Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies

Rafael S. Gonçalves, Bijan Parsia, and Uli Sattler

School of Computer Science
University of Manchester
Manchester, United Kingdom

Abstract. Due to the high worst case complexity of the core reasoning problem for the expressive profiles of OWL 2, ontology engineers are often surprised and confused by the performance behaviour of reasoners on their ontologies. Even very experienced modellers with a sophisticated grasp of reasoning algorithms do not have a good mental model of reasoner performance behaviour. Seemingly innocuous changes to an OWL ontology can degrade classification time from instantaneous to too long to wait for. Similarly, switching reasoners (e.g., to take advantage of specific features) can result in wildly different classification times. In this paper we investigate performance variability phenomena in OWL ontologies, and present methods to identify subsets of an ontology which are performance-degrading for a given reasoner. When such (ideally small) subsets are removed from an ontology, and the remainder is much easier for the given reasoner to reason over, we designate them "hot spots". The identification of these hot spots allows users to isolate difficult portions of the ontology in a principled and systematic way. Moreover, we devise and compare various methods for approximate reasoning and knowledge compilation based on hot spots. We verify our techniques with a select set of varyingly difficult ontologies from the NCBO BioPortal, and were able to, firstly, successfully identify performance hot spots against the major freely available DL reasoners, and, secondly, significantly improve classification time using approximate reasoning based on hot spots.

1 Introduction

Reasoning tasks on ontologies expressed in a rich description logic such as that underlying OWL 2 have a high worst case complexity. As a consequence, reasoning time can be highly unpredictable: seemingly innocuous changes to an ontology might shift reasoning time from seconds to days; different reasoners might have wildly different behaviour on the same input. Even seasoned reasoner developers do not have a mental performance model sufficient to deal with many, particularly novel, cases (indeed, this fact keeps reasoner optimisation research a lively area).

Mere high worst case complexity, of course, does not entail unpredictability. The difficulty of determining the satisfiability of propositional k-CNF formulae (the k-SAT problem), for example, is highly predictable by attending to the "density" (i.e., the ratio of number of clauses to number of distinct variables) of a formula. Not only is it predictable, but there is an increasingly sophisticated theoretical understanding of this behaviour. This predictability has been observed in various modal logics which correspond to the description logics commonly used as ontology languages [14, 10]. However, several observations belie the utility of these results: 1) Even for comparatively simple logics such as $\mathcal{ALC}$ the number of parameters becomes unwieldy: while propositional logic has two main parameters (for a given size, k!) — number of clauses (L) and number of variables (N) — $\mathcal{ALC}$ adds (at least) modal depth (d), the number of roles (i.e., modalities, m), and the proportion of modal to propositional atoms [10]. 2) The inputs are highly regimented and bear little relationship to the sorts of formulae found in practice, especially in manually crafted artifacts such as ontologies. For example, all ontologies have axioms, not just concept expressions, these axioms often "break up" complex concepts, and reasoners exploit this fact.[1] Thus, to predict behaviour of realistic or naturally occurring ontologies, we need to understand even more parameters (perhaps dozens), and normalizing away that complexity is unlikely to be helpful. 3) Reasoners have different suites of optimizations and even underlying calculi, thus respond differently to these inputs.

Together, these observations suggest that users crafting ontologies are likely to be surprised[2] by the enormous variation in performance behaviour which does not relate intuitively to the changes they make (either in the ontology or in the reasoner used). Three basic phenomena startle users: 1) An ontology which takes seconds to classify[3] in one reasoner, effectively fails to terminate with another. 2) Ontologies of similar size and apparent complexity take wildly different times to classify on the same reasoner. 3) Apparently innocuous changes to a single ontology result in large increases (or decreases) in classification time.[4] Of course, the primary negative phenomenon is excessive reasoning time.

The most prominent, principled way to cope with this problem is to shift to a less expressive logic, such as OWL $\mathcal{EL}$, for which classification is decidable in polynomial time. Reasoning in $\mathcal{EL}$ (and similar logics) is not only polynomial (in general) but has proven to be rather robust to novel input [1, 2]. This move is not always possible, as it involves severe limitations on what can be expressed. Similarly, approximate reasoning (i.e., giving up on soundness or completeness) can make reasoning performance significantly better and more predictable, but at the cost of increased uncertainty about the results [18, 16, 15]. In practice, users often modify their ontologies based on folk wisdom ("negation is hard", "inverses are hard"), on bespoke advice from reasoner developers, or randomly.

[1] "In realistic KBs, at least those manually constructed, large and complex concepts are seldom described monolithically, but are built up from a hierarchy of named concepts whose descriptions are less complex." [9]

[2] "[Reasoner] performance can be scary, so much so, that we cannot deploy the technology in our products." — Michael Shepard `http://lists.w3.org/Archives/Public/public-owl-dev/2007JanMar/0047.html`

[3] Throughout, we focus on *classification* as the key reasoning task, as it is the most prevalent service invoked by ontology developers.

[4] Esp. distressing are removals that increase time, and additions which decrease it dramatically.

We need a better understanding of reasoning performance variability, or at least methodologies for analyzing it in particular cases. The contributions of this paper are as follows: for an ontology $\mathcal{O}$ that a reasoner R takes 'too long' to classify, we have *designed and thoroughly evaluated* (1) *a technique for analyzing the performance variability of* R *on* $\mathcal{O}$, (2) *a technique to isolate subsets of* $\mathcal{O}$ *that contribute negatively to* R*'s high classification time, so called* hot spots, and (3) *a series of techniques to approximate the hot spot in* $\mathcal{O}$.

Firstly, we have verified, via technique (1), that there exist two kinds of performance profiles; an ontology-reasoner pair can be performance "heterogeneous" or performance "homogeneous", depending on whether there are certain kinds of performance variability between subsets of the ontology. Secondly, we identified very small subsets of an ontology whose removal causes a significant decrease in classification time, i.e., hot spots, using technique (2). Indeed we show that performance heterogeneous ontology-reasoner pairs are highly likely to have such subsets which are detectable by our methods. Thirdly, and finally, we show that if there is a hot spot for an ontology-reasoner pair, then we can approximate it in such a way that our criteria for a hot spot (i.e., classification time boost and size) are maintained.

2 Preliminaries

We assume the reader to be reasonably familiar with ontologies and OWL [22], as well as the underlying description logics (DLs) [8]. An ontology $\mathcal{O}$ is a set of axioms, and its *signature* (the set of individuals, concept and role names used) is denoted $\widetilde{\mathcal{O}}$. We use the notion of a *locality-based module* [5], which is a subset of an ontology $\mathcal{O}$ that preserves all consequences of $\mathcal{O}$ w.r.t. to a signature Σ. An x-module $\mathcal{M}$ extracted from an ontology $\mathcal{O}$ for a signature Σ is denoted x-$mod(\Sigma, \mathcal{O})$, for x one of $\top\bot^*$, $\top$ or $\bot$. A justification $\mathcal{J}$ of a consequence α is a $\subseteq$-minimal subset of an ontology $\mathcal{O}$ that is sufficient for α to hold [11]. The reasoning time of an ontology $\mathcal{O}$ using reasoner R, denoted $\mathrm{RT}(\mathcal{O}, R)$,[5] comprises the time for consistency checking, classification (computing atomic subsumptions) and coherence (concept satisfiability). The set of atomic subsumptions resulting from the classification of an ontology $\mathcal{O}$ is denoted $Cl(\mathcal{O})$.

3 Materials

In order to test our methods we need a reasonable corpus of "problem" ontologies. We derived one from the NCBO BioPortal, a large collection of user contributed, "working" ontologies covering a wide range of biomedical domains [13]. We gathered all ontologies from the BioPortal, and performed a reasoner performance test across this corpus. Four major, freely available DL reasoners were used: Pellet (v2.2.2) [20], HermiT (v1.3.6) [19], FaCT++ (v1.5.3) [21], and JFact (v0.9).[6] The experiment machine is an Intel Xeon Quad-Core 3.20GHz,

[5] When R is clear from the context, we also use $\mathrm{RT}(\mathcal{O})$.

[6] `http://jfact.sourceforge.net/`

with 32GB DDR3 RAM dedicated to the Java Virtual Machine (JVM v1.5). The system runs Mac OS X 10.6.8, all tests were run using the OWL API v3.3 [7].

The entire BioPortal corpus contains 216 ontologies. We discarded all ontologies with reasoning times, for all reasoners, below 60 seconds (i.e., the "easy" ontologies). This leaves 13 ontologies, 3 of which did not classify within 10 hours: the IMGT[7] ontology with Pellet, GALEN[8] with all reasoners, and GO-Ext. (Gene Ontology Extension),[9] with FaCT++ and JFact.

The naive approach to determining heterogeneity is to enumerate the "acceptably" small subsets of the ontology and measure the classification time for each. Given that our ontologies range from 100s to over 100,000 axioms, this is obviously infeasible. Random testing of acceptably small subsets might be effective assuming that a sufficiently large proportion of those subsets were, in fact, hot spots, though our preliminary experiments in this direction were unpromising. Instead, we performed two sorts of heterogeneity detection. In the first, "coarse grained" method, we classify ontology-reasoner pairs as performance heterogeneous or homogenous by attending to performance fluctuations (or lack thereof) over relatively large, evenly increasing subsets of the ontology. In the second, we apply two simple heuristics for selecting candidate subsets, and then verify whether they conform to our hot spot criteria. The second method directly verifies our heterogeneity condition.

4 Coarse Grained Prediction

For current purposes, we focus on performance variability of a single reasoner for a given ontology. In particular, we are always examining the difference in reasoning time of select subsets of a given, hard-for-a-specific-reasoner ontology. We do this for several reasons: 1) it simulates a common user scenario (e.g., editing or trying to "optimize" an ontology) and 2) we are investigating the background assumption that ontologies which are difficult (e.g., approach the worst case) are often so in a "fragile" way, i.e., their performance is sensitive to small changes.

We say that an ontology is *performance homogenous* for a reasoner if there is a linear factor L and variable k such that for all $\mathcal{M} \subseteq \mathcal{O}$ and for $k \cdot |M| = |\mathcal{O}|$, we have that $L \cdot k \cdot \mathrm{RT}(\mathcal{M}) \approx \mathrm{RT}(\mathcal{O})$. An ontology which is not *performance homogeneous* we call *performance heterogeneous*. It is important to note that, in both cases, the performance profile of the ontology and its subsets may be *predictable* (even if we currently do not know how to predict it).

In this experiment, each ontology is divided into 4 and 8 random subsets of equal size, and the classification times of these subsets as increments are measured (i.e., we measure, for the 4-part division, $\mathrm{RT}(\mathcal{O}_1)$, $\mathrm{RT}(\mathcal{O}_1 \cup \mathcal{O}_2)$, $\mathrm{RT}(\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}_3)$, $\mathrm{RT}(\mathcal{O})$, where $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ are subsets of $\mathcal{O}$). Both mea-

[7] `http://www.imgt.org/IMGTindex/ontology.html`
[8] `http://www.co-ode.org/galen/`
[9] `http://www.geneontology.org/`

surements are carried out several times per ontology (at least 10, though often more), where each time the list of axioms in $\mathcal{O}$ is shuffled.

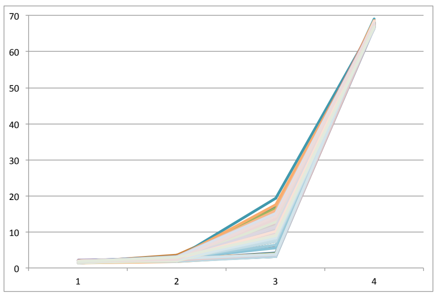
Note that we are testing a very small number of subsets of each ontology, so, in principle, that we see "smooth" behaviour could be due to insufficient sampling. However, because each increment is rather large, we hope that it will contain (in a behaviour exhibiting way) any hot spots.

Overall 4 out of 13 ontology/reasoner pairs exhibit roughly linear performance growth in our tests (see Figures 1c and 1d for characteristic graphs). GALEN proved infeasible to work with (even only half the ontology gave reasoning times of over 10 hours), and was discarded. The remainder exhibited non-linear and sometimes highly variable performance behaviour. For example, Figure 1e shows that even the very coarse, 4-part division method can detect strange performance patterns, although the more fine grained, predictably, is more detailed (Figure 1f). Contrariwise, Figure 1a shows a rather smooth, if non-linear, curve. It is tempting to think that that smoothness indicates a relatively predictable performance profile, but as we see in the more fine grained view (Figure 1b) this is not true. However, this supports (though, obviously, does not confirm) our hypothesis that ontologies with non-linear growth, whether smooth or jaggy, are performance heterogeneous. In our corpus, they certainly exhibit surprising variability.

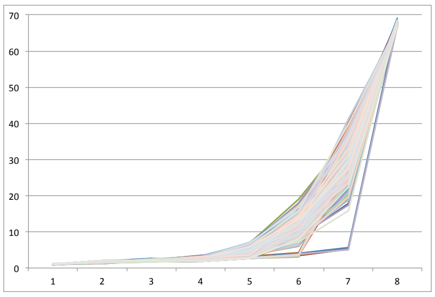
While we were unable to run this test sufficient enough times to attain statistical significance for all ontologies, the data gathered is already highly suggestive of reasoner performance behaviour on our test corpus. During the execution of this experiment we noted a curious phenomenon: While in most ontologies we managed to achieve convergence on the overall classification time on each run, in the GO-Ext ontology this did not happen. Surprisingly, the classification time of GO-Ext with Pellet, under exactly the same experimental conditions, varies from seconds to hours; more specifically, the range is from 27 seconds to 1 hour and 14 minutes (Figure 2). A unique case as it may be (in our corpus), it suffices to illustrate not only the need for performance analysis solutions, but also the difficulty of the problem in cases such as this one.

5 Performance Hot Spots

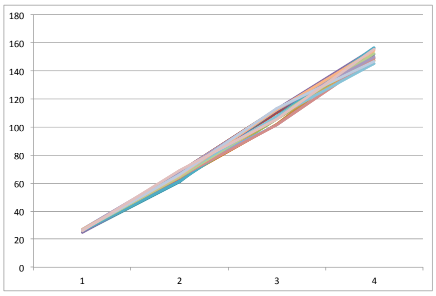
We hypothesise that if an ontology is *performance heterogeneous* for a reasoner, then there exists at least one "small" subset of that ontology whose removal results in a "significant" change in the classification time (positive or negative). That is, when there exists a subset $\mathcal{M}$ of a given ontology $\mathcal{O}$ such that (1) $\mathcal{M}$ is "acceptably small" (typically, $|\mathcal{M}| \ll |\mathcal{O}|$), and (2) $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}) \ll (\mathit{or} \gg) \, \mathrm{RT}(\mathcal{O})$. We call such a subset $\mathcal{M}$, which witnesses the performance heterogeneity of $\mathcal{O}$, a "hot spot", by way of analogy with program profilers. The analogy is imperfect as we cannot say whether such bits themselves consume an inordinate amount of time, or whether they have some more diffuse triggering effect.



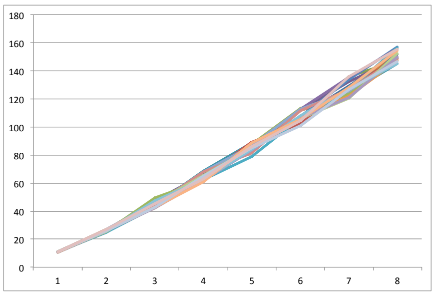
(a) ChEBI 4-part division (Pellet)



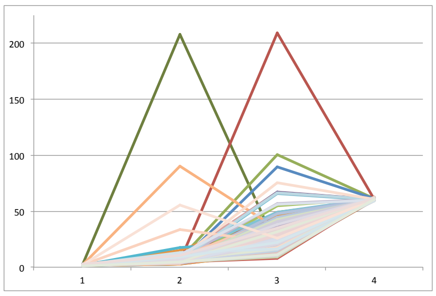
(b) ChEBI 8-part division (Pellet)



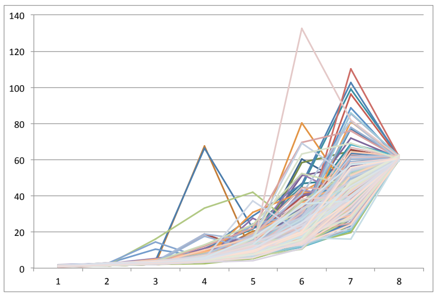
(c) Gazetteer 4-part division (HermiT)



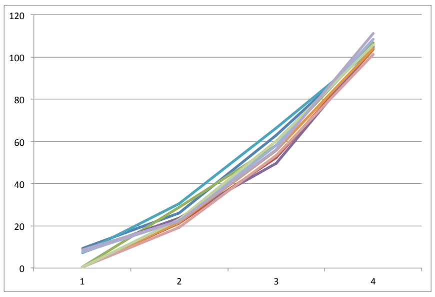
(d) Gazetteer 8-part division (HermiT)



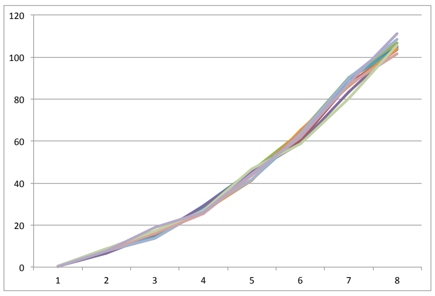
(e) EFO 4-part division (Pellet)



(f) EFO 8-part division (Pellet)

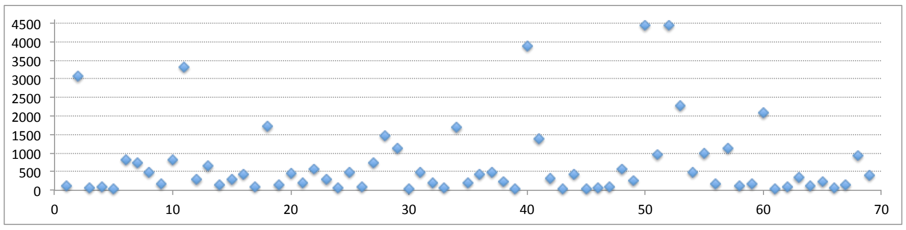


(g) ICF 4-part division (HermiT)

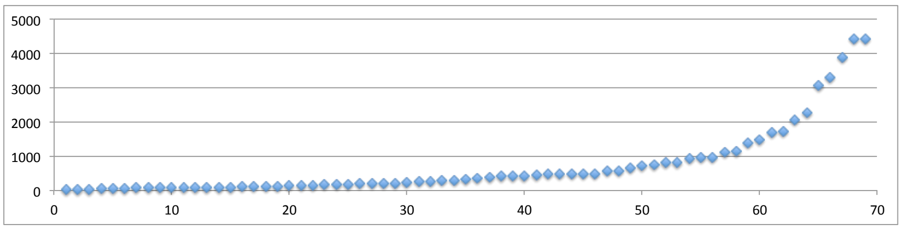


(h) ICF 8-part division (HermiT)

Fig. 1: Performance heterogeneity tests of select ontologies. All times in seconds.



(a) Times in chronological order.



(b) Times in ascending order.

Fig. 2: Classification times (in seconds) of the GO-Ext ontology with Pellet.

Obviously, the exact nature of the smallness of $\mathcal{M}$ relative to $\mathcal{O}$ and the respective classification times depend on non-intrinsic considerations. In general, we consider subsets below 20% of the ontology and speed-ups of at least an order of magnitude, and preferably more.

Given that exhaustive search is unpromising, indeed the search space is unmanageable; for a number of axioms n, variable k, and considering only subsets of size below 20% of n, the possible subsets are all unique combinations of n of size k, for $1 \leqslant k \leqslant 0.2n$, we need some other method for producing good "candidate hot spots", i.e., subsets that are likely to be hot spots. In [23], the authors suggest that the satisfiability checking (SAT) time of an atomic concept is an indicator of the total time the reasoner spends on or "around" those atomic concepts during classification. In particular, they observe that in their examined ontologies, relatively few concepts (2-10 out of 1000s) took enormously more time to check their satisfiability than for the rest of the concepts. Since subsumption testing is reduced to satisfiability checking, it is at least *prima facie* plausible that the stand alone satisfiability time is correlated with a "hot spot". Indeed, the authors were able to "repair" their sample ontologies, by removing a small number of axioms based on guidance from SAT times.

5.1 Hot Spot Detection

Just knowing the "hard" concepts does not give us a corresponding set of axioms. For a candidate C, we use the $\top\bot^*$-module of the terms co-occurring with C in an axiom in $\mathcal{O}$ as the module "around" C. This roughly approximates what an ideal user might do: identify the problem (C) and then "remove it" (i.e., remove its explicit and implicit presence; the usage gets the explicit while the module gets the rest; this is an approximation, obviously). We rely on $\top\bot^*$-modules as these were shown to be the smallest kind of locality-based module [17]. The full technique is described by Algorithm 1. To test whether our indicator is effective, we compare it to candidates generated from randomly selected concepts. For each member of our set of 12 "hard" BioPortal ontologies we attempted to find 3 witness hot spots while testing no more than 1,000 hot spot candidates. In each case, we selected candidate hot spots using both the SAT-guided and the randomly selected concept methods.

Algorithm 1 Identification of hot spots in ontologies.

```
Input: Ontology $\mathcal{O}$
Output: Set of modules $S$, wherein for each $\mathcal{M}_i \in S$: $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}_i) \ll \mathrm{RT}(\mathcal{O})$

$S \leftarrow \emptyset$;  $Candidates \leftarrow \emptyset$;  $Times \leftarrow \emptyset$;  $max = 1000$;
for all atomic concepts $C \in \widetilde{\mathcal{O}}$ do {S1: Get SAT times}
    $Times \leftarrow Times \cup \langle C, SATtime(C) \rangle$
end for
Sort $Times$ in descending order of $SATtime(C)$
$Candidates \leftarrow Candidates \cup \{C$ with highest $SATtime$ up to $max$ concepts$\}$
for all $C \in Candidates$ do {S2: Verify candidate hot spots}
    $\mathcal{M} = \top\bot^*\text{-}mod(\{t \mid t$ co-occurs with $C$ in some $\alpha \in O\}, \mathcal{O})$
    if $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}) \ll \mathrm{RT}(\mathcal{O})$ then {S3: Test hot spot effectiveness}
        $S \leftarrow S \cup \mathcal{M}$
    end if
end for
return $S$
```

The first striking result is that we verified all the coarse-grained heterogeneity predictions. That is, if an ontology had a linear performance growth curve then neither method found a hot spot, whereas if the growth curve was non-linear then we found at least 1 hot spot, and usually 3.[10]

The hot spots found are described in Table 1. Both techniques were able to find hot spots most of the time, though the random approach failed in two cases. For the NEMO/HermiT combination, both approaches failed to find 3 before the limit, which suggests that hot spots are scarce. Contrariwise, for NCIt/HermiT, while the random approach failed to find any hot spots, the SAT-guided approach found them in 7 tests. In general, though not always, the SAT-guided approach found 3 hot spots in far fewer tests than the random approach (on average, respectively, in 129 vs. 426 tests), validating concept satisfiability as a significant indicator. Note that, at this point, we only present classification time boosts. The completeness of classification results is presented in Table 5.

A difficulty of the SAT-guided approach is the time to test all concepts for satisfiability. For example, we were unable to retrieve precise satisfiability-checking times for the GO-Ext ontology with FaCT++ and JFact. Instead, we

[10] Of course, this could be just that we failed to find the telltale hot spots in the linear-growth ontologies. However, the overall evidence is highly suggestive.

Ontology	Nr. Axioms	Nr. Concepts	Reasoner	RT($\mathcal{O}$)	Hot Spots	Avg. RT($\mathcal{O} \setminus \mathcal{M}$)	Avg. Boost	Nr. Tests	Avg. $\lvert\mathcal{M}\rvert$	Avg. %$\lvert\mathcal{O}\rvert$	Avg. RT($\mathcal{M}$)
ChEBI	60,085	28,869	Pellet	65.8	3	12.3	82%	3	186	0.3%	0.55
					3	3.5	95%	89	522	1%	0.72
EFO	7,493	4,143	Pellet	61.1	3	9.6	81%	128	68	1%	0.13
					3	10.9	82%	863	70	1%	0.14
GO-Ext.	60,293	30,282	Pellet	268.4	3	29.6	89%	36	98	0.2%	0.08
					3	31.9	88%	419	17	0.03%	0.06
IMGT	1,112	112	Pellet	>54,000	1	26.1	99%	112	98	9%	0.09
					1	26.1	99%	112	98	9%	0.09
			HermiT	80.4	3	7.8	90%	86	35	3%	8.86
					3	7.1	91%	103	36	3%	10.4
NEMO	2,405	1,422	HermiT	76.3	1	5.5	93%	1,000	44	2%	4.63
					0	-	-	1,000	-	-	-
OBI	25,257	3,060	HermiT	61.6	3	2.3	96%	3	570	2%	1.56
					3	4.3	93%	189	576	2%	1.48
			JFact	72.1	3	1.1	93%	3	570	2%	1.12
					3	7.4	90%	57	576	2%	1.19
			Pellet	119.8	3	11.1	91%	29	708	3%	2.05
					3	21.6	82%	133	593	2%	1.76
VO	8,488	3,530	Pellet	4275.9	3	30.4	99%	11	322	4%	1.56
					3	371.7	91%	725	262	3%	0.61
NCIt	116,587	83,722	HermiT	430.1	3	16.1	88%	7	3,611	3%	16.14
					0	-	-	1,000	-	-	-

Table 1: Comparison of hot spots found via SAT-guided (white rows) and random (grey rows) concept selection approach. "Nr. Tests" is the number of candidates tested before either finding 3 hot spots or exhausting the set search space (either the number of concepts in the ontology or 1000, whichever is smaller). CPU times in seconds.

used a timeout on each concept satisfiability check of 60 seconds. Also note that, for this particular ontology, we use (in Table 1 and subsequent ones) the median time value from the wide range of obtained classification times.

Overall, the hot spot finding mechanism described in Algorithm 1 is feasible, and successfully identified hot spots in all ontologies deemed performance heterogenous. The run time of our implementation is lower than the original classification time in 4 out of 11 cases, including one case (IMGT/Pellet) for which classification did not terminate within 15 hours. In general, the found hot spots were quite good: they typically were smaller than our limit (only IMGT/Pellet was above 5% of the ontology) and often giving massive speedups (e.g., IMGT/Pellet). There is no indication that hot spots, on their own, are particularly hard, which suggests an interaction effect, as expected.

5.2 Hot Spot Analysis

In order to investigate whether the removal of each hot spot happened to shift expensive constructs from the main input to the subset, we verify the expressivity of the hot spots and the remainder ontology (shown in Table 2).

Notice that, in several cases, the removal of the hot spot does not change the expressivity of the remainder w.r.t. the whole ontology, e.g. in ChEBI. However

Ontology	$\mathcal{O}$	$\mathcal{O} \setminus \mathcal{M}$	$\mathcal{M}$
ChEBI	$\mathcal{ALE}+$	$\mathcal{ALE}+$	$\mathcal{ALE}+$
EFO	$\mathcal{SHOIF}$	$\mathcal{SHIF}$	$\mathcal{SHOIF}$
GO-Ext.	$\mathcal{ALEH}+$	$\mathcal{ALEH}+$	$\mathcal{AL}$, $\mathcal{ALEH}+$, $\mathcal{ALE}$
IMGT	$\mathcal{ALCIN}$	$\mathcal{ALC}$, $\mathcal{ALCIN}$	$\mathcal{ALCI}$, $\mathcal{ALCIN}$
NEMO	$\mathcal{SHIQ}$	$\mathcal{SHIF}$	$\mathcal{SHIQ}$
OBI	$\mathcal{SHOIN}$	$\mathcal{SHOIN}$	$\mathcal{SHOIF}$, $\mathcal{SHOIN}$
VO	$\mathcal{SHOIN}$	$\mathcal{SHOIN}$	$\mathcal{SHOIF}$
NCIt	$\mathcal{SH}$	$\mathcal{ALCH}$	$\mathcal{S}$

Table 2: Expressivity of each original ontology ($\mathcal{O}$), its various hot spots ($\mathcal{M}$) and corresponding remainders ($\mathcal{O} \setminus \mathcal{M}$).

in other, yet few cases, there is a reduction of expressivity, e.g., the hot spots found in EFO leave the remainder without nominals. Similarly in NEMO the remainder no longer has qualified cardinality restrictions.

In order to get a better understanding of why this performance boost occurs, particularly the interaction effect between each hot spot and the ontology, we verify whether the removal of the hot spots from these ontologies changes the number of General Concept Inclusions (GCIs),[11] as these are an obvious potential source of hardness. The results gathered are shown in Table 3.

Ontology	$\mathcal{O}$	$\mathcal{O} \setminus \mathcal{M}_1$	$\mathcal{O} \setminus \mathcal{M}_2$	$\mathcal{O} \setminus \mathcal{M}_3$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
EFO	172	163	164	164	9	8	8
GO-Ext	4407	4398	4382	4382	9	25	16
NCIt	42	37	36	36	5	6	6
NEMO	31	30	-	-	1	-	-
OBI	227	182	193	193	44	33	33
VO	235	196	201	197	39	34	38
IMGT	38	0	0	0	38	38	38

Table 3: Number of GCIs contained in the each ontology, its hot spots, and their corresponding remainders.

The obvious thing to notice here is that the removal of each of the 3 hot spots found within IMGT (for HermiT) leaves the remainder with no GCIs at all. Other cases are not so obvious, indeed in, e.g., NEMO or NCIt, only a few GCIs are removed from the original ontology. However, there seems to be some relation between the loss of GCIs from the original ontology into the hot spot, and the improvement in classification time. We speculate that a glass box approach to investigating this relation may help disentangle performance difficulties in specific reasoners, though this is beyond the scope of the paper.

[11] Axioms with complex concepts on both sides, e.g., $\exists r.A \sqsubseteq \exists r.B$.

5.3 Comparison with Pellint

As a final check, we compared our technique with Pellint [12]. Pellint is a "performance lint" dedicated specifically to the Pellet reasoner; it draws on the knowledge of the Pellet developers to generate a set of rules for what sorts of constructs and modelling patterns are likely to cause performance degradation when using Pellet — essentially it is a Pellet specific, ontology performance tuning expert system. Pellint not only identifies problem constructs, but it suggests approximations (typically by weakening or rewriting axioms) which "should" improve performance. If the number of axioms touched by Pellint repairs is sufficiently small and the gain sufficiently large, then Pellint will have identified a hot spot (though, at most 1). Since we believe that the "predicted homogeneous" ontologies have no hot spots (and we did not find any), we would expect that, while perhaps improving their performance, Pellint would not identify a hot spot. Similarly, for non-Pellet reasoners, we would expect no improvements at all. To check these conjectures we ran Pellint on all our ontologies and compared reasoning times for all "bad" reasoner/ontology combinations for both the Pellint approximated versions, and by removing the modified axioms (thus providing a direct comparison with Table 1). The results are shown in Table 4. Note that ontologies for which Pellint found no lints at all are omitted (5, in total). If Pellint found lints but could not alter them, then the number of altered axioms will read as 0 and no tests performed.

Ontology	Reasoner	RT($\mathcal{O}$)	Nr. Axioms Altered (lints)	%$\|\mathcal{O}\|$ Altered	Altered($\mathcal{O}$)		$\mathcal{O} \setminus \{\text{lints}\}$	
					RT($\mathcal{O}$)	Boost	RT($\mathcal{O}$)	Boost
ChEBI	Pellet	65.8	0	-	-	-	-	-
EFO	Pellet	61.1	172	2%	3.7	94%	3.1	95%
GO-Ext.	Pellet	268.4	4407	7%	19.4	93%	5.85	98%
VO	Pellet	4275.9	231	3%	119.7	97%	3.32	99%
NCIt	HermiT	430.1	42	0.04%	443.4	-3%	448.1	-4%
Coriell	Pellet	923.5	46	0.03%	642.3	30%	631.0	32%
	FaCT++	156.1			159.2	-2%	159.1	-2%
	JFact	154.8			154.2	0.4%	143.9	7%
PRPPO	Pellet	118.9	0	-	-	-	-	-

Table 4: Ontologies for which Pellint found "lints".

First, Pellint was not able to find any hot spots in the predicted homogeneous ontologies, though for one (Coriell/Pellet) it was able to provide a significant performance boost (32%). This further confirms our linear/homogeneous hypothesis. Second, Pellint found hot spots in 3 out of 8 heterogeneous ontologies, performing much worse than even random concept selection. When found, the hot spots where competitive, but not all repaired lints improved performance (i.e., NCIt/HermiT). Pellint failed to find hot spots in our experiments due to finding no lints (5 ontologies), having no repairs[12] (2 ontologies), or just failing

[12] The set of suspect axioms might be a hot spot (or a subset thereof), but without access to them we cannot test.

to produce a dramatic enough (or any) effect (4 ontology/reasoner pairs, with most being non-Pellet). As expected, Pellint found no hot spots or performance improvements for other reasoners. Of course, this might be just be due to its overall poor hot spot finding.

Finally, Pellint's alterations had a noticeable negative effect on reasoning time compared to simple removal. Whether these approximations significantly save entailments needs to be investigated. Given the high development and maintenance costs of Pellint, it does not seem viable compared to search based methods.

6 Improving Classification via Hot Spots

The applicability of our hot spot finding method is dependent on how much information users are willing to lose. In a realistic edit-compile-deploy scenario, users may be wary to dispose of parts of their ontology. Thus, in order to avoid this predicament, we explore a series of approximation and knowledge compilation techniques, and compare them with a known approximate reasoning method. The latter is based on a reduction of the input into the tractable fragment of OWL: $\mathcal{EL}$, as implemented in TrOWL [15]. We implemented the $\mathcal{EL}$ reduction algorithm so as to apply it to any given reasoner other than REL (the reasoner used within TrOWL). Our approximation-based classifier is denoted *ELC*.

6.1 Approximate Reasoning

First off, given a hot spot and an ontology, we have an immediate approximation $\mathcal{O} \setminus \mathcal{M}$ of $\mathcal{O}$; It is much easier to reason over than the original ontology, though possibly too incomplete w.r.t. $Cl(\mathcal{O})$ (i.e., the set of inferred atomic subsumptions of $\mathcal{O}$). From hereon we derived two more approximations: 1) $Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$, which would naturally be more complete than $\mathcal{O} \setminus \mathcal{M}$ alone, and 2) $\mathcal{O} \setminus \mathcal{M} \cup Cl(\mathcal{M})$, where we expect that the interaction between inferred subsumptions in $\mathcal{M}$ and the remainder will bring us closer to $Cl(\mathcal{O})$. A comparison of these techniques is shown in Table 5, containing the results of each of the 3 approximations as well as *ELC* with the respective reasoner.

Overall the closest approximation is $\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$, which yields an average completeness of 99.84% and an average boost of 89.3% over the original times. *ELC* is typically more complete, though in several cases classifying an ontology with *ELC* is much slower than the original RT($\mathcal{O}$), e.g., *ELC* failed to classify the NCIt within 5 hours, compared to ≈7 minutes originally. Similarly with ChEBI and OBI, the approximation is no faster than the original times. Overall the average boost via *ELC* is non-existent, particularly due to the NCIt case. By excluding that one case, *ELC*'s average classification time boost is of 33.7%.

Applying the original TrOWL system, with its internal reasoner REL, is not so much better than using standard DL reasoners on the $\mathcal{EL}$ approximations, particularly since some DL reasoners (e.g., Pellet or FaCT++) are finely tuned to the $\mathcal{EL}$ fragment of OWL. Nevertheless, we analysed those results only to find

Ontology	Reasoner	$\mathcal{O} \setminus \mathcal{M}$		$Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$		$\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$		ELC	
		Compl.	Boost	Compl.	Boost	Compl.	Boost	Compl.	Boost
ChEBI	Pellet	55%	89%	55%	89%	100%	84%	100%	-207%
EFO	Pellet	78%	86%	79%	86%	100%	81%	100%	63%
NCIt	HermiT	75%	90%	80%	87%	100%	89%	-[12]	-2651%
NEMO	HermiT	97%	96%	98%	92%	100%	96%	99.94%	93%
OBI	HermiT	51%	96%	55%	94%	100%	94%	100%	14%
	JFact	51%	91%	55%	90%	99.92%	84%	99.95%	-10%
	Pellet	50%	88%	54%	88%	100%	86%	100%	54%
IMGT	Pellet	68%	100%	76%	100%	100%	100%	100%	100%
	HermiT	92%	92%	97%	78%	99.92%	92%	100%	100%
VO	Pellet	50%	98%	52%	98%	98.36%	94%	100%	97%
GO-Ext	Pellet	95%	90%	96%	90%	100%	81%	100%	33%
Average		69.2%	92.4%	72.3%	90.4%	99.84%	89.3%	99.99%	-210%

Table 5: Approximate reasoning results for the approximations $\mathcal{O} \setminus \mathcal{M}$, $Cl(\mathcal{O} \setminus \mathcal{M}) \cup Cl(\mathcal{M})$, $\mathcal{O} \setminus \mathcal{M} \cup Cl(M)$, and, finally, ELC. The completeness of each approach w.r.t. $Cl(\mathcal{O})$ is denoted "Compl.".

that TrOWL has the exact same problem with the NCIt, and out-performs ELC in 4 out of 7 cases by mere seconds.

6.2 Knowledge Compilation

While the loss of entailments via our best approximation is typically empty, or very low, we investigate whether a number of knowledge compilation techniques based on hot spots enjoy the same performance boosts as the approximations in Section 6.1. These techniques all maintain 100% completeness of knowledge contained in the original ontologies, i.e., they produce logically equivalent knowledge bases. The rationale behind these techniques is that by adding inferred knowledge (e.g., from a hot spot) to the original ontology, reasoners will not need to do certain (possibly expensive) subsumption tests, and, as a consequence, should (at least intuitively) perform faster. The results are shown in Table 6.

First thing to notice here is that adding the inferred class hierarchy of the parts does not necessarily improve classification time over the whole. There are cases, such as OBI with JFact, where all compilation techniques took much longer to classify than the original (note that we timed-out the operation at 5 hours). On the other hand, there are cases where there is mild to noteworthy improvement, for instance VO classifies 75% faster when we use the second compilation technique, which is a significant improvement with no loss of knowledge. Similarly the GO-Ext ontology classifies 92% faster with both the second and third compilation technique. Nevertheless, the results gathered are not nearly as stable w.r.t. classification time improvement as our approximations, and the improvements obtained are also not as high as those shown in Section 6.1.

[12] The classification of the NCIt was interrupted after running for 5 hours, well above the original classification time.

Ontology	Reasoner	$\mathcal{O} \cup Cl(\mathcal{M})$		$\mathcal{O} \cup Cl(\mathcal{O} \setminus \mathcal{M})$		$\mathcal{O} \cup Cl(\mathcal{M}) \cup Cl(\mathcal{O} \setminus \mathcal{M})$	
		Time	Boost	Time	Boost	Time	Boost
ChEBI	Pellet	74.5	18%	73.1	19%	73.6	19%
EFO	Pellet	51.3	30%	63	14%	62.9	14%
NCIt	HermiT	616.1	6%	603.2	8%	614.5	6%
NEMO	HermiT	94.9	5%	94.6	6%	98.6	2%
OBI	HermiT	71	-3%	69.1	0%	70.7	-2%
	JFact	>5hrs	-	>5hrs	-	>5hrs	-
	Pellet	264	-66%	207.5	-31%	276.6	-74%
IMGT	Pellet	36000	33%	36000	33%	36000	33%
	HermiT	94.8	-4%	94.9	-4%	94.8	-4%
VO	Pellet	1704.4	60%	1066.2	75%	2136.2	50%
GO-Ext	Pellet	161.4	56%	30.1	92%	30.6	92%
Average Boost		-	14%	-	21%	-	14%

Table 6: Compilation results for the techniques $\mathcal{O} \cup Cl(\mathcal{M})$, $\mathcal{O} \cup Cl(\mathcal{O} \setminus \mathcal{M})$ and $\mathcal{O} \cup Cl(\mathcal{M}) \cup Cl(\mathcal{O} \setminus \mathcal{M})$.

7 Related Work

In [23], a number of ontology profiling techniques are proposed and realized in a tool, Tweezers for Pellet. Tweezers allows users to investigate performance statistics, such as the satisfiability checking time for each concept in the ontology, but relies on the user to apply this information. Our goal driven technique can be seen as the automated exploitation of their statistics.

In [4] the author proposes three techniques to automatically identify potentially expensive "constructs" (concepts, roles or axioms). These techniques search for "suspect" constructs by recursively splitting an ontology in different manners, and individually testing performance over the parts until suspects are found. While their actual attempt was rather *ad hoc*, it does suggest an alternative discovery mechanism (as they did find some hot spots).

In [3] the authors present a form of OWL reasoner benchmarking based on justifications. JustBench computes all justifications for entailments in a given ontology, and measures the performance of reasoners on those justifications. The authors hoped that they would find justifications that were hot spots themselves (or indicators thereof), but this hope was not borne out by their experiments.

8 Discussion and Applications

Unlike with hot spots in programs, there is no straightforward relationship between the performance of a "hot spot" in isolation and the effect it has on the ontology as a whole (see the last column in Table 1). Our results have shown that there is no precise co-relation between the classification time of a hot spot alone, and the reduction in classification time when such hot spot is removed. This is somewhat similar to the fact that in a program, if an individually quick function is called sufficiently often, it may be the performance bottleneck for that program. That is, looking at the performance of the function in isolation is not sufficient to determine its effect on the overall runtime. However, in our case, there are many possible and currently unknown ways that a performance hot spot might affect overall runtime, and yet not exhibit pathological behaviour on its own. Indeed, the fact that sometimes adding axioms is sufficient to eliminate performance problems shows that isolating behaviour is not a reliable predictor of integrated effect. It would be interesting to seek out inverse hot spots, that is, acceptably small subsets whose removal greatly *increases* the classification time of an ontology, though these would have less end user applicability. Of course, merely finding hot spots does not provide any explanation of performance patterns, it merely provides tools for investigating them. On the other hand, it is a purely black box technique, thus, unlike Pellint, does not require such insight to be effective.

Our investigation was partly inspired by our observation of user coping techniques for recalcitrant ontologies, thus it is natural to seek to apply them in such scenarios. The basic idea is straightforward enough: Present the user with a selection of hot spots and let them select the most appropriate one to "set aside" (permanently or temporarily) or to rewrite into a less damaging approximation. Of course, we might want hot spots with somewhat different properties, e.g., that the remainder ontology is a module rather than the hot spot, so that "safe edits" to the remainder will not alter the meaning of the hot spot. We might use heuristics to select a hot spot for automated removal or approximation. Modular hot spots might be presented to the user so they can attempt to have a clearer understanding of "what was removed."

Our techniques could benefit reasoner developers as well. For example, a hot spot gives the developer a pair of almost identical ontologies with vastly different performance behaviour. By comparing the profiling reports on their reasoners processing these inputs, the developer might gain additional insight.

Currently, we have concentrated on satisfiability-checking time of atomic concepts as the indicator for hot spots. There are clearly alternatives for this, e.g., small atoms [6] or justifications, as well as brute force methods [4].

All our experiments, as they stand, can be improved in two dimensions: 1) more input ontologies are always better, and 2) our sampling, particularly in the coarse grained method, is very low. Clearly, they were sufficient to reveal some interesting phenomena, but not to establish statistically significant findings.

Finally, it may be possible to derive Pellint-like rules directly from hot spots extracted from a large number of ontologies. While requiring maintenance, it would be inherently much faster than our approaches as it would not require any reasoning at all.

References

1. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL — A polynomial-time reasoner for life science ontologies. In: IJCAR-06 (2006)
2. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In: Proc. of DL 2006 (2006)

3. Bail, S., Parsia, B., Sattler, U.: JustBench: A framework for OWL benchmarking. In: Proc. of ISWC-10 (2010)
4. Charaniya, S.: Facilitating DL Reasoners Through Ontology Partitioning. Master's thesis, Nagpur University, India (2006)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research 31 (2008)
6. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI-11 (2011)
7. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proc. of OWLED-09 (2009)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR-06 (2006)
9. Horrocks, I.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
10. Horrocks, I., Patel-Schneider, P.F.: Evaluating optimised decision procedures for propositional modal k(m) satisfiability. J. of Automated Reasoning 28, 173–204 (2002)
11. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Proc. of ISWC/ASWC-07 (2007)
12. Lin, H., Sirin, E.: Pellint - a performance lint tool for Pellet. In: Proc. of OWLED-08EU (2008)
13. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., Musen, M.A.: Bioportal: Ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37, W170 – W173 (2009)
14. Patel-Schneider, P.F., Sebastiani, R.: A new general method to generate random modal formulae for testing decision procedures. J. of Artificial Intelligence Research 18, 351–389 (2003)
15. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness Preserving Approximation for TBox Reasoning. In: Proc. of AAAI-10 (2010)
16. Rudolph, S., Tserendorj, T., Hitzler, P.: What is approximate reasoning? In: Proc. of RR-08 (2008)
17. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009 (2009)
18. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artificial Intelligence 74, 249–310 (1995)
19. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Proc. of OWLED-08EU (2008)
20. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2) (2007)
21. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of IJCAR-06 (2006)
22. W3C OWL Working Group: OWL 2 Web Ontology Language: Document overview. W3C Recommendation (27 Oct 2009), `http://www.w3.org/TR/owl2-syntax/`
23. Wang, T.D., Parsia, B.: Ontology performance profiling and model examination: First steps. In: Proc. of ISWC/ASWC-07 (2007)