First Experiments with Watchlist Guidance on Mizar

Zarathustra Goertzel Jan Jakubův Josef Urban
Stephan Schulz

Czech Technical University in Prague

DHBW Stuttgart

AITP'18, 29th March 2018

Watchlist Guidance

The watchlist contains clauses that guide E prover's search, Called *hints* in Prover9 by Bob Veroff

Can be thought of as containng:

- Mathematical Tricks (not already in E)
- Proof sketches/schema
- Conjectured lemmas/sub-goals

From other E (ATP) proofs. Thus **learning**!

Watchlist Guidance

The watchlist contains clauses that guide E prover's search, Called *hints* in Prover9 by Bob Veroff

Can be thought of as containng:

- Mathematical Tricks (not already in E)
- Proof sketches/schema
- Conjectured lemmas/sub-goals

From other E (ATP) proofs. Thus learning.

Current result: 7.2% more proofs in Mizar Mathematical Library!

Watchlist Guidance

The watchlist contains clauses that guide E prover's search, Called *hints* in Prover9 by Bob Veroff

Can be thought of as containng:

- Mathematical Tricks (not already in E)
- Proof sketches/schema
- Conjectured lemmas/sub-goals

From other E (ATP) proofs. Thus learning.

Current result: 7.2% more proofs!

- And some longer ones proved only by watchlist guidance.

First Experiment

For AITP18 we experimented with

- how to include the watchlist in E strategies (via PreferWatchlist).
- how to build the watchlist from prior proofs

Further Experiments

After AITP18 submission we continued experimenting with

- *matching* **hints** modulo skolem names (e.g., eskfoobar(A,B) $\sqsubseteq$ esk(A,B))
- Dynamic weighting of **hints** by proof completion
- K-NN recommendation of **hints** from prior proofs

Where's the AI in this?

- Watchlist provides logical interface between AI/human and ATP:
  - High and low level systems
- Watchlist with dynamic weighting can provide a vectorial proofstate characterization for AI methods (e.g. ENIGMA's SVM)
- Learning from prior proofs

E Basics

- E's a saturation-based refutational theorem prover (using superposition calculus)
- To prove P, E does inference from $\neg P$ until contradiction or all inferences have been done.
- Repeats *given-clause loop*:

```
while (no proof found)
{
  select a given clauses
  apply inference rules to selected clauses
  process inferred clauses
}
```

Motivation

```
while (no proof found)
{
  select a given clauses
  apply inference rules to selected clauses
  process inferred clauses
}
```

- **select a given clauses**

To prove hard conjectures, clauses must be selected intelligently and specifically for the given conjecture.

Basic Watchlist Mechanism

- Check if each clause subsumes a **hint**
  (That is, clause $\sqsubseteq$ hint,
  and **hint** led to the empty set before!)
- If yes, boost clause priority (and thus score), making selection more likely.
  And remove **hint** from watchlist (optional).

Used via the *PreferWatchlist* priority function

Priority? Weight? Score?

E chooses given clauses by weighted round-robbin selection, with priority queues as specified in a strategy containing **Clause Evaluation Functions**.

An example E strategy:

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

Where

- **Priority** functions: PreferWatchlist, PreferProcessed
- **Weight** functions: Clauseweight, FIFOWeight

baseline 02: an actual evolved strategy file

```
--definitional-cnf=24 --split-aggressive --simul-par
amod --forward-context-sr --destructive-er-aggressive
--destructive-er --prefer-initial-clauses -tKBO -winv
freqrank -c1 -Ginvfreq -F1 --delete-bad-limit=150000000
-WSelectMaxLComplexAvoidPosPred
-H(1*ConjectureTermPrefixWeight(DeferSOS,1,3,0.1,5,0,0.
,1*ConjectureTermPrefixWeight(DeferSOS,1,3,0.5,100,0,0.
,1*Refinedweight(PreferWatchlist,4,300,4,4,0.7)
,1*RelevanceLevelWeight2(PreferProcessed,0,1,2,1,1,1,20
,1*StaggeredWeight(DeferSOS,1),1*SymbolTypeweight(Defer
,2*Clauseweight(PreferWatchlist,20,9999,4)
,2*ConjectureSymbolWeight(DeferSOS,9999,20,50,-1,50,3,3
,2*StaggeredWeight(Def
```

Dynamic Watchlist

- Multiple watchlists: one for each loaded proof.
- Keep track of which prior proofs (files) **hints** come from.
- Count what % of each proof has been covered (subsumed)
- Assign more priority via **PreferWatchlistRelevant** for higher completion %.
  - –> closer to contradiction
  - –> closer to current proof
- Can represent proof state.

Dynamic Watchlist with Decay

Watchlist subsumption is sparse, so

- *Inherit watchlist priority from parents* with multiplicative decay.

Watchlist Relevance Calculation

For a clause C matching at least one watchlist W_i:

$$relevance_0(C) = \max_{W \in \{W_i : C \sqsubseteq W_i\}} \left(\frac{progress(W)}{|W|} \right)$$

Where $progress(W)$ is how many clauses in watchlist W have been subsumed.

Watchlist Relevance Calculation

For a clause C matching at least one watchlist W_i:

$$relevance_0(C) = \max_{W \in \{W_i : C \sqsubseteq W_i\}} \left(\frac{progress(W)}{|W|} \right)$$

With decay, for $\delta < 1$,

$$relevance_1(C) = relevance_0(C) + \delta * \operatorname{avg}_{D \in parents(C)} \Big(relevance_1(D) \Big)$$

Aditionally, reset to 0 if $relevance_1(C) < \alpha$ and $\frac{relevance_1(C)}{length(C)} < \beta$ for *threshold* parameters $\alpha, \beta > 0$.

Baseline Strategies

- We use 57897 Mizar40 conjectures with premises pre-selected. We previously evolved 32 strategies with a coverage of 24702 proofs (in 5s each).
- In practice, we use a *top 5 greedy cover* of strategies 2, 8, 9, 26 and 28.
  In 5 s (in parallel) they together solve 21122 problems.

Putting PreferWatchlist into Strategies

Some options:

- Stephan Schulz's -**xUseWatchlistEvo** (EVO)
- Add EVO's CEFs to baseline strategy (uwl_evo)
  Namely: *ConjectureRelativeSymbolWeight*(Prefer,...)
- Replace all **priority** functions with **PreferWatchlist** (pref)
- Always PreferWatchlist, default to other priority function if not on watchilst (uwl).
- Matching Skolems by name and arity only (ska).

Dynamic Strategies

Some dynamic options:

- PreferWatchlistRelevant instead of PreferWatchlist: (pref $\rightarrow$ dpref)
- *dpref* with decay (ddpref)

Watchlist Creation Options

- Use all available proof clauses (too slow at around $100,000$ clauses)
- Use proof clauses from the conjecture's Mizar article (size 1-4000)
- Use most frequent clauses
- Use k-NN to suggest clauses or proofs
- TODO: Use other ML/AI method

Informal/Preliminary Testing

Not rigorous, but

- Clauses from proofs by same strategy seem more useful.
- Not removing matched **hints** helps with small watchlists. (size < 100)
- Defaultweight(PreferWatchlist) sucks: **weight** functions are good.
- Including **hints** on per-article basis seems ok.

Small Most Frequent Clauses Test

Results of *pref28* run with top N frequent clauses in respective articles (on random sample of 10,000 conjectures):

Size	10	100	256	512	1000	10000
proved	2410	2279	2211	2180	2211	2212

Results of *pref02* run as above but **hints** taken from all articles. (PPS :- processed (clauses) per second)

Size	10	100	256	512	1000	10000
proved	3275	3275	3287	3283	3248	2912
PPS	8935	9528	8661	7288	4807	575

(Note no article had more than 10000 proof clauses.)

Per-article Watchlist Benchmark

The strategies are run for 10s each on a random sample of 5000 conjectures.

Strategy	baseline	pref_baseline	pref	pref_ska	dpref	EVO	uwl
02	1238	1493	1503	**1510**	1500	1303	1247
08	1255	1296	1315	1330	1316	1300	1277
09	1075	1166	**1205**	1183	1201	1068	1097
26	1102	1133	1176	**1190**	1175	**1330**	1132
28	1138	**1141**	1141	1153	1139	1070	1139
total	1853	1910	1931	1933	1922	1659	1868

- Greedy top 5 cover 1968 conjectures (39.3%)
- *auto-schedule* covers 1744 in 50s.

K-NN Dynamic Watchlist Benchmark Round 1

Use k-nn to suggest useful proofs for conjectures (trained on full dataset run from last slide's top 5 strategies), using symbol and term-based features (walks of length 2 on formula trees and common subterms):

size	dpref02	dpref08	dpref09	dpref26	dpref28	total
4	1531	1352	1235	1194	**1165**	1957
8	1543	1366	**1253**	1188	1170	1956
16	1529	1357	1224	**1218**	1185	1951
32	**1546**	1373	1240	1218	1188	1962
64	1535	**1376**	1216	1215	1166	1935
128	1506	1351	1195	1214	1147	1907
1024	1108	963	710	943	765	1404

- A top 5 greedy cover of dprefs proves 1972 conjectures

K-NN Dynamic Watchlist Benchmark Round 2

A second round of training is done based on top 5 from round 1. From each proof P in the first round's data, we create a training example associating P's conjecture features with the names of the proofs that matched (i.e., guided the inference of) the clauses needed in P.

size	dp202	dp208	dp209	dp226	dp228	total	rnd 1 total
4	1539	**1368**	1235	1209	**1179**	1961	1957
8	1554	1376	1253	1217	1183	1971	1956
16	**1565**	1382	**1256**	1221	1181	1972	1951
32	1557	1383	1252	**1227**	1182	1968	1962
64	1545	1385	1244	1222	1171	1963	1935
128	1531	1374	1221	1227	1171	1941	1907

- The top 5 greedy cover proves 1981 conjectures

Full Runs

Runs of the top 5 cover on full dataset (57897 conjectures)

	dp02_32	dp09_8	dp26_16	dp28_4	dp08_64
total	17964	14014	14294	13449	16175
added	17964	2531	1024	760	282

Table: K-NN round 1: 22579

	dp202_16	dp209_16	dp226_32	dp228_4	dp208_4
total	18583	14486	14720	13532	16244
added	18583	2553	1007	599	254

Table: K-NN round 2: 22996

K-NN Dynamic Inheritance Watchlist Benchmark Round 2

size	ddp202	ddp208	ddp209	ddp226	ddp228	total
4	**1432**	1354	**1184**	**1203**	**1152**	1885
16	1384	1316	1176	1221	1140	1846
32	1381	1309	1157	1209	1133	1820
128	1326	**1295**	1127	1172	1082	1769

Table: The top 5 greedy cover proves 1898 problems.

- 1898 is worse than the other *pref*-based strategies, but . .

.

Divergence from baseline

	baseline	pref_baseline	pref	dpref_5	dpref2_5	ddppref2_5
baseline	0	102	134	168	173	180
pref_baseline	45	0	51	88	93	126

Table: Conjectures proven not proven by the baseline strategies.

- Dynamic watchlist with inheritance is interesting.

Grand Total Results

total	dp202_16	dp209_16	dpp226_16	dp228_4	dpp202_128
2007	1565	230	97	68	47
23192	18583	2553	1050	584	422
23192	18583	14486	14514	13532	15916

Table: Top: Cumulative sum of the 5000 test set greedy cover. The k-NN based dynamic watchlist methods dominate, improving by 2% over the baseline and article-based watchlist strategy greedy cover of 1968. Bottom: Greedy cover run on the full dataset, cumulative and total proved. The top 5 baseline strategies covered 21657. Thus there's an improvement of 7.2%.

De Morgan's laws for Boolean lattices

```
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) )
```

helped significantly by

```
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)
```

Note the weaker assumptions on H: just lower bounded and Heyting. Yet, 62 (80.5%) of the 77 clauses from the proof of

De Morgan's laws for Boolean lattices - Statistics

- 32 related proofs results in 2220 clauses on the watchlists.
- The dynamically guided proof search takes 5218 (nontrivial) given clause loops
- The ATP proof is 436 inferences long
- 194 given clauses match the watchlist in the proof search
- 120 (61.8%) of them end up being part of the proof
- 27.5% of the proof steps guided by the watchlist
- The proof search without the watchlist takes 6550 nontrivial given clause loops (25.5% more)

Final proof progress

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

Table: Final state of the proof progress for the (serially numbered) 32 proofs loaded to guide the proof of `YELLOW_5:36`. We show the computed ratio and the number of matched and all clauses.

More Examples

```
for L being B_Lattice
for X, Y being Element of L holds
(X \+\ Y) \+\ (X "/\" Y) = X "\/" Y
```

helped by

```
theorem Th23: :: LATTICES:23
for L being B_Lattice
for a, b being Element of L holds (a "/\" b)‘ = a‘ "\/" b‘
theorem Th33: :: BOOLEALG:33
for L being B_Lattice for X, Y being Element of L holds
X \ (X "/\" Y) = X \ Y
theorem :: BOOLEALG:54
for L being B_Lattice for X, Y being Element of L
st X‘ "\/" Y‘ = X "\/"
```

Statistics

- proved in 1.2 s with guidance but cannot be proved in 10 s without guidance
- 32 related proofs results in 2768 clauses on the watchlists.
- Proof search takes 4748 (nontrivial) given clause loops
- The ATP proof is 633 inferences long.
- There are 613 given clauses that match the watchlist during the proof search
- 266 (43.4%) of them end up being part of the proof
- 42% of the proof consists of steps guided by the watchlist mechanism.

More Sophisticated Progress Metric

- One idea is to try and count the number of proof clauses from a subsumed **hint** to contradiction.

Clause Re-evaluation

- (Every N given clauses), update watchlist relevance for all clauses.

More Abstract/Approximate Matching

- Do more in-line with matching modulo skolem symbol/function name.
- Try to calculate distance to nearest watchlist clause: if small, give some priority.

Adding statistical methods for clause guidance

- Pass proof-completion ratios vector to other methods, such as ENIGMA.