EPiC Series in Computing

Volume 71, 2020, Pages 38–44

Vampire 2018 and Vampire 2019.
The 5th and 6th Vampire Workshops

Aiming for the Goal with SInE*

Martin Suda

Czech Technical University in Prague, Prague, Czech Republic

Abstract

The Sumo INference Engine (SInE) is a well-established premise selection algorithm for first-order theorem provers, routinely used, especially on large theory problems. The main idea of SInE is to start from the goal formula and to iteratively add other formulas to those already added that are related by sharing signature symbols. This implicitly defines a certain heuristical distance of the individual formulas and symbols from the goal.

In this paper, we show how this distance can be successfully used for other purposes than just premise selection. In particular, biasing clause selection to postpone introduction of input clauses further from the goal helps to solve more problems. Moreover, a precedence which respects such goal distance of symbols gives rise to a goal sensitive simplification ordering. We implemented both ideas in the automatic theorem prover Vampire and present their experimental evaluation on the TPTP benchmark.

1 Introduction

A typical task in saturation-based theorem proving is to show that a certain conjecture formula C logically follows from axioms $A_1, \ldots, A_n$. In other words, the task is to establish validity of the implication $A_1 \wedge \ldots \wedge A_n \rightarrow C$. As a first step in the most commonly used refutational approach, the implication is negated, by which the task is transformed into finding an inconsistency among the set $\{A_1, \ldots, A_n, \neg C\}$. While the subsequent theorem proving technology does not, in principle, need to keep track of the "distinction in origin" between the axioms A_i and the negated conjecture $\neg C$, doing the opposite and treating the descendants of $\neg C$—typically referred to as the *goal* here—in a special way has been the core idea behind several successful proving heuristics, most notably the *set of support strategy* [19, 8], applied in various contexts [11, 1], and others [15, 14, 10, 12]. The main reason why such goal-directed search often succeeds is that since the axioms $A_1, \ldots, A_n$ are on their own typically satisfiable it must be only in combination with $\neg C$ that an inconsistency can arise.

One of the aspects of theorem proving for which the information about the goal is of central importance is premise selection, which deals with the task of selecting only a subset of the input formulas $\{A_1, \ldots, A_n, \neg C\}$, mostly in cases where the whole set would be too large for efficient processing. A well-known premise selection algorithm called *SInE* [4], produces a reasonably small subset of a large input by starting from the goal $\neg C$ and iteratively adding formulas related to the goal by sharing signature symbols. SInE is especially successful with

*Supported by the ERC Consolidator grant AI4REASON no. 649043 under the EU-H2020 programme.

extensive axiomatisations such as ontologies, whose axioms may be describing many aspects of the formalised world while only a small subset is relevant for proving the given conjecture.

In this paper we show how the computation of SInE implicitly defines a certain heuristical distance of the individual formulas and their symbols from the goal (Section 2), which can be used as an essential ingredient of new goal-directed heuristics (Section 3). We define two such heuristics: The first modifies clause selection to postpone introduction of input clauses according to their distance from the goal. The second uses the distance of symbols from the goal to describe a new goal-sensitive simplification ordering. We implemented the new ideas in the automatic theorem prover Vampire [7] and present an experimental evaluation (Section 5) on the TPTP benchmark [17]. To put the new results into perspective, we also evaluated two techniques which were already implemented in Vampire and are recalled in the Section 4.

2 The SInE Algorithm and the Distance from the Goal

In this section we recall the workings of the SInE algorithm as described by Hoder and Voronkov [4] and show how a run of the algorithm naturally defines a *heuristical distance from the goal* of 1) input formulas, and 2) signature symbols.

Let us fix an input problem P over a symbol signature S. P is modelled as a set of formulas $\{A_1, \ldots, A_n\}$ some of which are marked as *goal*.[1] The signature S consists of both function and predicate symbols. The equality symbol $\approx$ is not considered a signature symbol. We denote by $\Sigma(A)$ the set of signature symbols occurring in formula A. This notation extends naturally to sets of formulas: $\Sigma(\mathcal{A}) = \bigcup_{A \in \mathcal{A}} \Sigma(A)$. Moreover, for every symbol $s \in S$ we define $occ(s)$ as the number of occurrences of s in problem P.

As a premise selection algorithm, SInE starts with the goal formulas of P and iteratively adds those remaining formulas that are *triggered* by the symbols of formulas already added. The trigger relation, which we define next, is parametrized by two numbers to be supplied by the user: 1) a *generality threshold* $g \in \mathbb{N}$, $g \geq 0$, and 2) a *tolerance* $t \in \mathbb{R}$, $t \geq 1.0$. Now, for a symbol s and a formula A the relation $trigger(s, A)$ holds whenever $s \in \Sigma(A)$ and

- either $occ(s) \leq g$,
- or for every $s' \in \Sigma(A)$ we have $occ(s) \leq t \cdot occ(s')$.

SInE uses the number of occurrences of a symbol $occ(s)$ as a proxy for the intuitive notion of *symbol generality* with the understanding that rarely occurring symbols are specific while those occurring often are general. We do not want general symbols to trigger formulas too often, because then most formulas would get triggered almost immediately. Thus, the default value for generality threshold is $g = 0$ (the first condition never applies) and the default tolerance is $t = 1.0$ (only the most specific symbols in a formula trigger it).

Mathematically, we can model the iterative computation of SInE by defining a sequence of formulas; more specifically, of subsets of P. We start with

$$\mathcal{A}_0 = \{A \in P \mid A \text{ is marked as goal}\},$$

and for every $i > 0$ set

$$\mathcal{A}_i = \{A \in P \mid trigger(s, A) \text{ for some } s \in \Sigma(\mathcal{A}_{i-1})\}.$$

[1] A common scenario is to check whether a conjecture C follows from a set of axioms A_i, i.e. whether the formula $A_1 \wedge \ldots \wedge A_n \rightarrow C$ is logically valid. Here we assume such formula has already been negated for refutational theorem proving and model problem P as the set $\{A_1, \ldots, A_n, \neg C\}$ with $\neg C$ marked as the goal.

SInE can be run only for a given number of iterations k (where k would be another parameter), yielding the set $\bigcup_{i\leq k}\mathcal{A}_i$ as the final result. An alternative that we use here for the purpose of defining our distance from the goal, is to keep running while new formulas are being triggered. Formally, let l be the smallest index such that $\bigcup_{i\leq l}\mathcal{A}_i = \bigcup_{i\leq l+1}\mathcal{A}_i$. It is clear that the computation can be stopped at the index l, since no new formulas would be added later.

Finally, we can define our *distance from the goal*. For every $A \in P$ we set

$$d(A) = \begin{cases} \text{the smallest } i \text{ such that } A \in \mathcal{A}_i, & \text{if } A \in \bigcup_{i\leq l}\mathcal{A}_i, \\ l+1, & \text{otherwise.} \end{cases}$$

In particular, $d(A) = 0$ exactly for those formulas $A \in P$ marked as goal. Analogously, for every $s \in S$ we set

$$d(s) = \begin{cases} \text{the smallest } i \text{ such that } s \in \Sigma(\mathcal{A}_i), & \text{if } s \in \Sigma(\bigcup_{i\leq l}\mathcal{A}_i), \\ l+1, & \text{otherwise.} \end{cases}$$

The choice of value $l+1$ as the distance assigned to a formula/symbol that does not occur in the computed fixpoint $\bigcup_{i\leq l}\mathcal{A}_i$ is purely pragmatic and other (larger) values could be experimented with in specific contexts of applying the heuristic.

We also note that (non-goal) formulas that do not contain any symbols (and thus can never be triggered in principle) can be treated specially and assigned distance $d(A) = 0$. This can be achieved by simply changing the definition of $\mathcal{A}_0$ to

$$\mathcal{A}_0 = \{A \in P \mid A \text{ is marked as goal or } \Sigma(A) = \emptyset\}. \tag{1}$$

Formulas without symbols, such as $\forall XY(X = Y)$, are quite rare, but often represent constraints that should not be ignored. We therefore define $\mathcal{A}_0$ according to (1) in our implementation.

3 Making Use of the Distance

In this section, we describe the two new theorem proving techniques that make use of the SInE-derived distance from the goal.

SInE to Age The first use of our heuristic distance is for clause selection. By default, Vampire selects clauses for activation by alternating between picking the current oldest clause (using a queue sorted by the *age* criterion) and the current lightest clause (using a queue sorted by the *weight* criterion). This alternation happens in a certain fixed *age-weigth* ratio.[2]

While the weight of a clause simply corresponds to the number of its symbols (although see below for a possible variation), the age of a clause is established as follows: Input clauses are assigned age 0 and each clause C derived by a generating inference, such as resolution or superposition, receives age $a_{max}+1$, where a_{max} is the maximum age among C's parents. A clause derived by a reduction, such as demodulation or subsumption resolution, receives the same age as its main parent, i.e. the reduced clause (as opposed to the reducing one).

The idea of the *"sine to age"* heuristic is to change the age of input clauses (from the default 0) to their established distance from the goal. Thus only the goal clauses will have age 0, while clauses further from the goal will receive higher age. This will postpone their introduction to the search (unless they are light and will get selected because of that via the weight criterion).

[2]Please consult the recent work by Rawson and Reger [9] for more details and an interesting extension.

SInE to Predicate Levels This second technique modifies the way Vampire constructs its simplification ordering used for constraining inferences and has been suggested by Kovács et al. [6]. Predicate levels are a cheap way of simulating transfinite Knuth-Bendix orders (KBO) for the special case of "top-level" comparison of predicate symbols in atoms. As explained in [6], when comparing two atoms $p(s1, \ldots, s_m)$ and $q(t_1, \ldots, t_n)$ one first compares the levels of p and q. If the level of p is greater, it is decided that $p(s1, \ldots, s_m) \succ q(t_1, \ldots, t_n)$. Ordinary KBO is only used for comparing two atoms having predicates of the same level.

Now the idea from [6] is to assign predicate levels to symbols based on how "close" they are to the goal. The intuition should be that symbols closer to the goal should receive higher levels. We implemented this idea using the SInE-derived distance and both with the suggested and the opposite "polarity" in order to verify the intuition.

4 Other Goal-directed Heuristics

To put the new techniques into perspective in our experiments, we also compare them to two other goal-directed heuristics that were already implemented in Vampire. Here we briefly describe the ideas behind each.

Non-goal Weight Coefficient As mentioned above, one of the two criteria for clause selection in Vampire is clause weight, which is normally computed as the number of symbols in the clause. Optionally, for every clause which has *not* been derived from a goal clause, this value is multiplied by a *non-goal weight coefficient*, a floating point parameter greater or equal to 1.0. Thus the input goal clauses and all the clauses directly or indirectly derived from them retain the actual symbol count as weight, while all other clauses are made artificially heavier and thus their selection tends to be postponed.

Set of Support While the original idea behind the well-known *set of support* strategy [19, 8] is slightly more general than what we describe here, the way it is realised in Vampire makes it another example of a goal-directed search technique. Set of support modifies the way in which we initialise the saturation algorithm (see e.g. [7] for a recap of the terminology). Instead of putting all the input clauses into the *Passive* set and initialising the *Active* set as empty, as usually done, the set of support strategy puts all the non-goal input clauses immediately into *Active* and only the goal input clauses in to *Passive* set. The immediate consequence is that only clauses that have at least one goal clause as an ancestor in their derivation tree are ever derived. Unlike the previously described techniques, the set of support strategy is in general incomplete but nevertheless very useful for solving hard problems.

5 Experiment

We implemented the described ideas in automatic theorem prover Vampire [7] version 4.4. We evaluated their impact on prover's performance on the problems from the TPTP library [17] version 7.2.0. The experiments were run on the StarExec cluster [16], whose nodes are equipped with Intel Xeon 2.4GHz processors.

A Base Strategy As a baseline, we chose a strategy using the *discount* saturation loop (for stability of results[3]) with *age-weight ratio* 10 (which works well with discount), turning off the *AVATAR architecture* [18] (to stay as close as possible to the standard saturation-based paradigm), and setting *symbol precedence* to `frequency` (which tends to perform better than the default).[4]

We set the *time limit* to 60 s and disabled the default *memory limit*.

Other strategies All the remaining evaluated strategies were derived from the base one by always changing a single option as follows:

- `nwc-2.0`, `nwc-5.0`, `nwc-10.0` set the *non-goal weight coefficient* to the respective values 2.0, 5.0, and 10.0.
- `s2a` enables the *"sine to age"* option.

  Only the goal input clauses are assigned the default age 0, while the remaining ones get artificially higher age values according to their distances from the goal.
- `s2pl-on` and `s2pl-no` deal with the option *"sine to predicate levels"*.

  Turning the option `on` makes predicate symbols closer to the goal have high precedence in the ordering (conjecture predicates will tend to occur in maximal literals), while the value `no` does the opposite (predicates furthest from the goal will tend to be maximal).
- `sos-on`, `sos-all` enable the *set of support* processing of the input clauses, i.e. all the input clauses except those derived from the goal are immediately put into the *Active* container. `sos-all` additionally makes all the literals of those input clauses selected (to enable more interaction during the follow-up inferences).

For the `s2a` and `s2pl` options derived from SInE, the *generality threshold* and *tolerance* parameters were set to their default values, i.e. to 0 and 1.0, respectively.

Results In the used versions of the TPTP library, there are 17 571 problems in a format accessible to Vampire (either CNF, FOF, or TF0). Out of these, 16 754 problems contain at least one goal formula and can thus in principle trigger different behaviour of our strategies.

Table 1 shows the number of problems solved by each strategy.[5] The Δbase column shows the difference to the base strategy. We can see that while the `nwc` strategies and `s2a` are better than `base`, overall performance goes down when enabling `s2pl` or `sos`.

The true value of a new strategy, however, often does not so much lie in its absolute performance, but in how well it can contribute to solving problems not solvable by other strategies. To estimate such contribution, Table 1 shows the number problems *uniquely* solved by each strategy, first (I) across all the strategies, for the second time (II) only for the best strategy in each group (to prevent similar strategies from "stealing" each others uniques). We can see that the overall winner in this regard is `sos` with 419 uniques by `sos-all`. `nwc` and `s2a` both seem to be reasonably interesting as well. On the other hand, the value of the `s2pl` technique, both in absolute terms and in the number of uniques, does not seem to be convincing.

[3] The default *limited resource strategy* [13] is sensitive to timing measurements and thus essentially non-deterministic.

[4] Originally inspired by E prover [14]; to Vampire, `frequency` has been introduced in [5].

[5] Note that we count only proofs (not the found saturations), because proving is the primary mode for which we want to find new good strategies.

strategy	refuted	Δbase	uniques I	uniques II
`base`	8196	0	3	3
`nwc-2.0`	8513	317	14	
`nwc-5.0`	8525	329	6	
`nwc-10.0`	8599	403	27	251
`s2a`	8507	311	149	207
`s2pl-on`	8148	−48	11	25
`s2pl-no`	8053	−143	32	
`sos-on`	5243	−2953	100	
`sos-all`	7530	−666	298	419

Table 1: Number of problems refuted by each tested strategy within 60 s per problem.

6 Conclusion

In this short paper, we have shown that the computation of the SInE premise selection algorithm implicitly defines a heuristical distance of input formulas and symbols from the goal. This distance can be used to design new goal directed extensions of the search algorithm. We proposed and implemented two such extensions, "sine to age" and "sine to predicate levels", and evaluated them on the TPTP benchmark. At least "sine to age" seems to be a very promising technique, which should be further explored. In particular, equating a single step from the goal with a one-step increase in the imposed age seems arbitrary and a different "unit alignment" could be even more effective. As part of future work, we also plan to explore the effect of using non-default values of the generality threshold and tolerance parameters.

References

[1] Ahmed Bhayat and Giles Reger. Set of support for higher-order reasoning. In Boris Konev, Josef Urban, and Philipp Rümmer, editors, *Proceedings of the 6th Workshop on Practical Aspects of Automated Reasoning co-located with Federated Logic Conference 2018 (FLoC 2018), Oxford, UK, July 19th, 2018*, volume 2162 of *CEUR Workshop Proceedings*, pages 2–16. CEUR-WS.org, 2018.

[2] Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors. *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*. Springer, 2011.

[3] Pascal Fontaine, editor. *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*. Springer, 2019.

[4] Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Bjørner and Sofronie-Stokkermans [2], pages 299–314.

[5] Jan Jakubuv, Martin Suda, and Josef Urban. Automated invention of strategies and term orderings for vampire. In Christoph Benzmüller, Christine L. Lisetti, and Martin Theobald, editors, *GCAI 2017, 3rd Global Conference on Artificial Intelligence, Miami, FL, USA, 18-22 October 2017*, volume 50 of *EPiC Series in Computing*, pages 121–133. EasyChair, 2017.

[6] Laura Kovács, Georg Moser, and Andrei Voronkov. On transfinite Knuth-Bendix orders. In Bjørner and Sofronie-Stokkermans [2], pages 384–399.

[7] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Confer-*

ence, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[8] William McCune. OTTER 2.0. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, volume 449 of *Lecture Notes in Computer Science*, pages 663–664. Springer, 1990.

[9] Michael Rawson and Giles Reger. Old or heavy? Decaying gracefully with age/weight shapes. In Fontaine [3], pages 462–476.

[10] Giles Reger and Martin Riener. What is the point of an SMT-LIB problem? In *16th International Workshop on Satisfiability Modulo Theories*, 2018.

[11] Giles Reger and Martin Suda. Set of support for theory reasoning. In Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov, editors, *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations, Maun, Botswana, May 7-12, 2017*, volume 1 of *Kalpa Publications in Computing*. EasyChair, 2017.

[12] Giles Reger and Andrei Voronkov. Induction in saturation-based proof search. In Fontaine [3], pages 477–494.

[13] Alexandre Riazanov and Andrei Voronkov. Limited resource strategy in resolution theorem proving. *J. Symb. Comput.*, 36(1-2):101–115, 2003.

[14] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th LPAR, Stellenbosch*, volume 8312 of *LNCS*. Springer, 2013.

[15] Stephan Schulz and Martin Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2016.

[16] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec, a cross community logic solving service. https://www.starexec.org, 2012.

[17] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[18] Andrei Voronkov. AVATAR: the architecture for first-order theorem provers. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.

[19] Lawrence Wos, George A. Robinson, and Daniel F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12(4):536–541, October 1965.