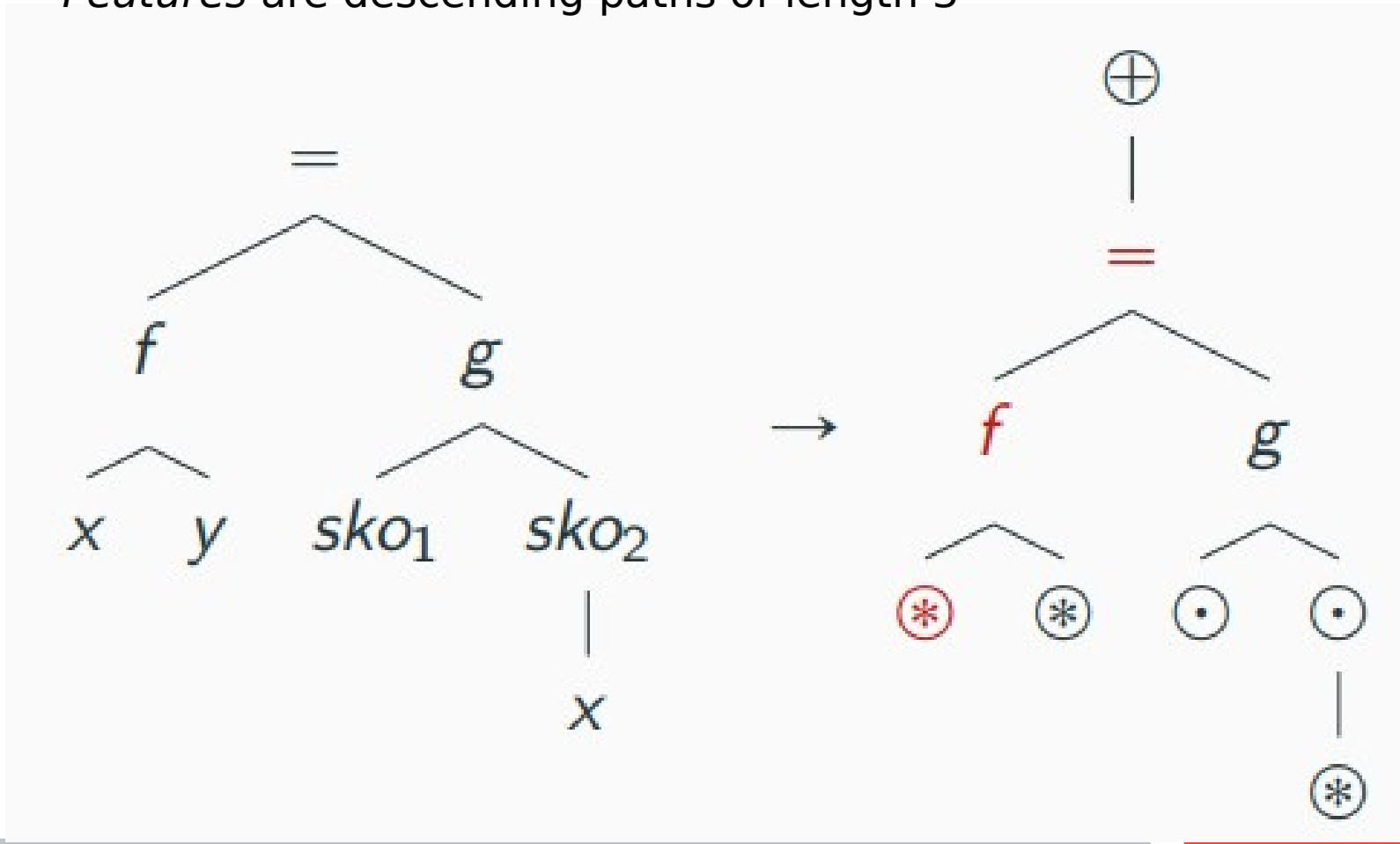
- **Learns from given-clauses in E proofs, i.e. only positive examples.**
- **Uses clauses as is.**
- **Uses *logical subsumption* on clause and watchlist (related proof) clause.**
- **Only ranks clauses that subsume watchlist.**
- **Priority function**
- **Yes proof state.**

ENIGMA

- Use linear classifier to select given clauses (LIBLINEAR)
- Input:
  - Positive examples + conjectures
  - Negative examples + conjectures
- Output:
  - Linear model that predicts whether clauses are *positive* or *negative*.

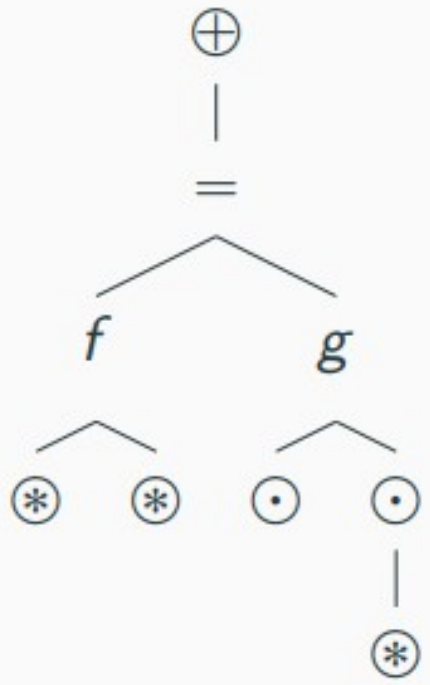
Clauses ⟶ Vectors

- Treat clause as tree. Abstract vars and skolem symbols
- *Features* are descending paths of length 3



Clauses ⟶ Vectors

Enumerate features (→ R^|Features| vector space)
Count features in a clause for its vector



#	feature	count
1	(⊕,=,a)	0
⋮	⋮	⋮
11	(⊕,=,f)	1
12	(⊕,=,g)	1
13	(=,f,⊛)	2
14	(=,g,⊙)	2
15	(g,⊙,⊛)	1
⋮	⋮	⋮

ENIGMA

- Use linear classifier to select given clauses (LIBLINEAR)
- Input:
  - Positive examples + conjectures
  - Negative examples + conjectures
- Enumerate feature map **π**: feature → R
- Output:
  - Linear model **w** that predicts whether clauses are *positive* or *negative*
- Final model: (**π, w**)..

Enigma Weight Function

- Feature vector $\varphi = (\varphi_C, \varphi_G)$
  - $\varphi_C = \pi(\text{clause})$
  - $\varphi_G = \pi(\text{conjecture})$

- $\mathit{weight}_0(C) = 1$ if $w \cdot \varphi > 0$ else 10
- $\text{weight}(C) = \text{weight0}(C) + \delta \cdot |C|$

- It would be to include the proof-state in φ.

Watchlists

- A ***watchlist*** is a set of clauses loaded into the ATP.
- Logical subsumption is used to check the watchlist.
- For example:
  - Let W_2 = cnf(c_0_57,plain, (k4_xboole_0(k1_xboole_0,X1)=k1_xboole_0)).
  - Let C = cnf(i_0_1611, plain, (k4_xboole_0(X1,X1)=k1_xboole_0)).
  - Then C $\sqsubseteq$ W (with X1 = k1_xboole_0)
  - We say clause C matches the watchlist.

Brief Watchlist History

1. Hint list used by Bob Veroff (96)
   - In Prover9 and Otter (ATPs).
   - Have proven extensions of AIM conjecture (Abelian Inner Mapping) in loop theory.
   - Enable very long proofs (1000+ steps)
2. E prover's watchlist mechanism implemented by Stephan Schulz.
   - Uses a priority function: *PreferWatchlist*
   - So 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*.

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.*

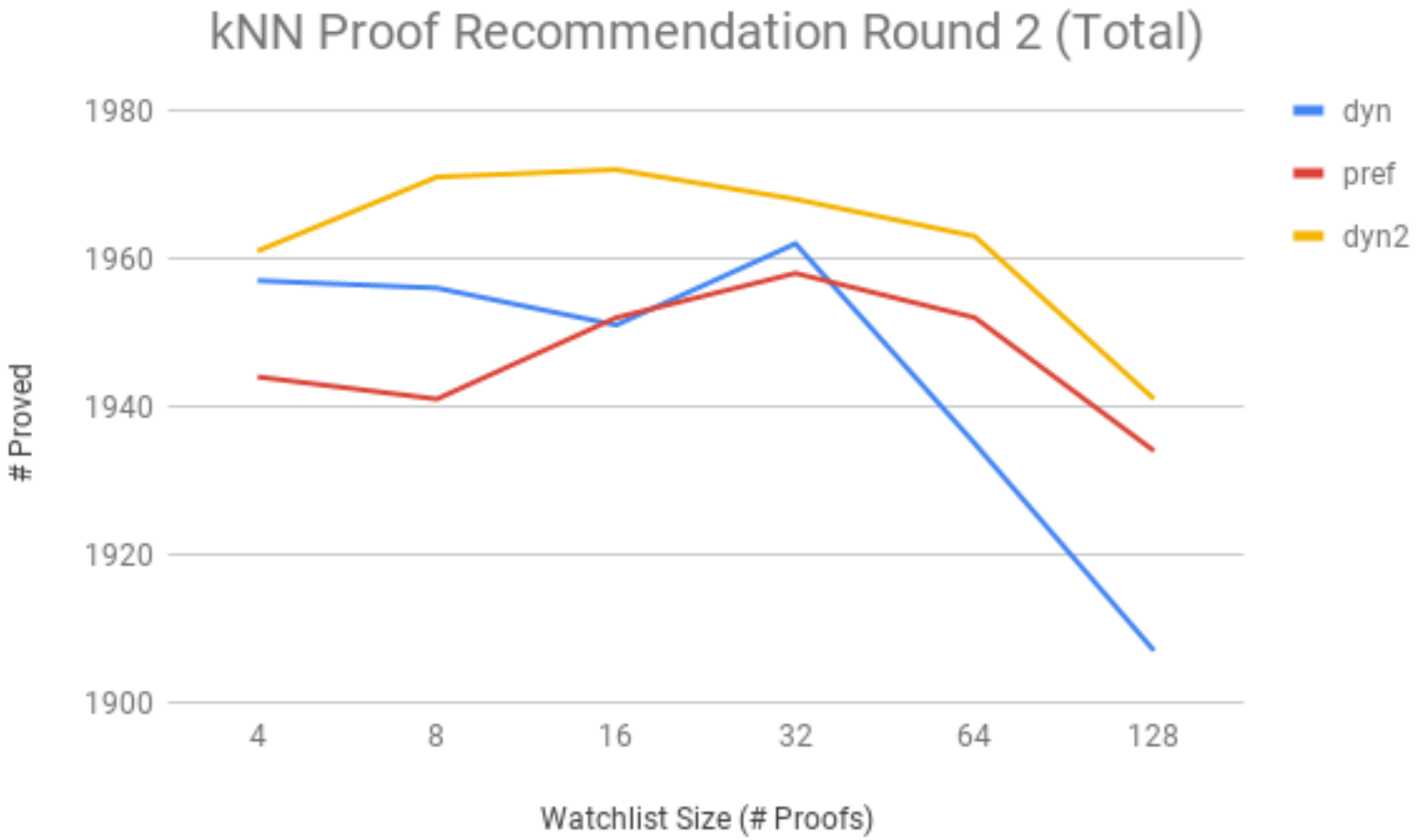
Watchlist Curation

In the ProofWatch paper we

1. Used 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

ENIGMAWatch

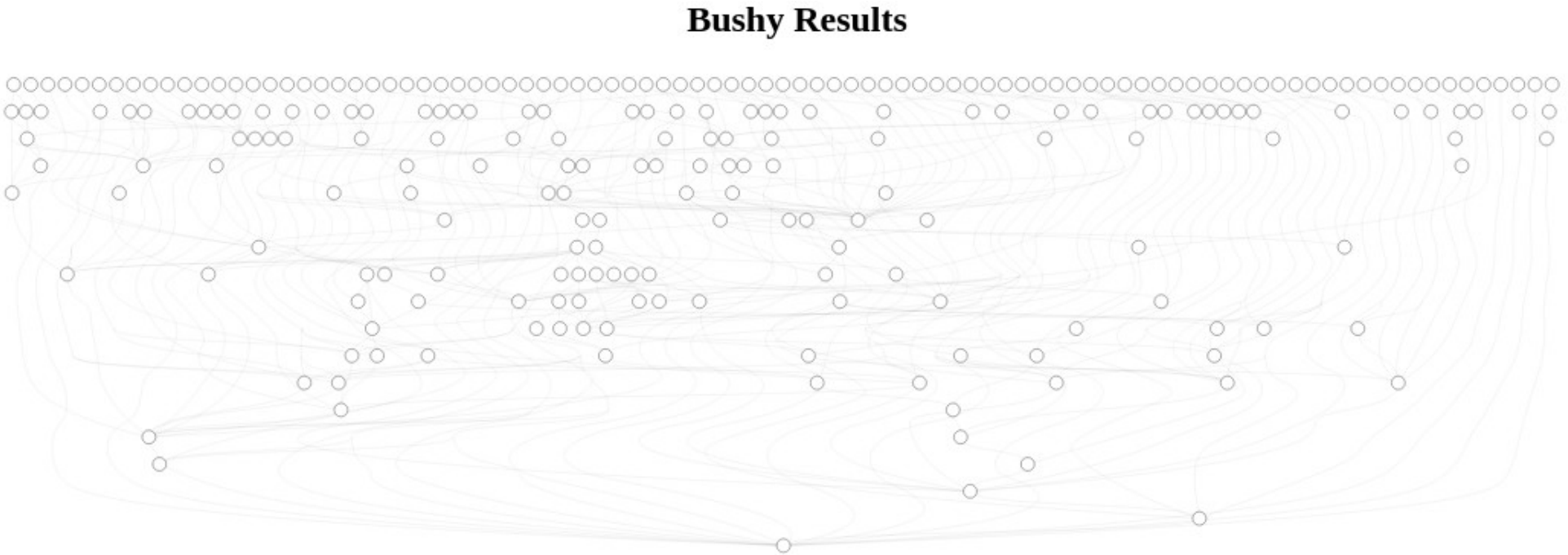
Idea: ProofWatch's proof-vector can capture some proof-state information. Give this to ENIGMA.

- 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.

Enter MPTP Challenge

- Small dataset of 252 problems leading up to the Bolzano-Weierstrass theorem.
- Problems range from easy to difficult.



Experiments (methods)

- **Baseline**: 10 strategies we previously evolved to perform well on Mizar problems.
- **ENIGMA**: A separate model is trained for each baseline strategy.
- **ProofWatch**: A static watchlist is made of all successful proofs from each baseline strategy.
- **ENIGMAWatch**: Baseline strategies are re-run to record proof-vectors at the point when given-clauses are selected. Then ENIGMA models are trained.

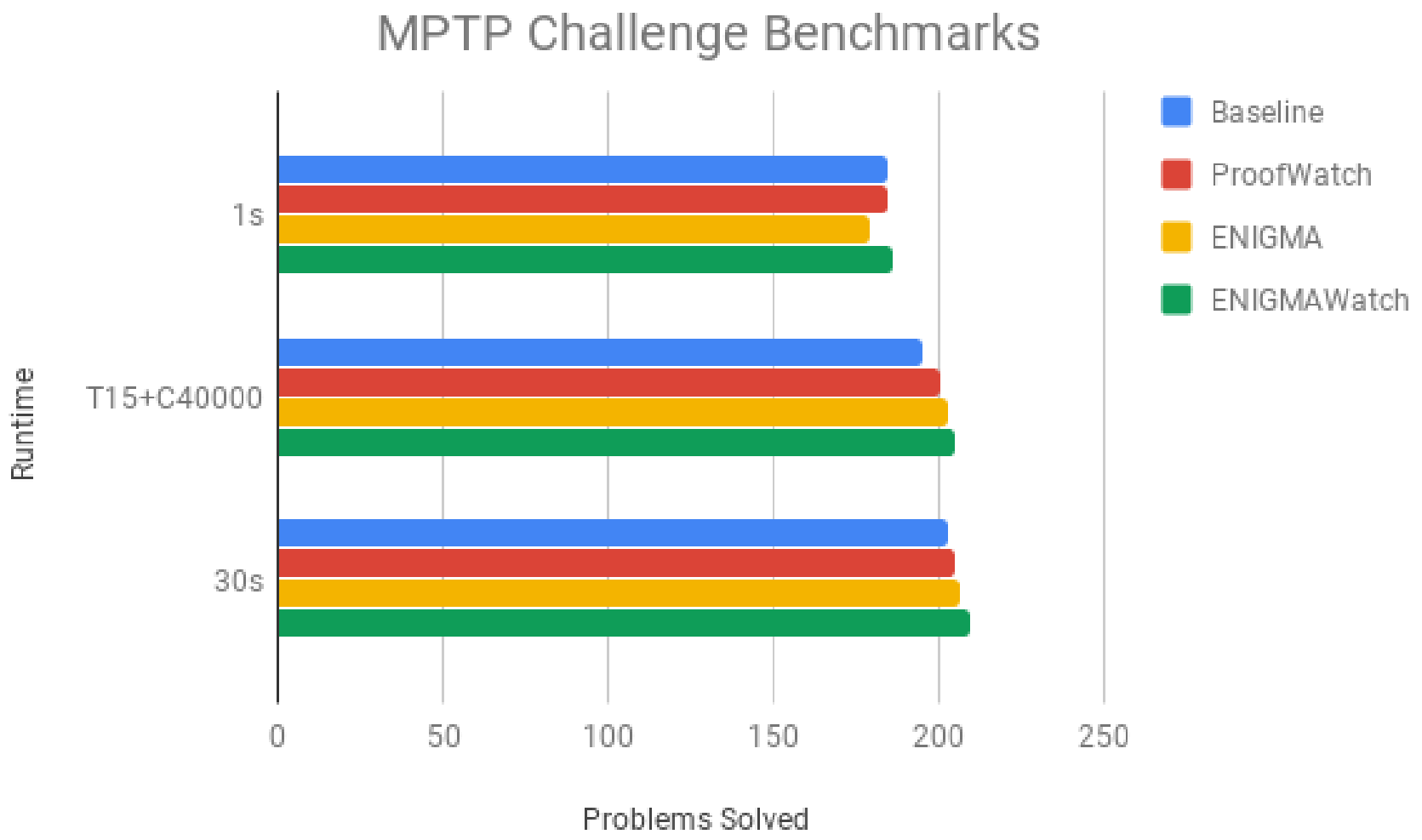
Experiments (time limits)

- **1s**
- **30s**

Abstract time:

- **T15+C40000**
  - E prover runs until 15 seconds passes OR
  - 40,000 given clauses are processed.

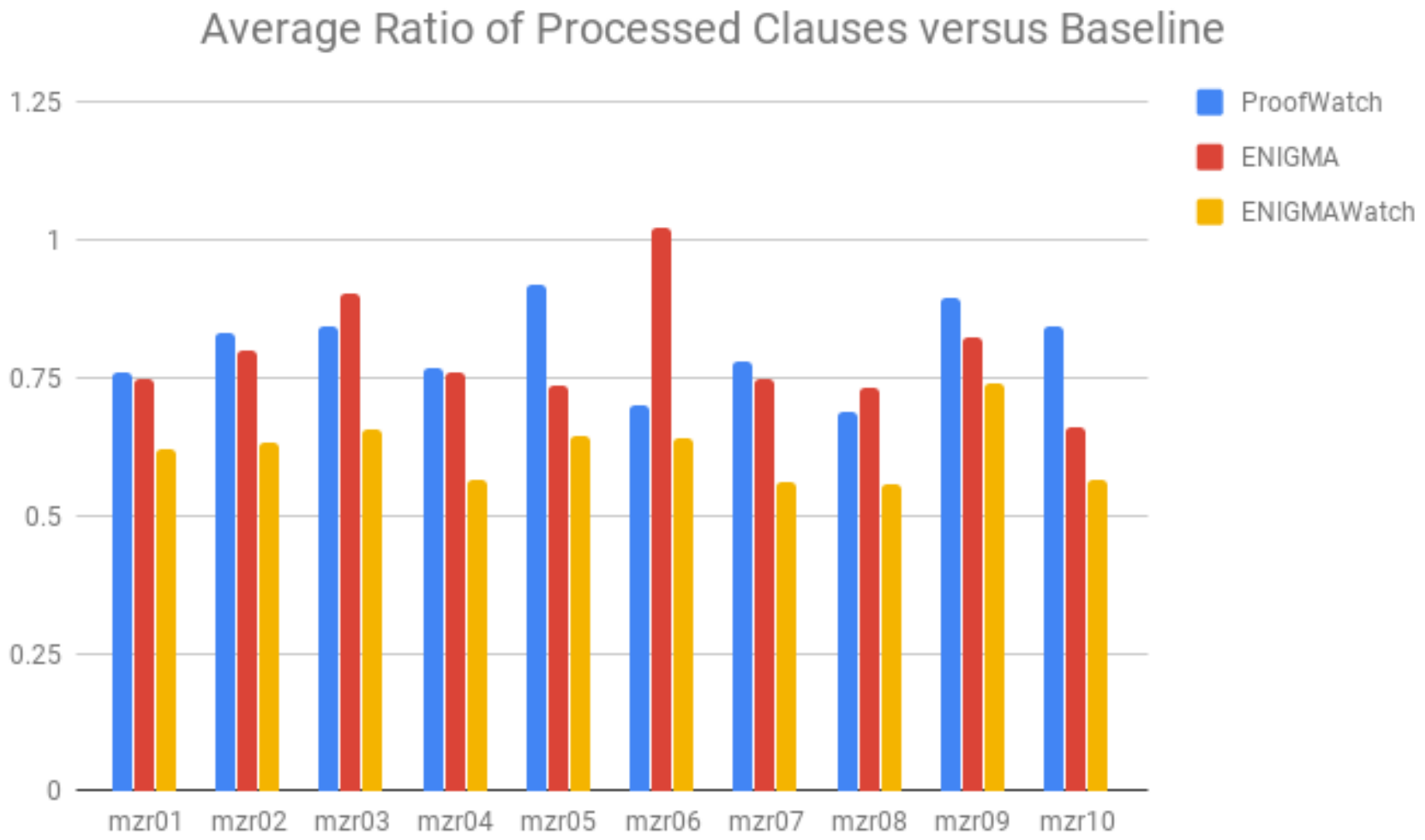
Results


MPTP Challenge Benchmarks
Runtime
1s
T15+C40000
30s
0
50
100
150
200
250
Problems Solved
Baseline
ProofWatch
ENIGMA
ENIGMAWatch

Results


Average Processed Clauses
Processed Clauses
1000
800
600
400
200
100
80
60
mzr01
mzr02
mzr03
mzr04
mzr05
mzr06
mzr07
mzr08
mzr09
mzr10
Baseline
ProofWatch
ENIGMA
ENIGMAWatch

Results


Average Ratio of Processed Clauses versus Baseline
1.25
1
0.75
0.5
0.25
0
mzr01
mzr02
mzr03
mzr04
mzr05
mzr06
mzr07
mzr08
mzr09
mzr10
ProofWatch
ENIGMA
ENIGMAWatch

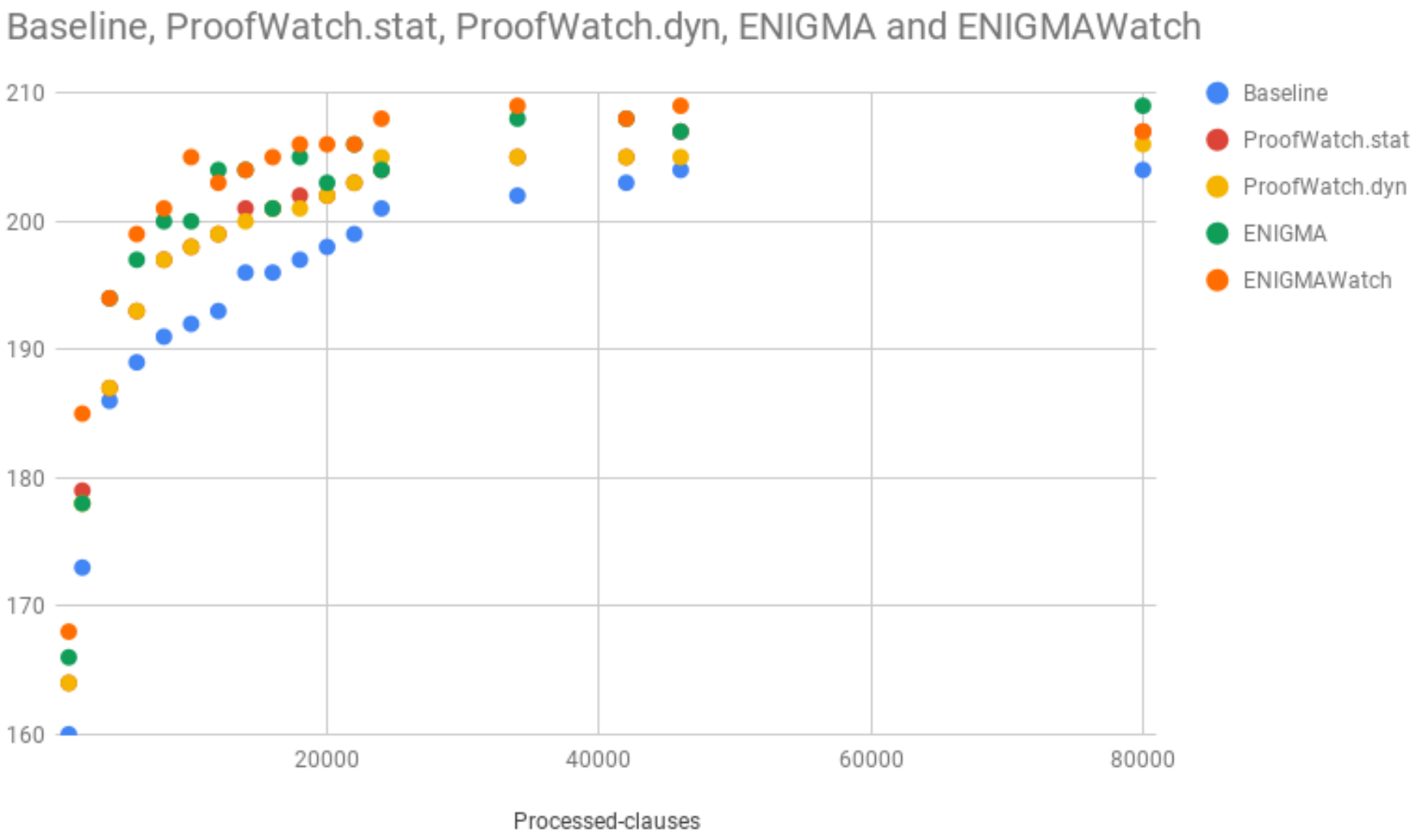
MPTP 2078

- This dataset includes the 33 full Mizar articles from the MPTP Challenge.
- Baseline ensemble prove 1461.
  - Too big for watchlist.
- k-mediods (of size 2-64) to form proof-vector watchlist didn’t work.
- TODO: k-NN recommendation a la ProofWatch with many empty-watchlists.
- TODO: Experiment with approximate matching.

Conclusion

- Initial MPTP Challenge benchmark is encouraging.
- ENIGMAWatch:
  - Enables E to prove more of the challenge problems
  - Enables E to prove the same problems more efficiently
- Good paradigm of merging symbolic and statistical machine learning.
- Needs more work to extend to larger datasets.

Epilogue


Baseline, ProofWatch.stat, ProofWatch.dyn, ENIGMA and ENIGMAWatch
210
200
190
180
170
160
20000
40000
60000
80000
Processed-clauses
Baseline
ProofWatch.stat
ProofWatch.dyn
ENIGMA
ENIGMAWatch

Epilogue

Enigma & ProofWatch * 2, ENIGMAWatch * 3