BliStr: The Blind Strategymaker

Josef Urban
Czech Technical University in Prague

October 18, 2015

Introduction: Large-theory Automated Reasoning

- Reason automatically in large formal theories
- Since 2003: ATP translation of the Mizar Mathematical Library (ca 50k theorems/proofs in 2014)
- Since 2005: Isabelle-HOL - ca 20k theorems/proofs, ca 40k in AFP in 2014
- Since 2012: HOL Light/Flyspeck - ca 23k theorems/proofs in 2014
- More corpora in 2014: HOL4, ACL2, Coq?

Introduction: Large-theory Automated Reasoning

- Useful for ITP - Sledgehammer, MizAR, HOL(y)Hammer
- Interesting AI research:
- We can try to learn how to prove theorems from many related proofs and proof developments
- Hopefully closer to how we learn math and science than solving small isolated problems
- *Data-driven AI* algos vs. theory-driven AI algos:
- Do not design very complex algos completely manually, but learn their parts from large amount of data
- Used to build self-driving cars, recent machine-translation systems, etc. - scary AI?

Large-theory Benchmarks/Competition

- Suitable benchmarks and competitions to foster the large-theory research:
- 2006: the MPTP Challenges
- Since 2008: Large-Theory Batch category of the CADE Automated System Competition (CASC)
- 2010: Judgement Day, 2011: MPTP2078, 2013: HH 7150
- Performance on the benchmarks/competitions corresponds to performance in the ITP deployment

The Mizar@Turing 2012 large-theory competition

- Sponsored by 3000 GBP by Google at the Turing100 conference
- The MPTP2078 benchmark: 2078 related Mizar problems in general topology
- 1000 allowed for pre-competition training with their Mizar and Vampire proofs
- 400 unknown would be used for the competition
- Just a concrete example on which the Blind Strategymaker (BliStr) was developed
- Later used also to develop ATP strategies for Flyspeck problems

Initial ATP performance in May 2012

- Measured on the 1000 training problems
- Vampire solves 691 of them with 60s time limit (Vampire well tuned on Mizar in 2010)
- E 1.6pre in auto-mode solves 519 of them with 60s time limit
- E clearly needed to improve on the problems - but how?

ATP Search Strategies

- Automated Theorem Provers (ATPs) are *programmed* using complex *search strategies*
- E prover has a nice language for strategy specification

The E strategy with longest specification in Jan 2012

```
G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

--definitional-cnf=24 --simplify-with-unprocessed-units --tstp-in
--split-aggressive --split-clauses=4 --split-reuse-defs
--simul-paramod --forward-context-sr --destructive-er-aggressive
--destructive-er --prefer-initial-clauses -winvfreqrank -c1 -Ginvfreq
-F1 --delete-bad-limit=150000000 -WSelectMaxLComplexAvoidPosPred
-H'(
4 * ConjectureGeneralSymbolWeight(
       SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight(
       PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight(PreferProcessed,1,1,1),
1 * FIFOWeight(PreferProcessed))'
-s --print-statistics --print-pid --resources-info --memory-limit=192
```

Its clause evaluation heuristic

```
G-E--_029_K18_F1_PI_AE_SU_R4_CS_SP_S0Y:

4 * ConjectureGeneralSymbolWeight(
      SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3 * ConjectureGeneralSymbolWeight(
      PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1 * Clauseweight(PreferProcessed,1,1,1),
1 * FIFOWeight(PreferProcessed)
```

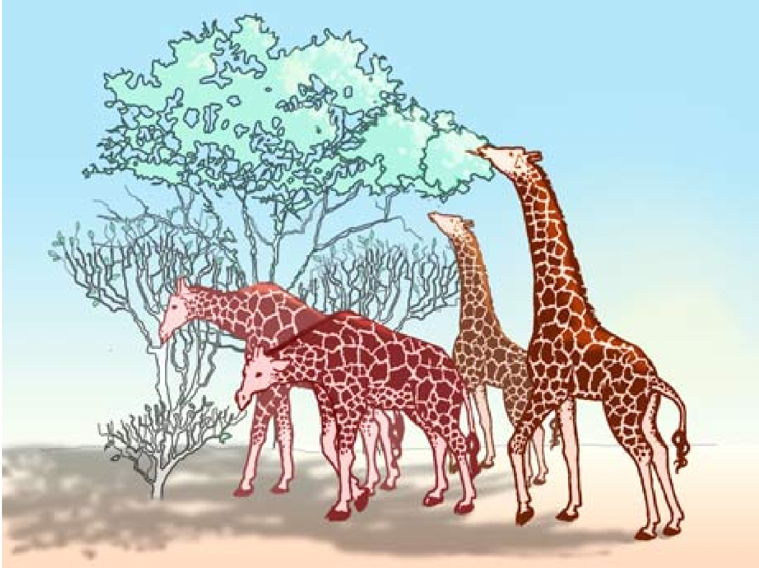
ATP Search Strategies - continued

- Different strategies fit different mathematical problems
- But most of the lemmas proved by (formal) mathematicians are not so new
- Problems often share some structure, particularly in large formal libraries
- So let us group the "similar" problems together and find good strategies for such groups
- But how again? - Certainly not manually for all of math!

Dawkins - cumulative selection (Blind Watchmaker)

- The strategy space is very large
- Guessing a good strategy at random is hopeless
- (a bat sonar “cannot be developed by a random mutation, right??”)
- It needs an “intelligent designer” (Watchmaker)!?
- But if there is a selection function that chooses the most fitting mutations
- Then the iterative process can converge very fast
- “Methinks it is like a weasel” found in 40 iterations by Dawkins’ program
- Compare it to the chance of hitting one of 27^{28} sentences at random

The Blind Watchmaker, a.k.a. Cumulative Evolution



www.personal.psu.edu/drs18/blogs/screaming_red_ass_sock_monkey/2009/06/post-8.html

The Blind Watchmaker, a.k.a. Cumulative Evolution

- The strategies are like giraffes, the problems are their food
- The better the giraffe specializes for eating problems unsolvable by others, the more it gets fed and further evolved

The Main Idea

- Evolve faster strategies on groups of *similar solvable and easy* problems.
- If they get much faster some more (related but harder) problems might become solvable.
- What are similar problems? Problems that behave similarly wrt. existing strategies (this is evolving!)
- What are easy problems? Problems that are quickly solvable by some strategy (this concept is evolving too!)
- So we need a loop that co-evolves the strategies and the concepts of similar and easy problems

The Main Strategymaking Loop

- Interleave fast strategy improvement (by Iterated Local Search) on its “similar easy” problems with the evaluation of the strategy on all problems
- That way, the notions of “similar” and “easy” evolve, and the strategies are invented on harder and harder problems
- The giraffes are getting taller and taller, covering more and more resources

ParamILS - Iterated Local Search

- Start with an initial configuration θ_0
- Loop between two steps:
- (i) perturbing the configuration to escape from a local optimum,
- (ii) iterative first improvement of the perturbed configuration.
- The result of step (ii) is accepted if it improves the previous best configuration.

Input : Initial configuration $\boldsymbol{\theta}_0 \in \boldsymbol{\Theta}$, algorithm parameters r, $p_{restart}$, and s
Output : Best parameter configuration $\boldsymbol{\theta}$ found.

for $i = 1, \ldots, r$ **do**
    $\boldsymbol{\theta} \leftarrow$ random $\boldsymbol{\theta} \in \boldsymbol{\Theta}$;
    if better($\boldsymbol{\theta}, \boldsymbol{\theta}_0$) **then** $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}$;
$\boldsymbol{\theta}_{ils} \leftarrow$ *IterativeFirstImprovement* ($\boldsymbol{\theta}_0$);
while not *TerminationCriterion()* **do**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{ils}$;
    // ===== Perturbation
    for $i = 1, \ldots, s$ **do** $\boldsymbol{\theta} \leftarrow$ random $\boldsymbol{\theta}' \in Nbh(\boldsymbol{\theta})$;
    // ===== Basic local search
    $\boldsymbol{\theta} \leftarrow$ *IterativeFirstImprovement* ($\boldsymbol{\theta}$);
    // ===== AcceptanceCriterion
    if better($\boldsymbol{\theta}, \boldsymbol{\theta}_{ils}$) **then** $\boldsymbol{\theta}_{ils} \leftarrow \boldsymbol{\theta}$;
    with probability $p_{restart}$ **do** $\boldsymbol{\theta}_{ils} \leftarrow$ random $\boldsymbol{\theta} \in \boldsymbol{\Theta}$;
return overall best $\boldsymbol{\theta}_{inc}$ found;
Procedure *IterativeFirstImprovement* ($\boldsymbol{\theta}$)
repeat
    $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$;
    foreach $\boldsymbol{\theta}'' \in Nbh(\boldsymbol{\theta}')$ *in randomized order* **do**
        if better($\boldsymbol{\theta}'', \boldsymbol{\theta}'$) **then** $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}''$; **break;**
until $\boldsymbol{\theta}' = \boldsymbol{\theta}$;
return $\boldsymbol{\theta}$;

Governing the Iterated Local Search

1. Start with an initial set of E strategies
2. Evaluate them with high time limit (5s) on all problems
3. For each strategy collect its best-solvable problems
4. This partitions the set of all solvable problems
5. Remove from such sets the problems that still take too much time
6. Run ParamILS on each strategy with low time limit (1s) on its set of cheap best-solvable problems
7. After ParamILS invents a new strategy S, evaluate S with the high time limit on all problem
8. Recompute the problem partitioning (goto 2), some problems might have become cheaper (eligible for the training phase)
9. End when there is no more improvement
10. Variations: make even smaller clusters of problems randomly - risk of overfitting

Two BliStr runs and a union of 6 runs done within 30 hours (on the 1000 Mizar@Turing training problems)

description	iterations	best strat.	solved
$BliStr_1^{400}$	37	569	648
$BliStr_3^{2500}$	23	576	643
Union of 6 runs	113	576	659

description	t_{low}	$T_{ParamILS}$	real time	user time
$BliStr_1^{400}$	1s	400s	593m	3230m
$BliStr_3^{2500}$	3s	2500s	1558m	3123m
Union of 6 runs			1800m	

More Results

- The best BliStr strategy on the 1000 training problems: 598 problems solved
- 6 best E1.6pre strategies could solve only 597 together (in 60s)
- 6 best BliStr strategies could solve 653 together (in 60s)
- The Turing100 competition (400 problems for evaluation):
- 257 MaLARea/E/BliStr vs 248 Vampire/SInE
- MaLARea/E without the new strategies: only 214
- 14195 Flyspeck/HH problems (2012):
- E1.6pre: 32.6%, using BliStr strategies: 38.4% (in 30s)

The E strategy with longest specification in May 2014

```
atpstr_my_c7bb78cc4c665670e6b866a847165cb4bf997f8a:

6 * ConjectureGeneralSymbolWeight(PreferNonGoals,100,100,100,50,50,1000,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight(PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1)
8 * ConjectureGeneralSymbolWeight(SimulateSOS,100,100,100,50,50,50,50,1.5,1.5,1)
4 * ConjectureRelativeSymbolWeight(ConstPrio,0.1, 100, 100, 100, 100, 1.5, 1.5, 1.5)
10 * ConjectureRelativeSymbolWeight(PreferNonGoals,0.5, 100, 100, 100, 100, 1.5, 1.5, 1)
2 * ConjectureRelativeSymbolWeight(SimulateSOS,0.5, 100, 100, 100, 100, 1.5, 1.5, 1)
10 * ConjectureSymbolWeight(ConstPrio,10,10,5,5,5,1.5,1.5,1.5)
1 * Clauseweight(ByCreationDate,2,1,0.8)
1 * Clauseweight(ConstPrio,3,1,1)
6 * Clauseweight(ConstPrio,1,1,1)
2 * Clauseweight(PreferProcessed,1,1,1)
6 * FIFOWeight(ByNegLitDist)
1 * FIFOWeight(ConstPrio)
2 * FIFOWeight(SimulateSOS)
8 * OrientLMaxWeight(ConstPrio,2,1,2,1,1)
2 * PNRefinedweight(PreferGoals,1,1,1,2,2,2,0.5)
10 * RelevanceLevelWeight(ConstPrio,2,2,0,2,100,100,100,100,1.5,1.5,1)
8 * RelevanceLevelWeight2(PreferNonGoals,0,2,1,2,100,100,100,400,1.5,1.5,1)
2 * RelevanceLevelWeight2(PreferGoals,1,2,1,2,100,100,100,400,1.5,1.5,1)
6 * RelevanceLevelWeight2(SimulateSOS,0,2,1,2,100,100,100,400,1.5,1.5,1)
8 * RelevanceLevelWeight2(SimulateSOS,1,2,0,2,100,100,100,400,1.5,1.5,1)
5 * rweight21_g
3 * Refinedweight(PreferNonGoals,1,1,2,1.5,1.5)
1 * Refinedweight(PreferNonGoals,2,1,2,2,2)
2 * Refinedweight(PreferNonGoals,2,1,2,3,0.8)
8 * Refinedweight(PreferGoals,1,2,2,1,0.8)
10 * Refinedweight(PreferGroundGoals,2,1,2,1.0,1)
20 * Refinedweight(SimulateSOS,1,1,2,1.5,2)
1 * Refinedweight(SimulateSOS,3,2,2,1.5,2)
```

Current Limitations and Future Work

- Term orderings and weighting schemes are another important problem-specific parameters to explore - not done yet
- Even then the E strategy language is likely not expressive enough
- Particular subproblems might benefit from very different targeted search strategies
- More difficult problems might benefit from splitting into smaller ones where a good strategy is known
- Similar to splitting ITP proofs into smaller lemmas discharged by different tactics
- Such process will likely again be highly parameterized and subject to such data-driven programming

Thanks and Advertisement

- Thanks for your attention!
- To push AI methods in math and theorem proving, we'll organize:
- AITP'16 – Artificial Intelligence and Theorem Proving
- April 3–6, 2016, Obergurgl, Austria, `aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental
- Tom Hales, John Lafferty, Bob Veroff, Noriko Arai, Stephan Schulz, Sean Holden, deep-learning people from Google, ...
- Call for abstracts of contributions next week