Part II: Enhancing ATPs with Machine Learning

Course Machine Learning and Reasoning 2020

MLR 2020[1]

[1]Czech Technical Univeristy in Prague (CIIRC)

April 3, 2020

1. Automated Strategy Invention
   - BliStr: Blind Strategy Maker
   - BliStrTune: Hierarchical Tuning
   - EmpireTune: Term Orderings Invention

2. ENIGMA: Efficient Inference Guidance Machine
   - Basic Enigmas
   - Enhancing Enigma
   - Enigma Anonymous

Outline

Using Automated Theorem Provers

- Solve problems in First-Order Logic
- Built-in automated strategy selection
- E Prover:

```
$ eprover --auto-schedule problem.tptp
```

- Vampire:

```
$ vampire --mode casc problem.tptp
```

No Success?

```
$ eprover --auto-schedule problem.tptp
...
# Failure: Resource limit exceeded (time)
# SZS status ResourceOut
eprover: CPU time limit exceeded, terminating
```

Try your own strategy!

```
$ eprover --auto-schedule problem.tptp
```

Try your own strategy!

```
$ eprover --definitional-cnf=24 --oriented-simul-paramod \
  --forward-context-sr --destructive-er-aggressive \
  --destructive-er --prefer-initial-clauses -tAuto \
  -Garity -F1 -WSelectMaxLComplexAvoidPosPred \
  -H(1*ConjectureRelativeTermWeight(PreferProcessed,1,1,1,...), \
     1*ConjectureTermPrefixWeight(SimulateSOS,1,3,0.5,10,...), \
     34*ConjectureRelativeTermWeight(DeferSOS,1,3,0.2,10,...)) \
  problem.tptp
```

Try your own strategy!

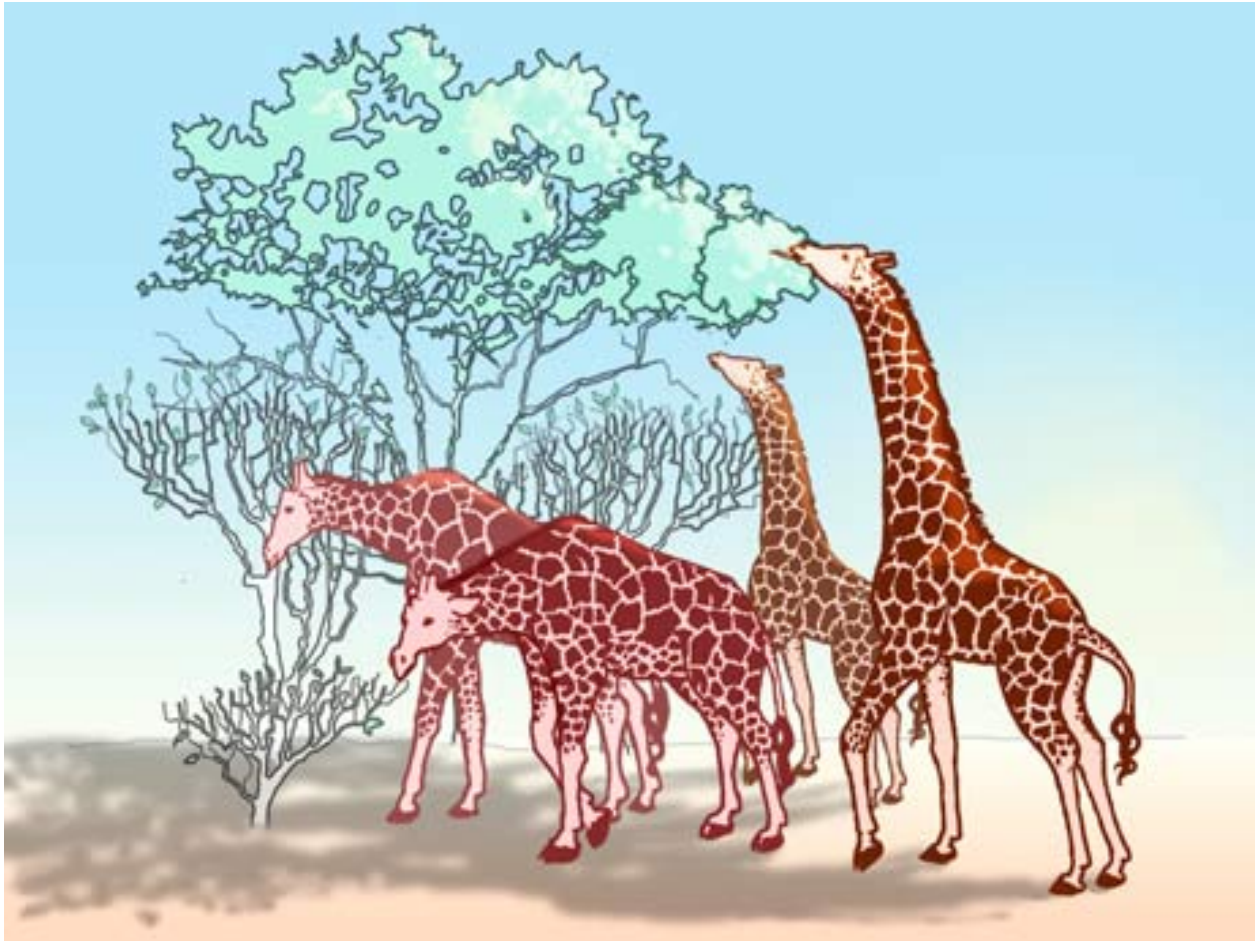
```
$ eprover --definitional-cnf=24 --oriented-simul-paramod \
  --forward-context-sr --destructive-er-aggressive \
  --destructive-er --prefer-initial-clauses -tAuto \
  -Garity -F1 -WSelectMaxLComplexAvoidPosPred \
  -H(1*ConjectureRelativeTermWeight(PreferProcessed,1,1,1,...), \
     1*ConjectureTermPrefixWeight(SimulateSOS,1,3,0.5,10,...), \
     34*ConjectureRelativeTermWeight(DeferSOS,1,3,0.2,10,...)) \
  problem.tptp

# Proof found!
# SZS status Theorem
```

Our Task

- Invent targeted strategies for E
- . . . specific for a given benchmark set
- . . . using machine learning methods (BliStrTune).
- . . . also for Vampire – EmpireTune

BliStr: Blind Strategy Maker



BliStr Basics

- giraffes = strategies
- food = problems
- the better a giraffe specializes . . .
- . . . the more it gets fed and evolves

Strategy Invention: BliStr Loop

Input: Initial strategies & benchmark problems
Output: Strategies which perform better on the benchmark

$All := Initials;$
loop
  $Evaluate(All, eval, min, max);$
  $G := Reduce(All, tops, bests);$
  $S := Select(G);$
  if S is undefined then
    return G;
  $S' := Improve(S, cutoff, imp);$
  $All := All \cup \{S'\};$
end

Step 1/4: Generation Evaluation

- evaluate all the strategies on all the problems
- compute overall result (solved/unsolved)
- measure length of the proof search
- for each strategy, compute best-performing problems
- discard too easy and too hard problems

Step 2/4: Generation Reduction

- consider only strategies performing best on *bests* problems
- . . . restrict the size of individuals
- keep only *tops* best strategies
- . . . restrict the count of individuals

Step 3/4: Strategy Selection

- select a strategy to improve
- . . . on its best performing problems
- never improve a strategy on the same problems
- prefer strategies with more best-performing problems
- prefer improving strategies on diverse problems

Step 4/4: Strategy Improvement

- improve a strategy on its best-performing problems
- using ParamILS software[1] [2]
- . . . parameter tuning and algorithm configuration
- different BliStr "clones" use ParamILS differently
  - BliStr: single ParamILS run
  - BliStrTune: several "hierarchical" ParamILS runs
  - EmpireTune: hierarchical runs for E, single run for Vampire

[1] `http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/`
[2] Frank Hutter, Holger Hoose, . . . (Uni. British Columbia)

BliStr: Basic Tuning

- One ParamILS run for one strategy improvement
- Given: initial strategy, set of problems, parameter space
- Task: find a better strategy w.r.t. the objective

$$penalty * |unsolved| + \sum_{P \in solved} processed(P)$$

- e.g. $23,001,234$ with $penalty = 1,000,000$

BliStrTune: Hierarchical Tuning

- BliStr explores a limited E protocol space
- Considers fixed set of clause weight functions
- Only 12 fixed functions:

```
ConjTermWeight(ConstPrio,0,1,0.1,18,400,50,300)
ConjSymbolWeight(PreferGround,0.2,50,100,5)
...
```

BliStrTune: Global Tuning Phase

- Explore different values for top-level parameters...

```
-tKBO6 -Garity -WSelectComplexG –oriented-simul-paramod -H'(
   3 * ConjTermWeight(ConstPrio,0,1,0.1,18,400,50,300),
   34 * ConjTermWeight(PreferUnits,1,1,0.1,100,9999,100,5),
   8 * ConjSymbolWeight(PreferGround,0.2,50,100,5)
)'
```

BliStrTune: Global Tuning Phase

- Explore different values for top-level parameters...

```
-tKBO6 -Garity -WSelectComplexG –oriented-simul-paramod -H'(
   3 * ConjTermWeight(ConstPrio,0,1,0.1,18,400,50,300),
   34 * ConjTermWeight(PreferUnits,1,1,0.1,100,9999,100,5),
   8 * ConjSymbolWeight(PreferGround,0.2,50,100,5)
)'
```

BliStrTune: Between Phases

- ...some values get improved.

```
-tLPO4 -Ginvarity -WSelectComplexG -H'(
   3 * ConjSymbolWeight(PreferGround,0.2,50,100,5)
   4 * ConjTermWeight(ConstPrio,0,1,0.1,18,400,50,300),
   23 * ConjTermWeight(PreferUnits,1,1,0.1,100,9999,100,5),
   16 * ConjPrefixWeight(PreferGoals,0.2,50,100,5)
)'
```

BliStrTune: Between Phases

- Next, improve weight function arguments, ...

```
-tLPO4 -Ginvarity -WSelectComplexG -H'(
   3 * ConjSymbolWeight(PreferGround,0.2,50,100,5)
   4 * ConjTermWeight(ConstPrio,0,1,0.1,18,400,50,300),
   23 * ConjTermWeight(PreferUnits,1,1,0.1,100,9999,100,5),
   16 * ConjPrefixWeight(PreferGoals,0.2,50,100,5)
)'
```

BliStrTune: Fine Tuning Phase

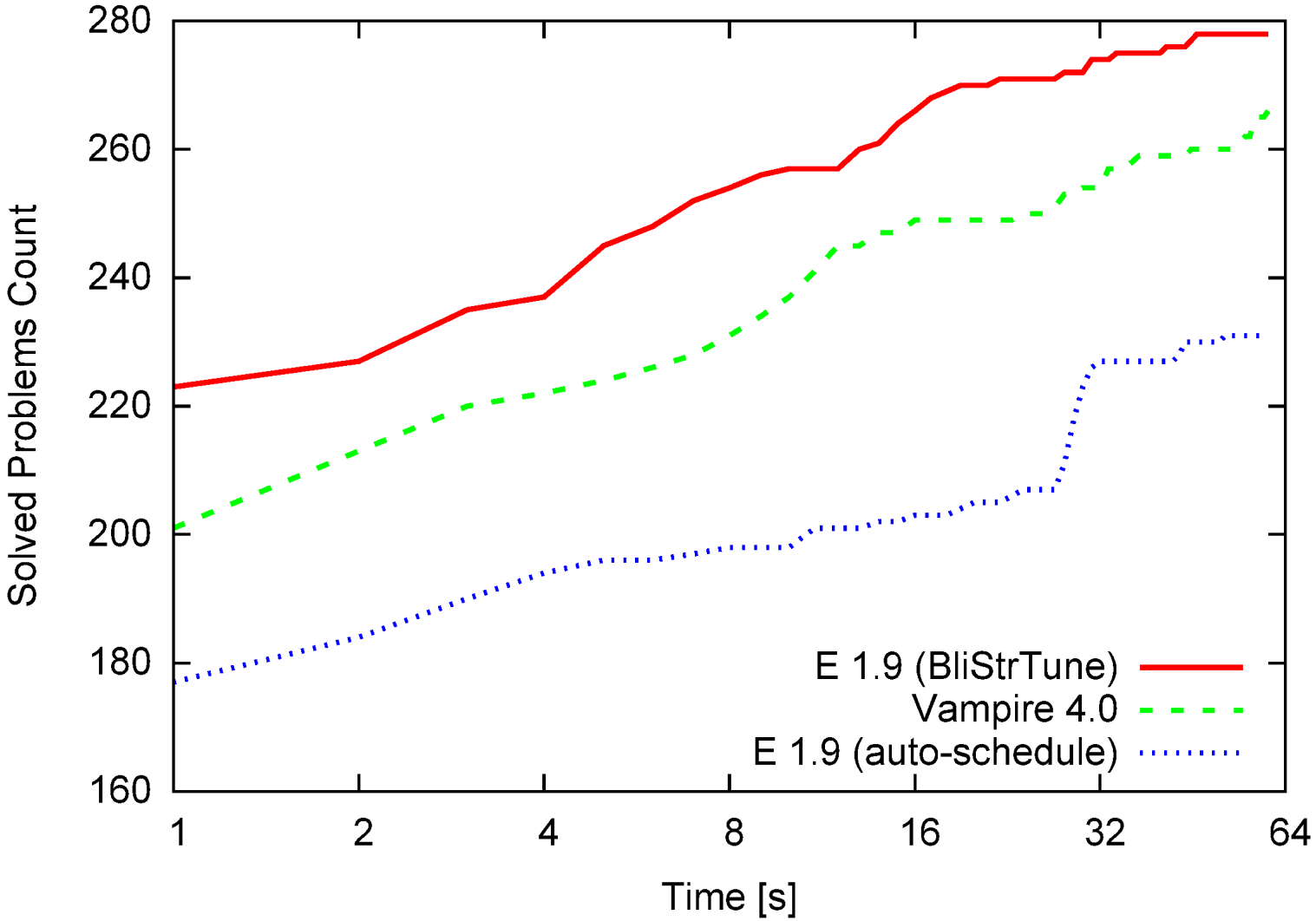
- . . . and values get changed again.

```
-tLPO4 -Ginvarity -WSelectComplexG -H'(
   3 * ConjSymbolWeight(ConstPrio,0.4,10,10,50)
   4 * ConjTermWeight(PreferGround,0,1,1.5,9,100,50,300),
   23 * ConjTermWeight(PreferGoals,1,1,-0.1,200,9,100,5),
   16 * ConjPrefixWeight(PreferUnits ,0.2,10,100,5)
)'
```

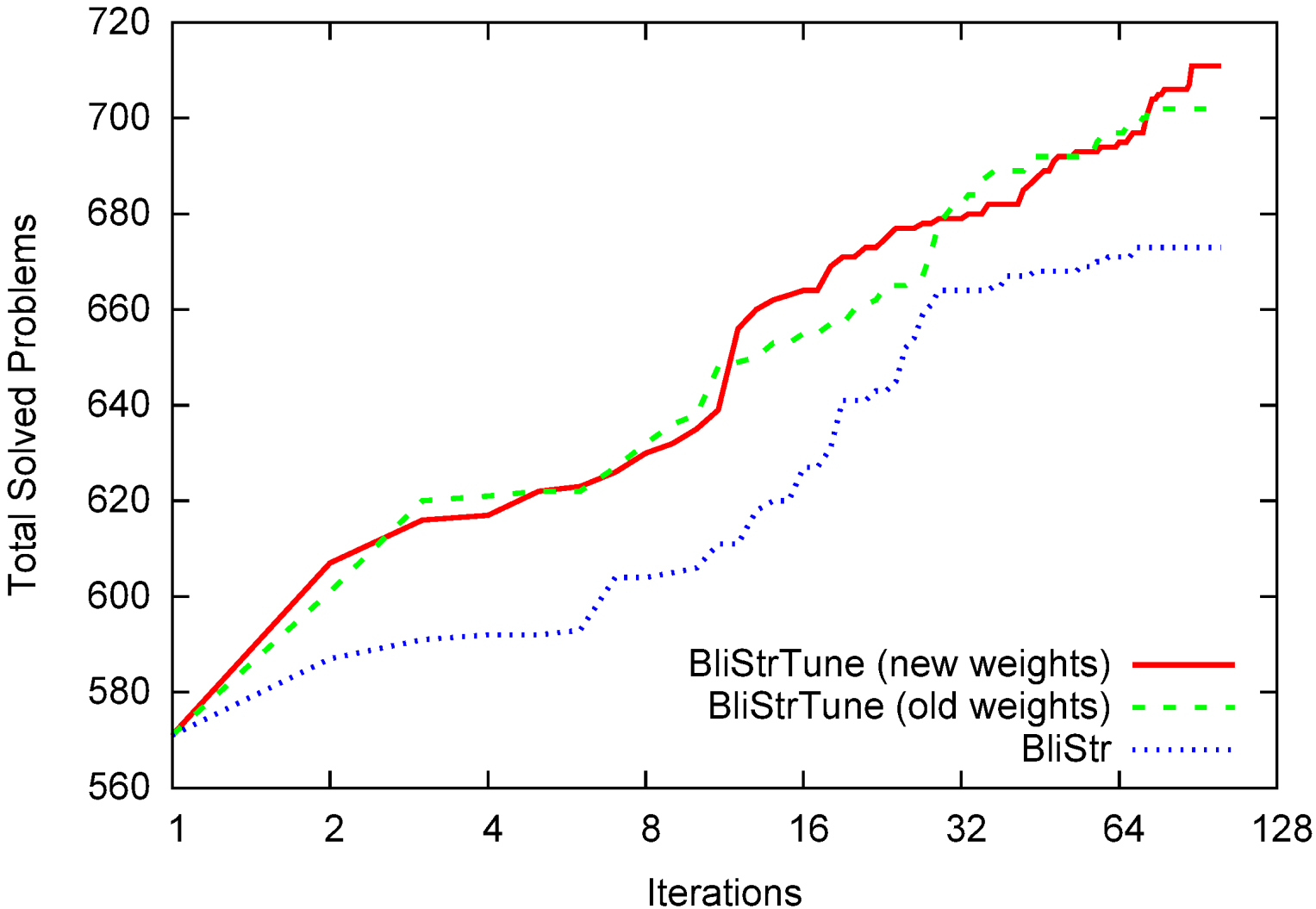
BliStrTune: Experiments

- Mizar @ Turing division of competition CASC'12
- problems exported from Mizar
- 1000 training problems known beforehand
- 400 testing problems in the competition

BlistrTune: Progress on Testing Problems



BlistrTune: Impact of Hierarchical Tuning



EmpireTune: E and Vampire Tuning

- select Vampire options for strategy selection
- ...(Sine, saturation alg., AVATAR, inference rules)
- describe their possible values
- rule out incompatible combinations

```
sa {discount,inst_gen,lrs,otter} [otter]
erd {off,input_only} [input_only]
fde {all,none,unused} [unused]
gsp {input_only,off} [off]
ins {0,1,2,4,8} [0]
...
```

Term Orderings in ATP

- partial ordering on terms (from a symbol precedence)
- used to restrict and guide a proof search
- . . . ordered resolution, orient rewriting rules
- for n symbols, $n!$ precedences
- the right ordering can have a dramatical effect
- however, not clear which one is the right one

Term Orderings in Vampire 4.2

Standard Vampire:

- **occurrence** - order symbols by their occurrence in the problem
- **arity** - order symbols by their arity
- **frequency** - order symbols by frequency in the problem

Term Orderings in Vampire 4.2

EmpireTune extension:

- user specifies coefficients: $spoc, spac, spfc$
- $val(s) = spoc * occ(s) + spac * arity(s) + spfc * freq(s)$
- symbols are order by $val(s)$
- additionally: explicitly specified precedence

Tuning Ordering in EmpireTune

- use ParamILS to find the best possible ordering
- . . . best values for *spoc*, *spac*, *spfc*
- . . . or best explicit precedence
- hierarchical approach:
  1. tune everything except ordering
  2. tune ordering only

Example: Tuning Ordering in EmpireTune

Take the input Vampire strategy:

-av off -awr 2:3 -bd preordered -drc off -fd preordered -fde unused
-fsr off -nm 64 -s -1004 -sa otter -sas z3 -updr off -urr on
-spoc 1 -spac 2 -spfc 1

Example: Tuning Ordering in EmpireTune

Phase 1: Allow only non-ordering options to be changed:

-av off -awr 2:3 -bd preordered -drc off -fd preordered -fde unused
-fsr off -nm 64 -s -1004 -sa otter -sas z3 -updr off -urr on
-spoc 1 -spac 2 -spfc 1

Example: Tuning Ordering in EmpireTune

Phase 1 output: New Vampire strategy:

-av off -awr 5:4 -bce on -bd preordered -drc off -fd preordered -fde unused -fsr off -nm 26 -s 1004 -sa otter -sas z3 -updr off -urr on -spoc 1 -spac 2 -spfc 1

Example: Tuning Ordering in EmpireTune

Phase 2: Allow only ordering options to be changed:

-av off -awr 5:4 -bce on -bd preordered -drc off -fd preordered -fde unused -fsr off -nm 26 -s 1004 -sa otter -sas z3 -updr off -urr on -spoc 1 -spac 2 -spfc 1

Example: Tuning Ordering in EmpireTune

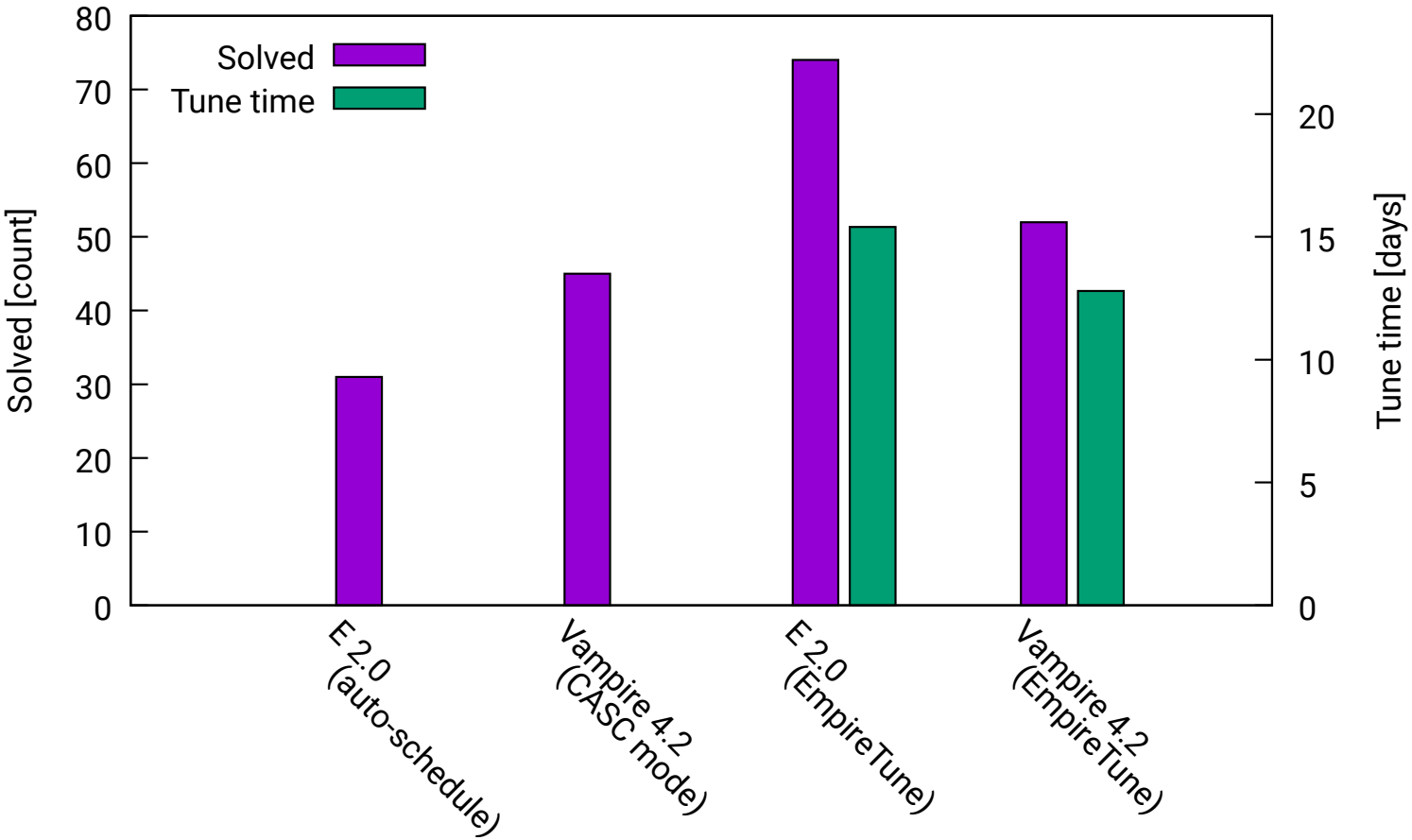
Phase 2 output: New Vampire strategy:

-av off -awr 5:4 -bce on -bd preordered -drc off -fd preordered -fde unused -fsr off -nm 26 -s 1004 -sa otter -sas z3 -updr off -urr on -spoc 0 -spuc 1 -fp identity:0:1,associator:3:2,multiply:2:3

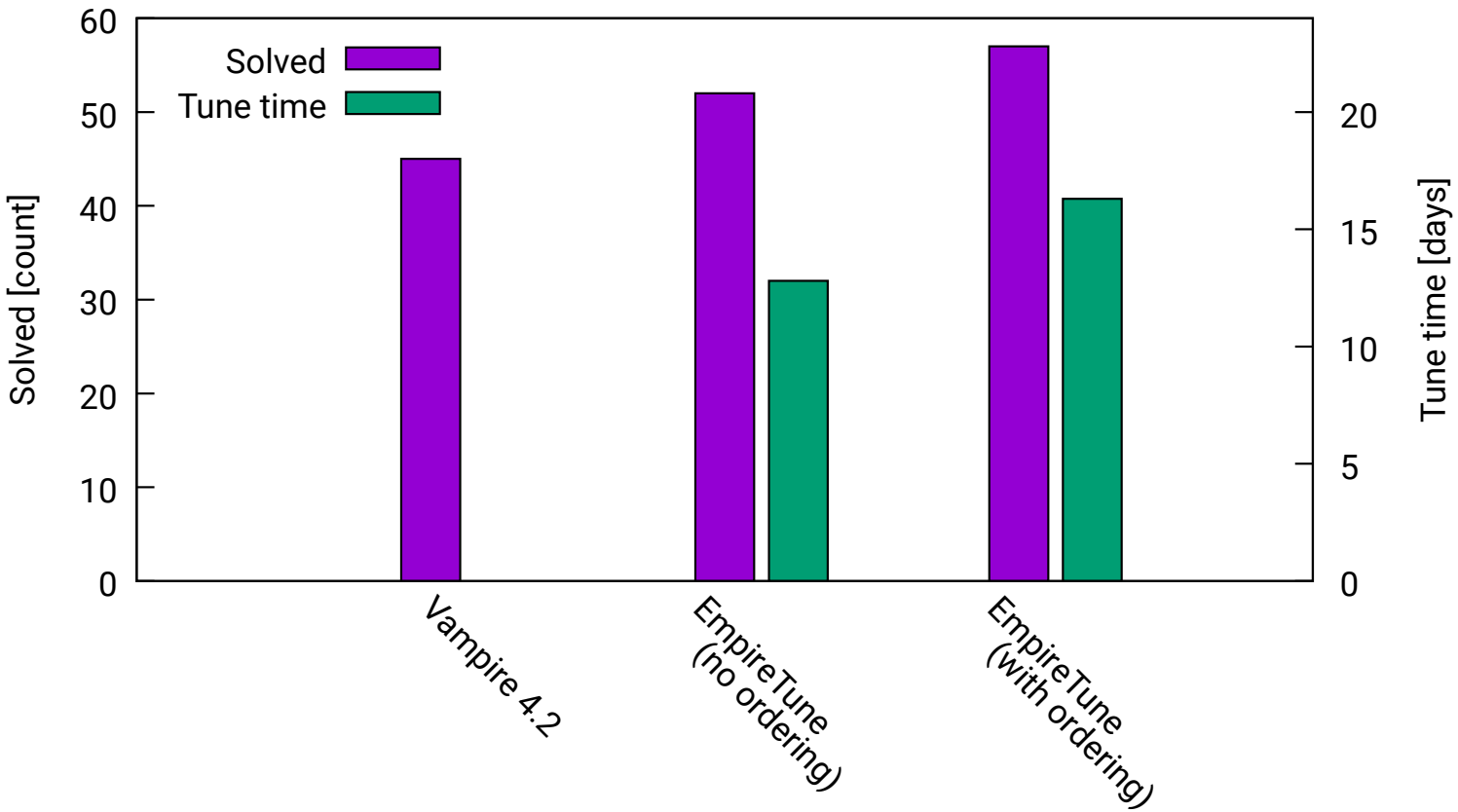
Experiments Setting

- AIM problems from CASC (LTB category)
- . . . 1020 training problems, 200 testing problems
- advantages (simplifications):
- . . . different conjectures in the same theory
- . . . small number of symbols (8+4)
- . . . symbols are used consistently

EmpireTune: Impact of Tuning



EmpireTune: Impact of Ordering Tuning



Outline

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for...

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for...

Efficient learNing-based Inference Guiding MAchine

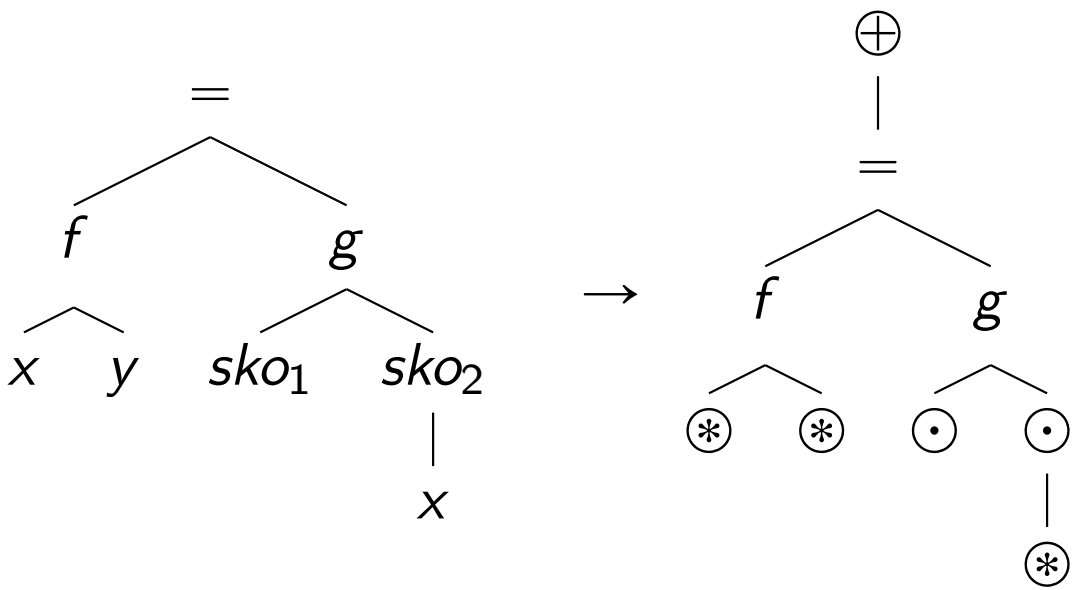
LIBLINEAR: Linear Classifier

- LIBLINEAR: open source library[3]
- **input:** positive and negative examples (float vectors)
- **output:** model ($\sim$ a vector of weights)
- **evaluation** of a generic vector: dot product with the model

[3] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

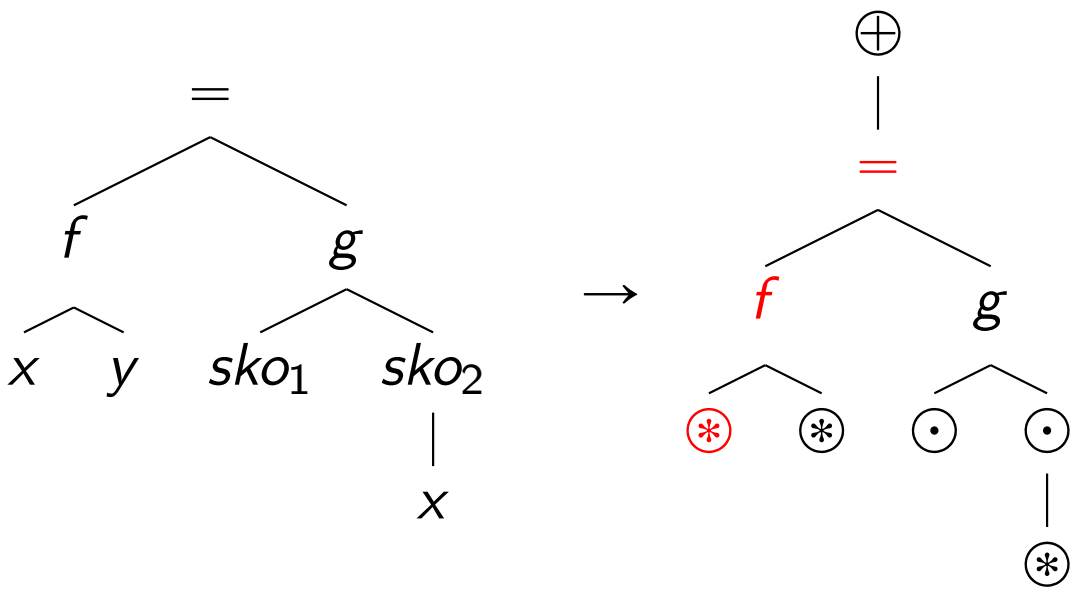
Clauses as Feature Vectors

Consider the literal as a tree and simplify (sign, vars, skolems).



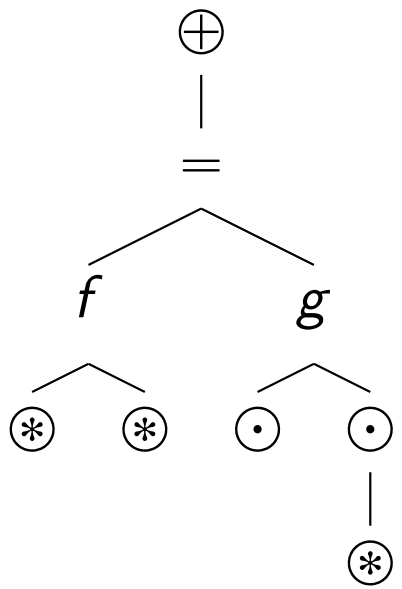
Clauses as Feature Vectors

Features are descending paths of length 3 (triples of symbols).



Clauses as Feature Vectors

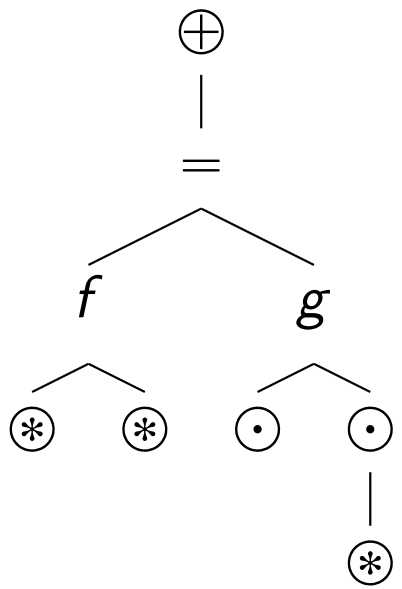
Collect and enumerate all the features. Count the clause features.



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

Clauses as Feature Vectors

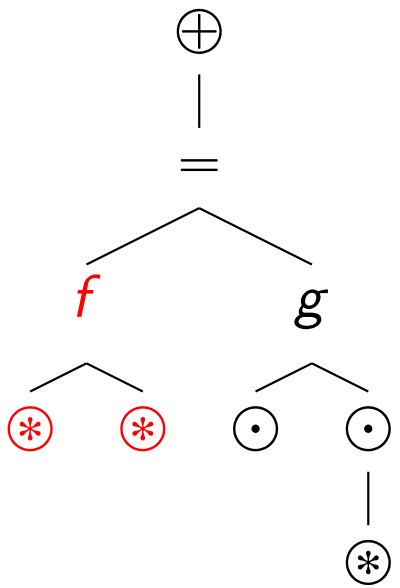
Take the counts as a feature vector.



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

Horizontal Features

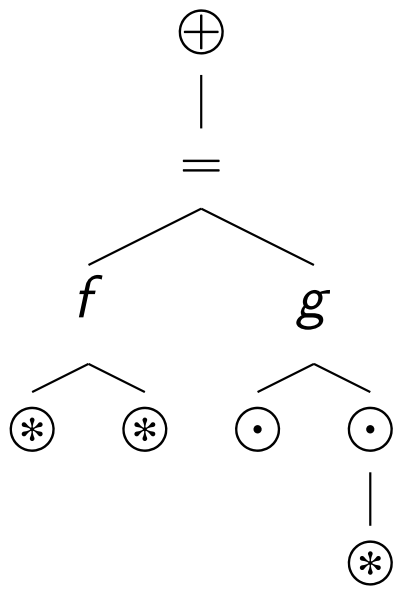
Function applications and arguments top-level symbols.



#	feature	count
1	$(\oplus,=,a)$	0
⋮	⋮	⋮
100	$=(f,g)$	1
101	$f(\circledast,\circledast)$	1
102	$g(\odot,\odot)$	1
103	$\odot(\circledast)$	1
⋮	⋮	⋮

Static Clause Features

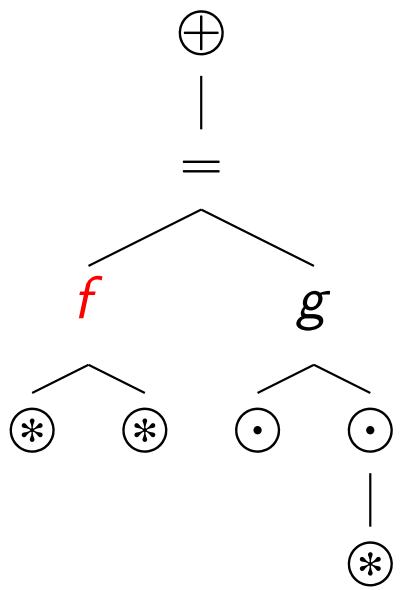
For a clause, its length and the number of pos./neg. literals.



#	feature	count/val
103	⊙(⊛)	1
⋮	⋮	⋮
200	len	9
201	pos	1
202	neg	0
⋮	⋮	⋮

Static Symbol Features

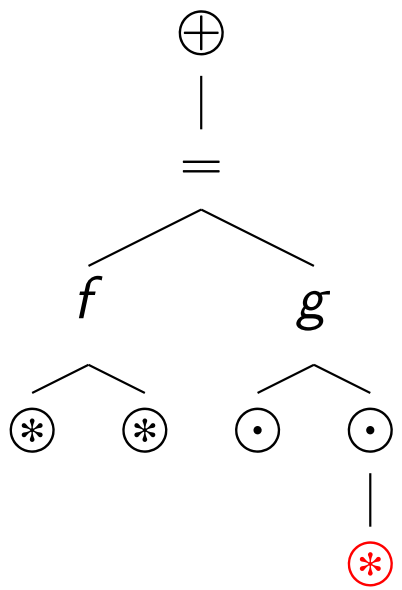
For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Static Symbol Features

For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Enigma Model Construction

1. Collect training examples from E runs (useful/useless clauses).
2. Enumerate all the features (π :: feature $\rightarrow$ int).
3. Translate clauses to feature vectors.
4. Train a LIBLINEAR classifier (w :: float$^{|\mathsf{dom}(\pi)|}$).
5. Enigma model is $\mathcal{M} = (\pi, w)$.

Conjecture Features

- Enigma classifier $\mathcal{M}$ is independent on the goal conjecture!
- Improvement: Extend Φ_C with goal conjecture features.
- Instead of vector Φ_C take vector (Φ_C, Φ_G).

Given Clause Selection by Enigma

We have Enigma model $\mathcal{M} = (\pi, w)$ and a generated clause C.

1. Translate C to feature vector Φ_C using π.
2. Compute prediction:

$$\text{weight}(C) = \begin{cases} 1 & \text{iff } w \cdot \Phi_C > 0 \\ 10 & \text{otherwise} \end{cases}$$

Enigma Given Clause Selection

- We have implemented `Enigma` weight function in E.
- Given E strategy $\mathcal{S}$ and model $\mathcal{M}$.
- Construct new E strategy:
- $\mathcal{S} \odot \mathcal{M}$: Use $\mathcal{M}$ as the only weight function:

  (1 * `Enigma(`$\mathcal{M}$`)`)
- $\mathcal{S} \oplus \mathcal{M}$: Insert $\mathcal{M}$ to the weight functions from $\mathcal{S}$:

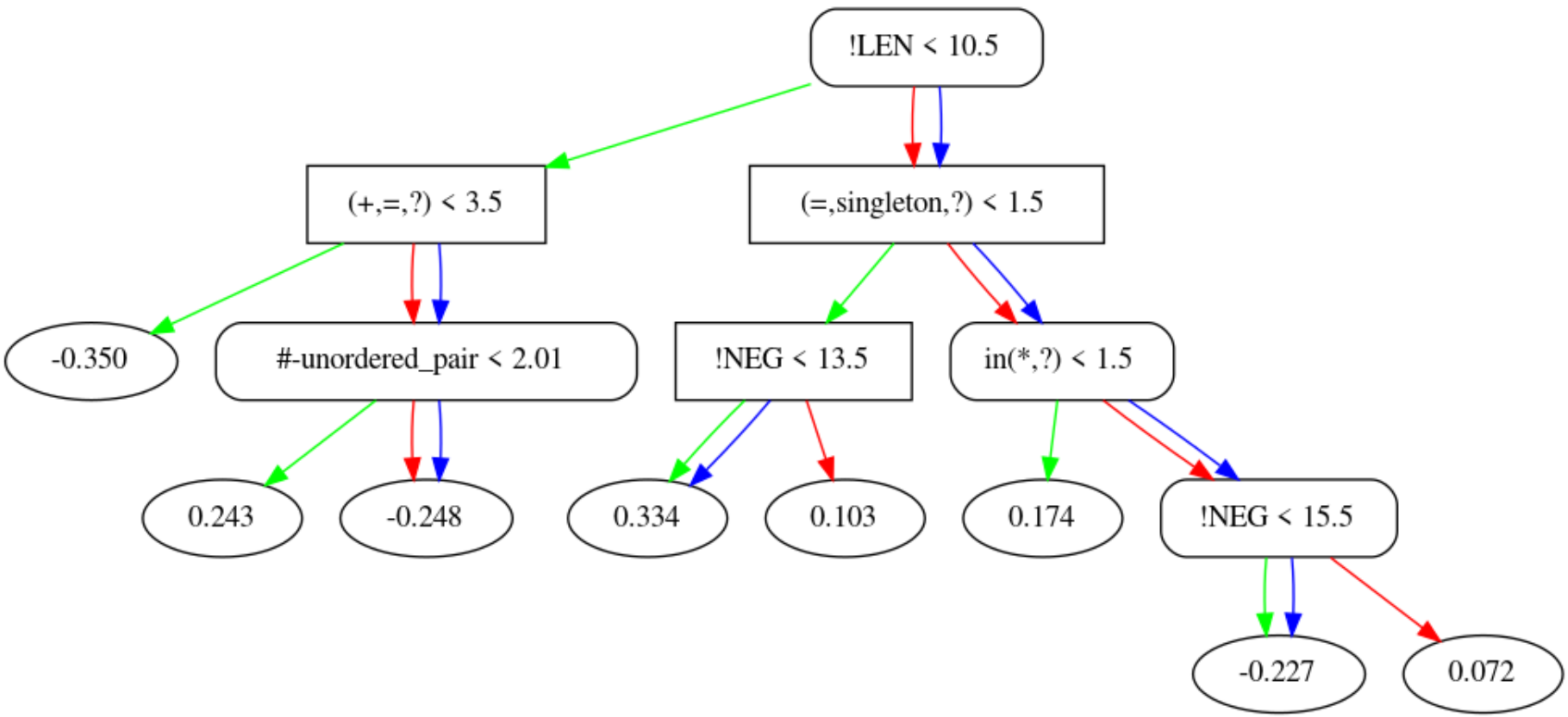
  (23 * `Enigma(`$\mathcal{M}$`)`,
  3 * `StandardWeight(...)`,
  20 * `StephanWeight(...)`)

Decision Tree Models

- **Idea:** Use decision trees instead of linear classifier.
- Gradient boosting library XGBoost/LightGBM
- Provides C/C++ API and Python (and others) interface.
- Uses exactly the same training data as LIBLINEAR.
- We use the same Enigma features.
- No need for training data balancing.

XGBoost/LightGBM Models

- An model consists of a set of decision trees.
- Leaf scores are summed and translated into a probability.



Feature Hashing

- With lot of training samples we have lot of features.
- LIBLINEAR/XGBoost can't handle too long vectors ($> 10^5$).
- Why? Input too big... Training takes too long...
- Solution: Reduce vector dimension with feature hashing.
- Encode features by strings and ...
- ... use a general purpose string hashing function.
- Values are summed in the case of a collision.

Experiments: Hammering Mizar

- MPTP: FOL translation of selected articles from Mizar Mathematical Library (MML).
- Contains $57,880$ problems.
- Small versions with (human) premise selection applied.
- Single good-performing E strategy $\mathcal{S}$ fixed.
- All strategies evaluated with time limit of 10 seconds.

Solved problems: one looping iteration

- Decision trees depth $= 9$
- $\mathcal{M}^0$ is trained on problems solved by $\mathcal{S}$
- $\mathcal{M}^n$ $(n > 0)$ is trained on problems solved by $\mathcal{S}$ and $\mathcal{S} \odot \mathcal{M}^i$ (for all $i < n$) and $\mathcal{S} \oplus \mathcal{M}^i$ (for all $i < n$)

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}^0$	$\mathcal{S} \oplus \mathcal{M}^0$	$\mathcal{S} \odot \mathcal{M}^1$	$\mathcal{S} \oplus \mathcal{M}^1$
solved	14933	16574	20366	21564	22839
$\mathcal{S}\%$	+0%	+10.5%	+35.8%	+43.8%	+52.3%
$\mathcal{S}+$	+0	+4364	+6215	+7774	+8414
$\mathcal{S}-$	-0	-2723	-782	-1143	-508

Solved problems: more loops

	$\mathcal{S}$	$\mathcal{S} \oplus \mathcal{M}^0$	$\mathcal{S} \oplus \mathcal{M}^1$	$\mathcal{S} \oplus \mathcal{M}^2$	$\mathcal{S} \oplus \mathcal{M}^3$
solved	14933	20366	22839	23467	23753
$\mathcal{S}\%$	+0%	+35.8%	+52.3%	+56.5%	+58.4
$\mathcal{S}+$	+0	+6215	+8414	+8964	+9274
$\mathcal{S}-$	-0	-782	-508	-430	-454

Solved problems: deeper trees

- Increase tree depth to 12 and 16
- Train the model on the same data as $\mathcal{M}^3$

	$\mathcal{S} \odot \mathcal{M}^3_{12}$	$\mathcal{S} \oplus \mathcal{M}^3_{12}$	$\mathcal{S} \odot \mathcal{M}^3_{16}$	$\mathcal{S} \oplus \mathcal{M}^3_{16}$
solved	24159	24701	25100	25397
$\mathcal{S}\%$	+61.1%	+64.8%	+68.0%	+70.0%
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-535	-295	-309	-183

Training Statistics: different tree depths

- 1.8 M features (hashed to 2^{15})
- vector dimension is 2^{16}
- input trains file is 38 GB
- . . . and contains 63 M training samples (4.2M pos x 59M neg)
- . . . with 5000 M non-zero values (density 0.1%)

depth	error	real time	CPU time	size (MB)	speed
9	0.201	2h41m	4d20h	5.0	5665.6
12	0.161	4h12m	8d10h	17.4	4676.9
16	0.123	6h28m	11d18h	54.7	3936.4

Symbol Anonymization

Replace symbol names in features by their arities.

- identity → f0
- plus(X,Y) → f2(X,Y)
- multiply(X,Y) → f2(X,Y)
- less(X,Y) → p2(X,Y)

Additional Symbol Independent Features

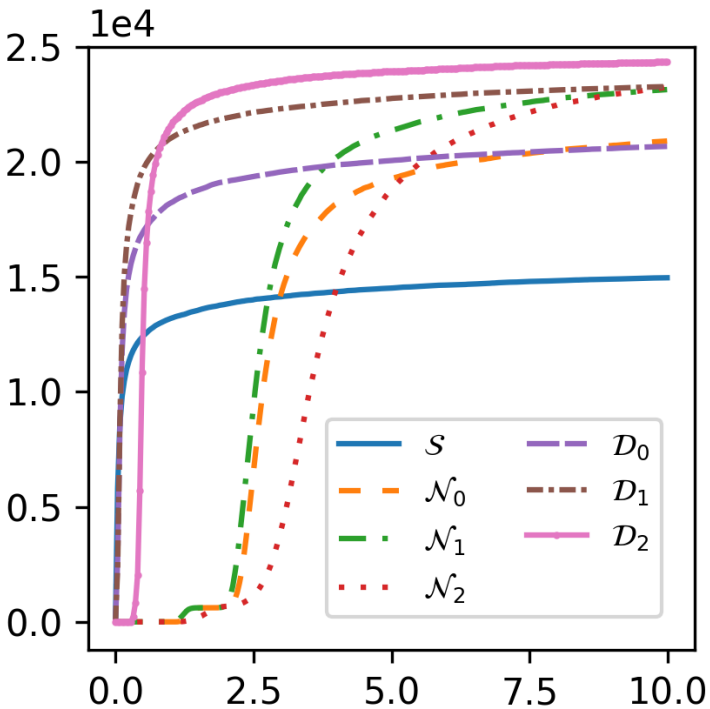
Additional variable/symbols statistics.

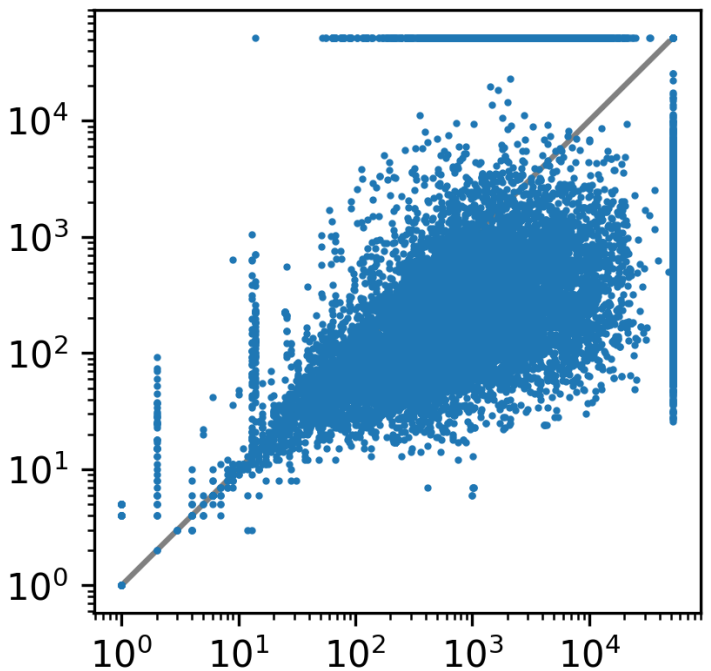
- the number of variables/symbols in a clause
- the number of variables/with with one/more occurrences
- the number of occurrences of the most occurring variable
- ...

Hammering Mizar Anonymously

$\mathcal{M}$	TPR [%]	TNR [%]	training size	training time	training params	real time $\mathcal{S} \oplus \mathcal{M}$	real time +%
$\varnothing$	-	-	-	-	-	14 966	0.0
$\mathcal{D}_0$	84.9	68.4	14M	2h29m	X,d12	20 679	38.1
$\mathcal{D}_1$	79.0	79.5	29M	4h33m	X,d12	23 679	58.2
$\mathcal{D}_2$	80.5	79.2	47M	40m	L,d30,l1800	24 347	62.7

Solved in Time & Processed Clauses





Conclusion

Thank You! Good Bye!