Strong Within Benchmarks, Weak Across Them: Evaluating Neural Guidance in Vampire

Martin Suda*

Czech Technical University in Prague, Czech Republic

Gothenburg CHAIR Workshop, May 2026

*Supported by the Czech Science Foundation standard project 24-12759S.

Efficient Neural Clause-Selection Reinforcement [CADE2025]

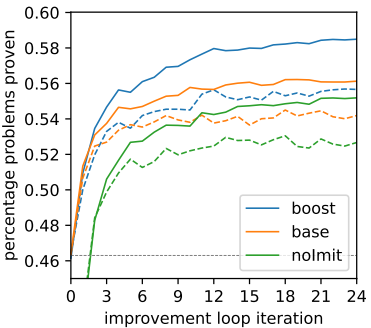
Improving Vampire's performance on TPTP by 20 %

- clause selection heuristic learned from prover's own successes via RL
- carefully engineered neural network to discriminate clauses
- single strategy setting, fixed evaluation time limit, train/test split

Efficient Neural Clause-Selection Reinforcement [CADE2025]

Improving Vampire's performance on TPTP by 20 %

- clause selection heuristic learned from prover's own successes via RL
- carefully engineered neural network to discriminate clauses
- single strategy setting, fixed evaluation time limit, train/test split



Neural Clause Evaluation

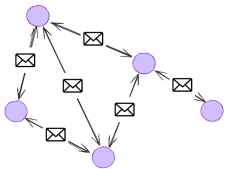
Aim for a balance between expressivity and speed of inference!

Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

One-off GNN Invocation:

- Graph Neural Networks
- name-invariant formula representations
- relatively expensive to evaluate
- therefore: we only apply once to the input

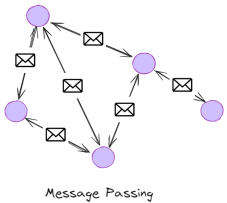


Neural Clause Evaluation

Aim for a balance between expressivity and speed of inference!

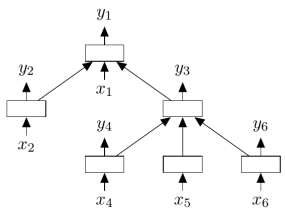
One-off GNN Invocation:

- Graph Neural Networks
- name-invariant formula representations
- relatively expensive to evaluate
- therefore: we only apply once to the input

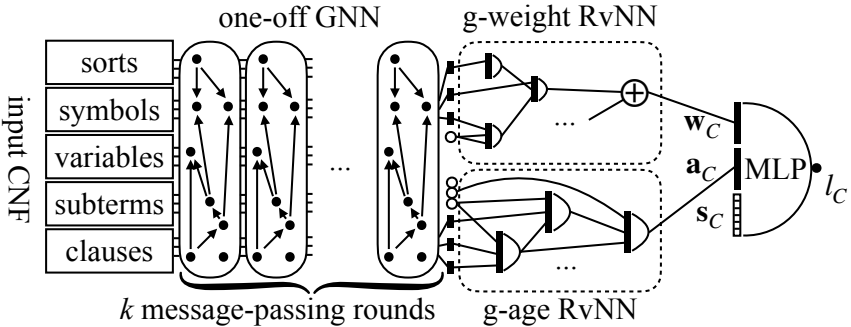


Generalizing Age and Weight with RvNNs:

- Recursive Neural Networks
- g-age: grow along the clause derivation tree
- g-weight: grow along the clause syntax tree
- share substructures (dag) and cache results



Architecture Diagram



Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$

Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$
- $A_1, \ldots, A_n, \neg G \xrightarrow{\textit{preprocessing \& clausification}} C_1, \ldots, C_m$

Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$
- $A_1, \ldots, A_n, \neg G \xrightarrow{\textit{preprocessing \& clausification}} C_1, \ldots, C_m$
- logical calculus *Calc* (resolution and superposition):
  derives new conclusions from already established premises

Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$
- $A_1, \ldots, A_n, \neg G \xrightarrow{\textit{preprocessing \& clausification}} C_1, \ldots, C_m$
- logical calculus *Calc* (resolution and superposition):
  derives new conclusions from already established premises
- search for the empty clause $\bot$ using *Calc*

Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$
- $A_1, \ldots, A_n, \neg G \xrightarrow{\textit{preprocessing \& clausification}} C_1, \ldots, C_m$
- logical calculus *Calc* (resolution and superposition):
  derives new conclusions from already established premises
- search for the empty clause $\bot$ using *Calc*

Saturation:

- systematically compute a closure of $C_1, \ldots, C_m$ under *Calc*

Saturation-based Theorem Proving

The basic pieces of a modern ATP:

- refutational (i.e., proof by contradiction):
  instead of $A_1, \ldots, A_n \models G$ we show $A_1, \ldots, A_n, \neg G \models \bot$
- $A_1, \ldots, A_n, \neg G \xrightarrow{\textit{preprocessing \& clausification}} C_1, \ldots, C_m$
- logical calculus *Calc* (resolution and superposition):
  derives new conclusions from already established premises
- search for the empty clause $\bot$ using *Calc*

Saturation:

- systematically compute a closure of $C_1, \ldots, C_m$ under *Calc*

Given-clause algorithms:

- a particular way of organizing saturation
- maintain two sets of clauses: *Passive*, *Active*
- the next given clause: via a clause selection heuristic

Outline

Outline

Out Four Datasets

TPTP: (v9.1.0); CNF+FOF+TF0; noSAT, noARI $\Rightarrow$ 15 500 problems; no split this time

Out Four Datasets

TPTP: (v9.1.0); CNF+FOF+TF0; noSAT, noARI $\Rightarrow$ 15 500 problems; no split this time

Mizar40: 57 880 "bushy" problems in the TPTP format; 15K train / 5K test

Out Four Datasets

TPTP: (v9.1.0); CNF+FOF+TF0; noSAT, noARI $\Rightarrow$ 15 500 problems;
no split this time

Mizar40: 57 880 "bushy" problems in the TPTP format;
15K train / 5K test

Isabelle: 276 363 Isabelle/Sledgehammer problems (TF0)
15K train / 5K test

Out Four Datasets

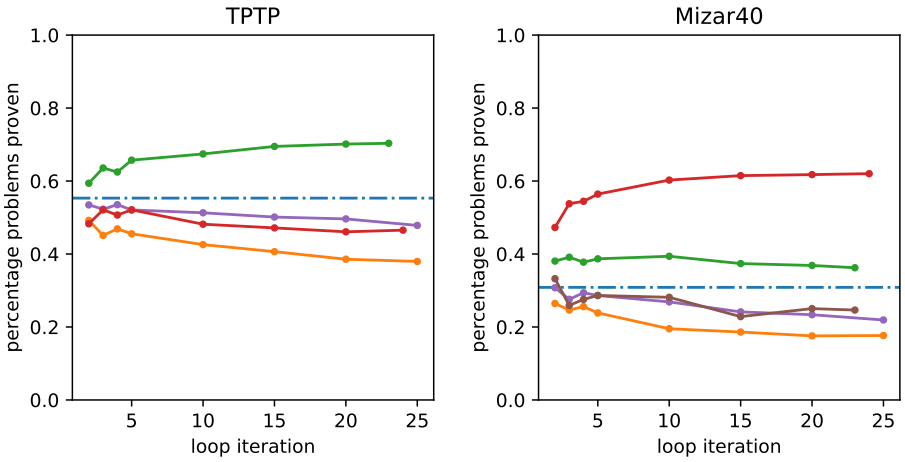
TPTP: (v9.1.0); CNF+FOF+TF0; noSAT, noARI $\Rightarrow$ 15 500 problems;
no split this time

Mizar40: 57 880 "bushy" problems in the TPTP format;
15K train / 5K test

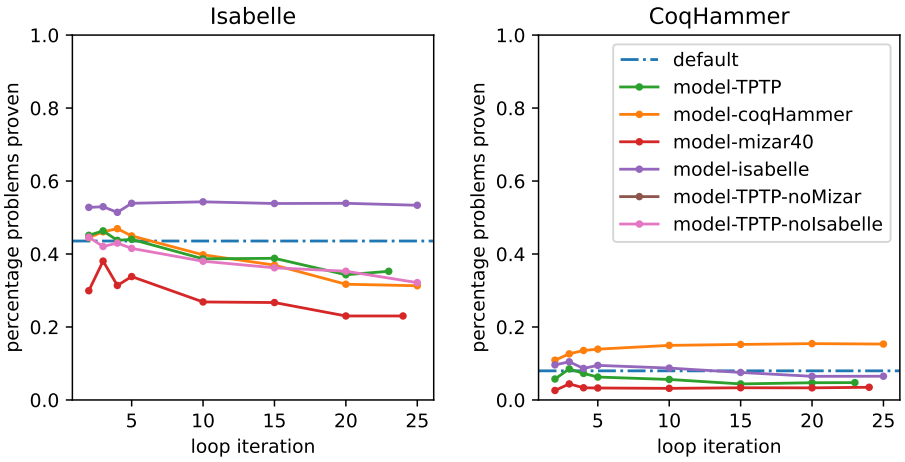
Isabelle: 276 363 Isabelle/Sledgehammer problems (TF0)
15K train / 5K test

CoqHammer: 82 366 FOL problems thanks to Blaauwbroek
30K train / 5K test (don't mix packages)

"Boostability" and (Negative) Transfer Learning I



“Boostability” and (Negative) Transfer Learning II



“Boostability” in Plain Numbers

dataset	train		test	
TPTP	(8511 → 10 864)/15 500	(+26.5 %)	—	
Mizar40	(4733 → 10 098)/15 000	(+113.4 %)	(1543 → 3101)/5000	(+101.0 %)
Isabelle	(6435 → 8788)/15 000	(+36.6 %)	(2178 → 2715)/5000	(+24.7 %)
CoqHammer	(2813 → 9799)/30 000	(+248.3 %)	(399 → 772)/5000	(+93.5 %)

Outline

Greedy Cover with Local Search

```
def hillClimbingGreed(all_problems, seed_strategy = default):
  cur_probs = all_problems
  cur_best = seed_strategy
  evaluate(seed_strategy,all_problems)
  for i in range(5):
    cur_best = best strategy on cur_probs evaluated so far
    cur_best_score = evaluate(cur_best,cur_probs)
    for j in range(3):
      for opt, values in options_to_vary:
        for val in values:
          score = evaluate(cur_best[opt:=val],cur_probs)
          if score > cur_best_score:
            cur_best_score = score
            cur_best = cur_best[opt:=val]
    print("Champion of iteration",i,"is",cur_best)
    cur_probs -= problems solved by cur_best
```

Five Complementary Strategies

```
lrs+1010_1:2_bce=on:bd=preordered:cond=fast:drc=off:fgj=on:lcm=predicate:newcnf=on:
    nm=4:nwc=1.0:sac=on:slsq=on:sp=unary_first:tgt=ground_0

lrs+1011_1:2_drc=off:er=known:fde=unused:fgj=on:nm=16:nwc=2.0:plsq=on:s2a=on:sac=on:
    slsq=on:sp=const_min_0

lrs+1011_1:1_cond=fast:drc=off:er=known:fgj=on:lcm=predicate:nm=4:nwc=1.0:sac=on:
    sfv=off:slsq=on:sp=const_frequency:tgt=ground:to=lpo_0

ott+1011_1:1_bd=preordered:fde=unused:fgj=on:newcnf=on:nm=2:plsq=on:s2a=on:sac=on:
    sp=reverse_frequency:ss=axioms:urr=on_0

lrs-1002_1:10_bce=on:cond=fast:er=known:fde=unused:fgj=on:nwc=2.0:sac=on:slsq=on:
    to=lpo_0
```

Boosting the Five Strategies

	plain proving strategies			neurally-boosted counterparts		
id	contrib.	#solved	(boost by)	#solved	contrib.	id
1	9265	9265	+21.6 %	11 264	11 264	$1'$
2	826	8395	+29.1 %	10 839	434	$2'$
3	298	8999	+18.6 %	10 675	98	$3'$
4	243	5936	+19.9 %	7118	338	$4'$
5	87	8078	+30.4 %	10 533	73	$5'$
union:	10 719	$\longrightarrow$	+13.9 %	$\longrightarrow$	12 207	
	(15.7 % of 1)				(8.4 % of $1'$)	

Making Vampire Smart Again

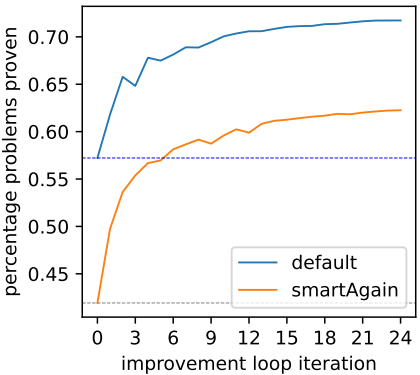
A possible way to make Vampire stupid:

- "turn off" ordering constraints (`-to incomp`); always select all literals (`-s 0`)
- so we get the prolific general resolution and paramodulation
  instead of ordered resolution and superposition

Making Vampire Smart Again

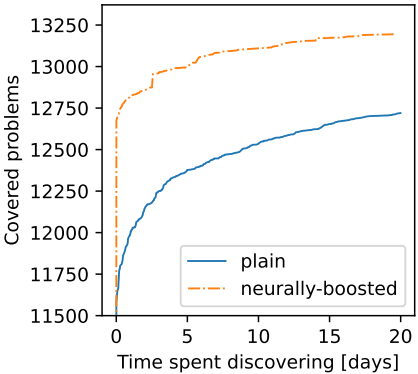
A possible way to make Vampire stupid:

- "turn off" ordering constraints (`-to incomp`); always select all literals (`-s 0`)
- so we get the prolific general resolution and paramodulation instead of ordered resolution and superposition

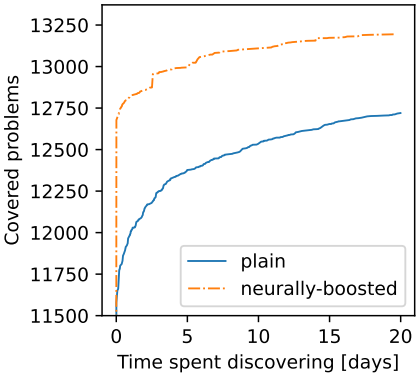


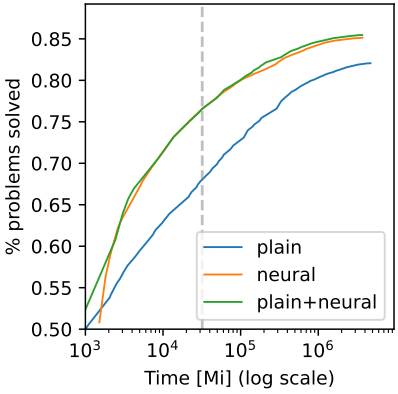
Outline

Spider-style Strategy Search and Greedy Schedule Construction [IJCAR2024]



Spider-style Strategy Search and Greedy Schedule Construction [IJCAR2024]





Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$

Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!

Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!
- Per strategy boost (on TPTP): $\sim 24\,\%$

Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!
- Per strategy boost (on TPTP): $\sim 24\,\%$
- Strategy coverage towards the limit (TPTP): 12720 $(-50 + 527 =)$ 13195

Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!
- Per strategy boost (on TPTP): $\sim 24\,\%$
- Strategy coverage towards the limit (TPTP): 12720 $(-50 + 527 =)$ 13195
- Schedule performance at 32 000 Mi mark: $67.8\,\% \rightarrow 76.1\,\%$ (i.e., $12.2\,\%$ boost)

Conclusion

Summary:

- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!
- Per strategy boost (on TPTP): $\sim 24\,\%$
- Strategy coverage towards the limit (TPTP): 12720 $(-50 + 527 =)$ 13195
- Schedule performance at 32 000 Mi mark: $67.8\,\% \rightarrow 76.1\,\%$ (i.e., $12.2\,\%$ boost)

Future Work:

- HOL
- Architecture Improvements
- Neural Strategies

Conclusion

Summary:

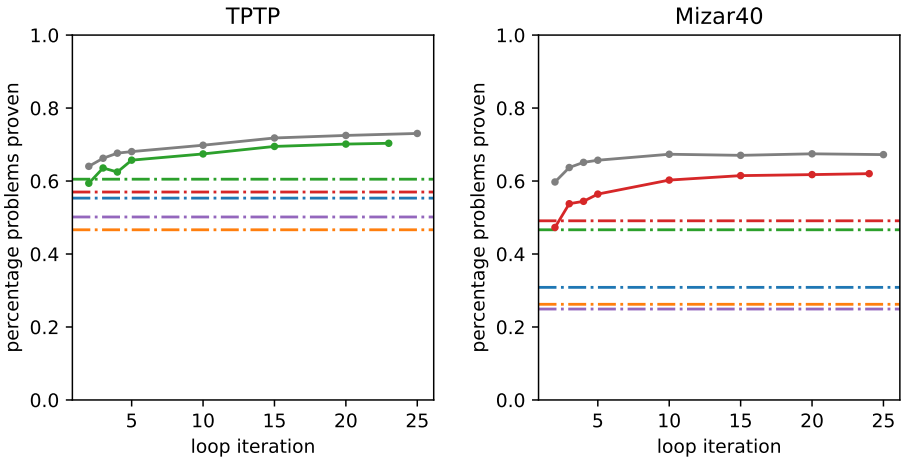
- Per dataset boost (default strategy): $20\,\% - 100\,\%$
- There is negative knowledge transfer only!
- Per strategy boost (on TPTP): $\sim 24\,\%$
- Strategy coverage towards the limit (TPTP): 12720 $(-50 + 527 =)$ 13195
- Schedule performance at 32 000 Mi mark: $67.8\,\% \rightarrow 76.1\,\%$ (i.e., $12.2\,\%$ boost)

Future Work:

- HOL
- Architecture Improvements
- Neural Strategies

Thank you!

Strategies versus Neural Boosting



Strategies versus Neural Boosting II