Automated Invention of Strategies and Term Orderings *for Vampire*

Jan Jakubův[1] Martin Suda[2] Josef Urban[1]

[1]CTU Prague [2]TU Vienna

VINO'17, Vienna, 7th September 2017

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).
- Now also for Vampire – EmpireTune

Given Clause Loop Paradigm

Problem representation

- first order clauses (ex. "$x = 0 \vee \neg P(f(x, x))$")
- posed for proof by contradiction

Given an initial set C of clauses and a set of inference rules, find a derivation of the *empty clause* (for example, by the resolution of clauses with conflicting literals L and $\neg L$).

Basic loop

```
Proc = {}
Unproc = all available clauses
while (no proof found)
{
    select a given clause C from Unproc
    move C from Unproc to Proc
    apply inference rules to C and Proc
    put inferred clauses to Unproc
}
```

Clause Selection Methods

Selection mechanisms

- symbol count (weighting)
- user-defined weighting patterns
- attribute-based restrictions (e.g., set of support)
- model-based selection (semantic guidance)
- subsumption-based selection (e.g., hints)
- selection by learned classifier

User-defined rules or scripts can be used to specify combinations of these mechanisms.

Outline

Strategy Invention: BliStr Loop

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

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

Improving Vampire Strategies

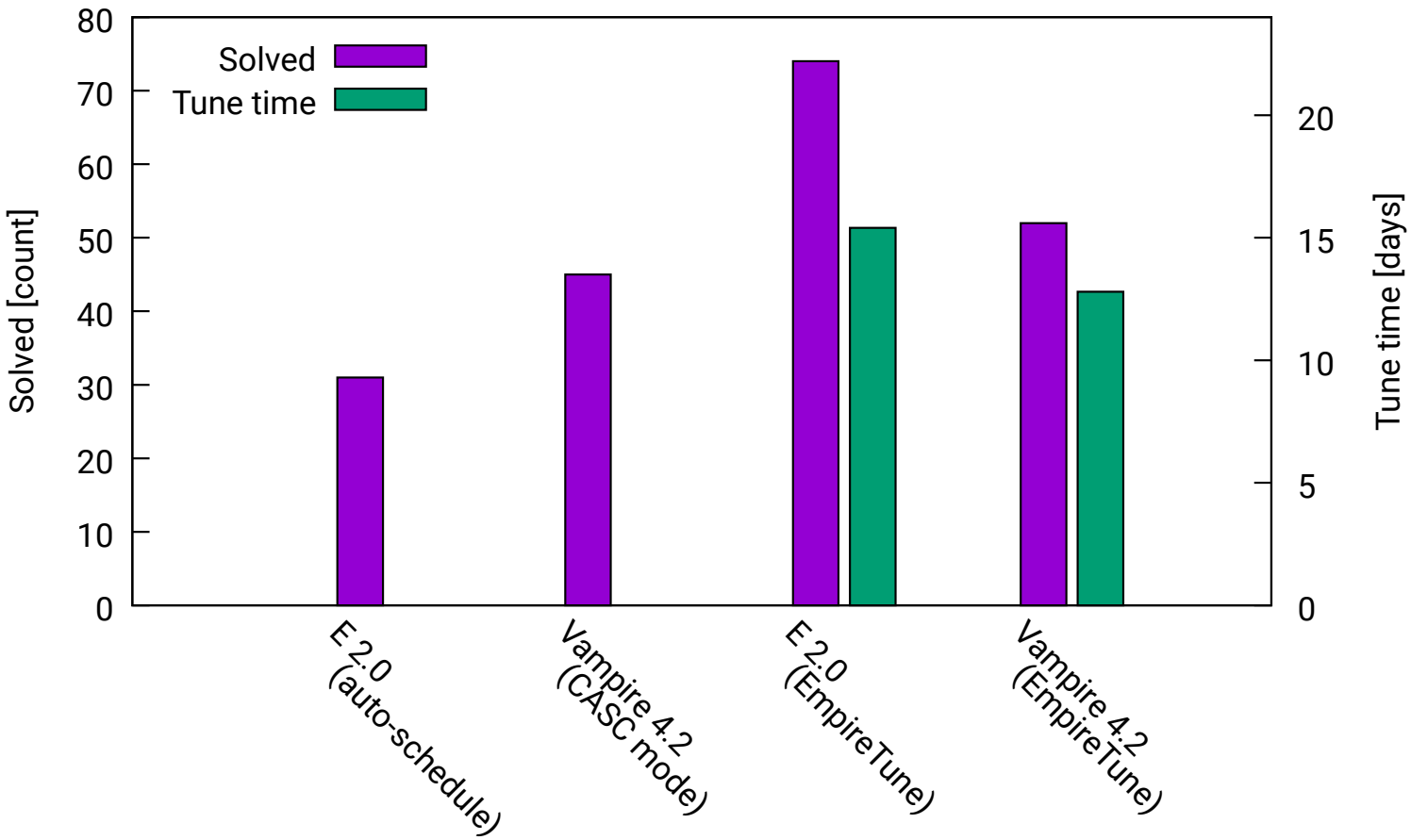
- select Vampire options for strategy selection
- . . . (Sine, saturation alg., AVATAR, inference rules)
- describe their possible values
- rule out incompatible combinations

```
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]
...
```

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



Outline

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 symbol 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

-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

-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

-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

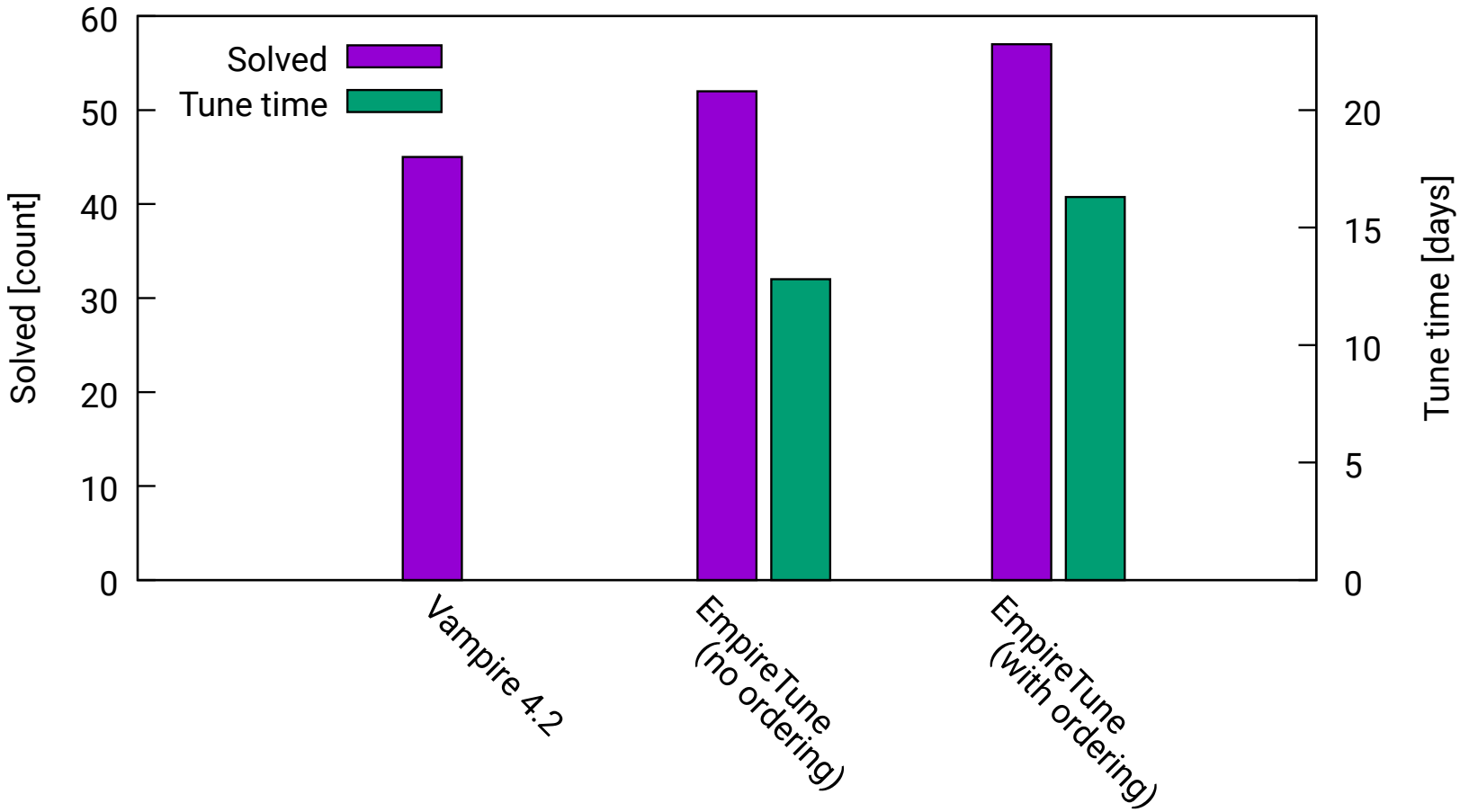
Example: Tuning Ordering in EmpireTune

-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

-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:0:7,multiply:2:9,associator:3:9

EmpireTune: Impact of Ordering Tuning



Outline

Future Work and Challenges

- Term ordering tuning is promising
- Term ordering tuning for large signatures
- Strategy invention for Prover9

Thank you

Questions?