R_2O_2: an Efficient Ranking-based Reasoner for OWL Ontologies

Yong-Bin Kang[†], Shonali Krishnaswamy[§], Yuan-Fang Li[†]

[†]Faculty of IT, Monash University, Australia
[§]Institute for Infocomm Research, A*STAR, Singapore
[†]{yongbin.kang,yuanfang.li}@monash.edu,
[§]spkrishna@i2r.a-star.edu.sg

Abstract. It has been shown, both theoretically and empirically, that performing core reasoning tasks on large and expressive ontologies in OWL 1 and OWL 2 is time-consuming and resource-intensive. Moreover, due to the different reasoning algorithms and optimisation techniques employed, each reasoner may be efficient for ontologies with different characteristics. In this paper, we present R_2O_2, a *meta-reasoner* that automatically combines, ranks and selects from a number of state-of-the-art OWL 2 DL reasoners to achieve high efficiency, making use of performance prediction models and ranking models. Our comprehensive evaluation on a large ontology corpus shows that R_2O_2 significantly and consistently outperforms 6 state-of-the-art OWL 2 DL reasoners on average performance, with an average speedup of up to 14x. R_2O_2 also shows a 1.4x speedup over Konclude, the current dominant OWL 2 DL reasoner.

1 Introduction

Core reasoning services such as consistency checking and classification are at the heart of ontology-based applications. For expressive description logics (DLs), such reasoning services have a very high worst-case complexity. For instance, satisfiability checking for $\mathcal{SROIQ}$, the description logic underpinning OWL 2 DL, has worst-case complexity of 2NExpTime-complete [4]. Recent work has also demonstrated empirically [5,11,15] that large and complex ontologies indeed pose a real computational challenge even for state-of-the-art reasoners.

In the past decade, highly optimised ontology reasoners such as FaCT++ [25], HermiT [8] and Konclude [22] have been developed that are capable of reasoning about highly expressive DLs. They implement different reasoning algorithms and employ different sets of preprocessing and optimisation techniques. As a result, they are optimised for certain, but not all ontologies. Dramatic differences in reasoning time among reasoners, sometimes by up to four orders of magnitude, have been observed for some ontologies [5]. Such disparities can cause significant and unnecessary loss in productivity for developers and users of ontologies.

The *robustness* of ontology reasoners was recently investigated [10], with a particular focus on reasoning efficiency. It was observed that given a corpus of ontologies and a number of state-of-the-art reasoners, it is highly likely that one of the reasoners performs sufficiently well on any given ontology in the corpus. However, this *virtual best*

reasoner is only found *a posteriori*, and the paper did not discuss how the best reasoner may be selected automatically. It only stated that this task is not straightforward.

The prediction of ontology reasoning performance was recently studied [15,17], where a prediction model, either a classifier or a regression model, is trained for a given reasoner to make predictions on reasoning time (discretised or actual) of a given ontology. High prediction accuracy is achieved for some state-of-the-art reasoners. These prediction models enable efficient and accurate estimation of a reasoner's performance on an ontology. However, it was not discussed how these models can be used to improve reasoning efficiency.

Portfolio-based algorithm selection methods [13] have been successfully applied to SAT and constraint satisfaction problems. The portfolio SAT solver SATvilla has consistently outperformed single SAT solvers in many SAT competitions [30]. Compared to SAT, ontology languages are more expressive with the inclusion of many more language constructs. As a result, it is more challenging to accurately characterising ontology complexity. Moreover, the performance of a portfolio-based algorithm heavily depends on the accuracy of performance prediction models, and this dependency makes it difficult to further improve its efficiency.

Recently we conducted a preliminary study [14] on constructing a portfolio-based OWL reasoner from six reasoners: FaCT++, HermiT, JFact, MORe, Pellet and TrOWL. This preliminary study made use of classifiers that predict discretised reasoning time. Evaluation shows that, for average performance, it outperforms all of the component reasoners. However, due to increasing sizes of the bins (discretised reasoning time), a best reasoner may not be identified. Moreover, Konclude, a dominant OWL 2 DL reasoner was not included in the study, making its results less significant.

The above work motivates and enables us to propose R_2O_2, a *meta-reasoner* that combines reasoners with their respective prediction models, and aims at determining the most efficient reasoner for a given ontology. It achieves this by (1) training *prediction models* for predicting actual reasoning time for all reasoners, (2) learning *ranking models* (simply *rankers*) that automatically and efficiently rank the reasoners according to their predicted reasoning performance, and (3) selecting a possible best reasoner given the outputs of the rankers.

Our main contribution is the proposal of a novel meta-reasoner, R_2O_2, that automatically and efficiently combines, ranks and selects OWL reasoners with the aim of determining the most efficient reasoner for a given ontology. We conducted a comprehensive evaluation on more than 2,000 ontologies and six state-of-the-art OWL 2 DL reasoners, including Konclude, the current dominant reasoner. The evaluation shows that, for average performance over a large ontology corpus, R_2O_2 significantly and consistently outperforms all six reasoners, achieving an average speedup of up to 14x, and a 1.4x speedup over Konclude. R_2O_2 also outperforms the traditional portfolio-based approach (that does not perform ranking) with a 1.5x speedup.

2 The Meta-reasoner R_2O_2

R_2O_2 is a supervised meta-reasoner that utilises the state-of-the-art performance prediction models of different reasoners and ranking models (or rankers). R_2O_2 encom-

passes a number of component reasoners and operates in two phases: offline training and online reasoning. During the training phase, R_2O_2 trains rankers that rank component reasoners on their predicted reasoning time data generated by performance prediction models for these reasoners. After training is completed, R_2O_2 makes predictions of the most efficient reasoners for unseen ontologies, and carries out actual reasoning.

More specifically, the training phase of R_2O_2 is divided into two steps:

1. Given a set of training ontologies, each represented by values of *ontology metrics*, and actual reasoning time for a set of reasoners on each ontology, the performance prediction model of each reasoner is built, following the methodology in [17]. That is, we build a Random Forest-based regression model for each reasoner with the metrics as features (see Section 2.2).
2. Given another set of training ontologies (distinct from the ones used in the first step), we generate a *ranking matrix* where each row represents the values of ontology metrics and a ranking of the reasoners where the ranking is made according to their *predicted* reasoning time. Rankers are then trained on this ranking matrix to learn how the characteristics of an ontology represented by its metrics can be optimally mapped to relative ordering of the predicted performance of reasoners (see Section 2.3).

In the reasoning phase, given an unknown ontology, R_2O_2 makes performance predictions for all the component reasoners. R_2O_2 then ranks the reasoners according to their predicted reasoning time. The rankings recommended by the trained rankers are averaged across all the rankers to determine a unique rank for each reasoner. The highest ranked reasoner will be eventually chosen to perform the reasoning task for the ontology (see Section 2.4).

In the reasoning phase, in our context, R_2O_2's main difference from the traditional portfolio-based approach (denoted PR) in the spirit of SATzilla [31] is that PR always selects the most efficient reasoner for any given ontology according to predicted reasoning time of all component reasoners. That is, given a new ontology, PR computes its ontology metrics, estimates the predicted reasoning time of each component reasoner using the corresponding prediction model, and recommends the reasoner predicted to be the fastest. In contrast, R_2O_2 uses a best possible reasoner by recommending the top ranked reasoner from an aggregation of the rankings of component reasoners, according to their predicted reasoning time, estimated by the trained rankers. Our evaluation shows that R_2O_2 highly and consistently outperforms PR.

Learning rankers from preferences has recently received much attention in the machine learning community [6]. Contrary to the *classification* problems, in the ranking matrix, a training example (i.e. an ontology) is not assigned a single label, but a set of preferences of multiple labels representing reasoners, where one is preferred over another according to their predicted reasoning performance (i.e., the more efficient a reasoner is, the higher its rank is). Once a ranker is learned, our goal is to use it in predicting the most likely relative ordering (i.e., ranking) of all reasoners under consideration for unknown ontologies represented by their ontology metrics.

In our approach, the learning of our rankers is based on *preference learning* [6], which is concerned with the acquisition of preference models from data. In general, the goal of preference learning is to learn preference orders (i.e. rankings) of all possible

labels (i.e. performance prediction models) from a training example (i.e. ranking matrix) and predict an ordering (i.e. ranking) to an unseen instance (i.e. ontology).

In the rest of this section, we describe how R_2O_2 is built in more detail.

2.1 Notation Definition

The following notations will be used in the paper:

- Let $R = \{r_1, \ldots, r_n\}$ be a set of n reasoners.
- Let $\hat{R} = \{\hat{r}_1, \ldots, \hat{r}_n\}$ be a set of n performance prediction models such that for each reasoner $r_i \in R$, $\hat{r}_i \in \hat{R}$ predicts the actual reasoning time of r_i on a given ontology, i.e., $\hat{r}_i$ estimates the performance of r_i.
- Let $RM = \{rm_1, \ldots, rm_m\}$ be a set of m rankers. Each ranker produces a ranking of the reasoners based on their predicted reasoning performance for a given ontology. Specifically, a ranker rm is a function that, given an ontology o and a set of reasoners R, $rm(o, R)$ produces an ordering, a *permutation*, of R.
- Let $OM = \{om_1, \ldots, om_q\}$ be a set of q ontology metrics.
- Given a reasoner r (resp. a performance prediction model $\hat{r}$) and on ontology o, let $RT(r, o)$ (resp. $RT(\hat{r}, o)$) represent the actual (resp. predicted) reasoning time of r on o.
- Let $O \subseteq \mathcal{O} = \{o_1, \ldots, o_p\}$ be a set of p ontologies that can be reasoned about by at least one reasoner in R, i.e., for each $o \in O$, there is at least one reasoner in R that can successfully complete the reasoning task (e.g., classification) without any errors and within the specified time limit. Note that O includes those ontologies that timeout for *some* reasoners. Let $O_c \subseteq O$ be the set of common ontologies that can be reasoned about by all reasoners in R (no errors and no timeout). Note that O_c also includes those ontologies that timeout for some reasoners. Two disjoint subsets O_r and O_t are drawn from O_c, for training the rankers and testing R_2O_2, respectively. Furthermore, for each reasoner r_i, let set $O_{p_i} \subseteq O \setminus (O_r \cup O_t)$ represent a separate subset of ontologies that r_i can successfully reason about, without timing out. O_{p_i} is used for training the performance prediction model $\hat{r_i}$ for reasoner r_i.

2.2 Building Performance Prediction Models

For each reasoner $r_i \in R$, we train a Random Forest-based prediction model, a *regression model*, $\hat{r}_i \in \hat{R}$ on the training data O_{p_i} with the aim of estimating the *actual*, but not discretised, reasoning time of r_i. Ontology metrics [17] are collected for ontologies in O and used as features to train prediction models in $\hat{R}$.

The produced prediction models in $\hat{R}$ will be used for generating a *ranking matrix* to train the rankers in RM. R^2 (i.e. the coefficient of determination) is widely used to assess the quality of regression models. In our context, R^2 indicates how well each prediction model $\hat{r}$ approximates the actual reasoning time of the corresponding reasoner r. It is possible that two different reasoners are predicted to have the same reasoning time on a given ontology. R^2 is used for *tie-breaking* purposes in R_2O_2, which will be explained in Section 2.4.

2.3 Generating the Ranking Matrix and Training Rankers

Once the performance prediction models in $\hat{R}$ are built, a *ranking matrix* is constructed for training the rankers in RM.

Recall that $O_r \subseteq O_c$ is the set of common ontologies for training rankers. Initially, we build a $|O_r| \times (q+n)$ data matrix $\mathbf{M_d}$ (recall that $|OM| = q$, $|R| = |\hat{R}| = n$), where row i represents $o_i \in O_r$ and is constructed as:

$$(\underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}), (\underbrace{RT(\hat{r}_1, o_i), \ldots, RT(\hat{r}_n, o_i)}_{\text{predicted reasoning time}}) \tag{1}$$

where $om_{i,j}$ is the value of the j-th ontology metric om_j of ontology o_i, and $RT(\hat{r}_s, o_i)$ denotes the reasoning time predicted by the prediction model $\hat{r}_s$ for ontology o_i.

Based on the data matrix $\mathbf{M_d}$, we build the corresponding $|O_r| \times (q+n)$ ranking matrix $\mathbf{M_r}$, where row i is represented as:

$$(\underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}), (\underbrace{\pi(\hat{r}_1, o_i), \ldots, \pi(\hat{r}_n, o_i)}_{\text{ranking of prediction models}}) \tag{2}$$

where $\pi(\hat{r}_s, o_i)$ denotes the *rank* of the prediction model $\hat{r}_s$ (hence the corresponding reasoner r_s) on ontology o_i, determined by $RT(\hat{r}_s, o_i)$. In the ranking matrix $\mathbf{M_r}$, the more efficient a performance prediction model is, the higher ranked it is (the smaller the ranking number is).

For example, suppose that there are 3 reasoners $\{r_1, r_2, r_3\}$, and thus 3 performance prediction models $\{\hat{r}_1, \hat{r}_2, \hat{r}_3\}$. Given an ontology o_i, suppose that the predicted reasoning time for o_i estimated by the models is 100s, 90s, and 10s respectively, i.e., $(RT(\hat{r}_1, o_i), RT(\hat{r}_2, o_i), RT(\hat{r}_3, o_i)) = (100s, 90s, 10s)$. Thus, the ranking of the prediction models is $(\pi(\hat{r}_1, o_i), \pi(\hat{r}_2, o_i), \pi(\hat{r}_3, o_i)) = (3, 2, 1)$. If the estimated reasoning time is $(10s, 10s, 100s)$ instead, the ranking produced will be $(1, 1, 3)$.

To recommend the most likely best reasoner, we note that our goal is not to predict the absolute expected reasoning time of any component reasoner, but rather the relative performance of the reasoners. Therefore, we generate a ranking matrix using ontology metrics and the rankings of the reasoners on the training data O_r to train rankers.

Once the ranking matrix $\mathbf{M_r}$ is generated, each ranker $rm_i \in RM$ is trained on $\mathbf{M_r}$. In our context, the problem of learning a ranker is to induce a ranking function f that can order n performance prediction models $\hat{r}_1, \ldots, \hat{r}_n \in \hat{R}$. That is, $rm_i \in RM$ takes as input an ontology and a set of reasoners R, and produces as output a permutation π of R. The interpretation of this permutation is that $\hat{r}_i$ is preferred to (more efficient than) $\hat{r}_j$ whenever $RT(\hat{r}_i, o) < RT(\hat{r}_j, o)$ for a given o. These function $rm_i \in RM$ are then used to estimate the rankings of the performance prediction models for unknown ontologies.

Different ranking models that use different ranking measure to evaluate the performance of the learned rankers [6]. Thus, the maximisation of the ranking measure will lead to the maximisation of a ranker's performance. Normalised Discounted

Cumulative Gain (NDCG) and Mean Average Precision (MAP) have often been used to measure ranking performance [3].

Five ranking models are included: *k-NN* (the nearest neighbor-based approach), *RPC* (the pairwise binary classification approach), *BinaryART* (the ranking tree-based approach), *ARTForests* (the ranking forest-based approach), and *RegRanker* (the regression-based approach). Readers interested in the details of the rankers are referred to [23].

In this work, we apply *rank average*, a widely-used rank aggregation method from a number of state-of-the-art rankers to induce the final ranking function f. Experimentally we also observe that aggregation of such rankings usually leads to better and more stable ranking performance.

2.4 Invoking the Meta-reasoner $\mathbf{R_2O_2}$

Once the rankers in RM are trained, for an unknown ontology, $\mathrm{R_2O_2}$ combines the rankings estimated by different trained rankers in RM to produce a final ranking, and selects the best reasoner based on it, as given in Algorithm 1. A detailed description is provided below.

Algorithm 1: Predict the most efficient reasoner in $\mathrm{R_2O_2}$.

Input: o_t a test ontology, $RM = \{rm_1, \ldots, rm_m\}$ the learned rankers, and $\Omega = \{R^2(\hat{r}_1), \ldots, R^2(\hat{r}_n)\}$ the R^2 values of prediction models in $\hat{R}$
Output: The most efficient reasoner r_{best} for ontology o_t

1 $om_t \leftarrow generateOntologyMetrics(o_t)$
2 $ranking \leftarrow (0, \ldots, 0)$
3 **foreach** $rm_i \in RM$ **do**
4 $\quad ranking_i \leftarrow recommendRanking(rm_i, om_t)$
5 $\quad ranking \leftarrow mergeRanking(ranking, ranking_i)$
6 **foreach** $\hat{r}_j \in \hat{R}$ **do**
7 $\quad \pi(\hat{r}_j, o_t) \leftarrow averageRanking(ranking, |\mathrm{RM}|, \hat{r}_j)$
8 $r_{bestCandidate} \leftarrow \arg\min_{\hat{r}_j \in \hat{R}} \pi(\hat{r}_j, o_t)$
9 **if** $(|r_{bestCandidate}| \geq 2)$ **then**
10 $\quad r_{best} \leftarrow tieBreaking(r_{bestCandidate}, \Omega)$
11 **else**
12 $\quad r_{best} \leftarrow r_{bestCandidate}$
13 **return** r_{best}

1. Given an unknown ontology o_t, $\mathrm{R_2O_2}$ first calculates its values of ontology metrics in OM (line 1).
2. Initialise a variable *ranking* as a sequence of 0's, where the length of *ranking* is $|\hat{R}| = n$. Intuitively, *ranking* keeps a merged ranking list of n prediction models produced by the trained rankers in RM. For instance, for $n = 3$ prediction models

(reasoners) $\hat{R} = \{\hat{r}_1, \hat{r}_2, \hat{r}_3\}$ with $RT(\hat{r}_1) < RT(\hat{r}_2) < RT(\hat{r}_3)$, *ranking* stores their current ranking, and is a permutation of $(1, 2, 3)$ (line 2).

3. For each ranker $rm_i \in RM$, $\mathrm{R_2O_2}$ finds and merges the ranking of n prediction models. The merge operation is implemented as a pointwise summation of rankings produced by all rankers. For example, if $n = 3$ and $ranking = (1, 1, 3)$ and a new ranking $(1, 2, 3)$ is produced by a ranker, then *ranking* is merged as $(1, 1, 3) + (1, 2, 3) = (2, 3, 6)$ (lines 3-5).
4. For each prediction model $\hat{r}_j \in \hat{R}$, $\mathrm{R_2O_2}$ computes its average ranking from the variable *ranking* over $|RM|$ (lines 6-7). Then, $\mathrm{R_2O_2}$ selects the prediction model(s) ($r_{bestCandidate}$) whose rank is the minimum. If only one top-ranked prediction model $\hat{r}_k \in \hat{R}$ is selected, the corresponding reasoner $r_k \in R$ is chosen to perform the reasoning task for o_t (line 12).
   However, if two or more prediction models are selected, a tie-breaking method is applied to select one of them (line 10). This method takes into consideration the R^2 values of the prediction models that are described from Section 2.2. Our tie-breaking finds the best possible prediction model $\hat{r}_k$ by identifying the prediction model that is the most accurate:

$$\hat{r}_k = \arg\max_{\hat{r}_i \in \hat{R}} R^2(\hat{r}_i), \tag{3}$$

   where we choose $\hat{r}_i$ that maximises its R^2 value $R^2(\hat{r}_i)$ (the higher the better). Finally, $\mathrm{R_2O_2}$ determines $r_k \in R$ as $\hat{r}_{best}$, and invokes it to perform reasoning for the ontology o_t.

3 Evaluation

3.1 Data Collection

For this work, we collected ontologies from the ORE 2014 reasoner competition [1], comprising a total of 16,555 ontologies.[1] In our evaluation, we randomly choose 25% of the ORE 2014 dataset by splitting it into four groups by percentiles of file size; and randomly sampling from within these groups. This is to ensure that files of different sizes are sufficiently represented. As a result, 4,138 ontologies were eventually used in our evaluation.

Six state-of-the art OWL 2 DL reasoners that participated in ORE 2014 reasoner competition are used as component reasoners for $\mathrm{R_2O_2}$: FaCT++ [25], HermiT [8], JFact,[2] Konclude [22], MORe [20] (with HermiT as the underlying OWL 2 DL reasoner), and TrOWL [24].[3] The versions of the reasoners are the same as those in ORE 2014. The competition framework is adapted to invoke the reasoners and to record their runtime.

[1] `http://www.easychair.org/smart-program/VSL2014/ORE-index.html`

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

[3] The Chainsaw reasoner [26] (an OWL 2 DL reasoner that participated in ORE 2014) is excluded due to reasoning errors in an excessive number of ontologies.

The reasoning time (for consistency checking and classification) of each reasoner is measured for each ontology in the dataset on a high-performance server running OS Linux 2.6.18 and Java 1.6 on two dual-core AMD Opteron 2218 processors each at 2.6GHz, with a maximum of 10GB memory allocated to the reasoner. A timeout of one hour wall-time is imposed on each (reasoner, ontology) pair.

Of the 4,138 ontologies, 2,847 ontologies (which we denote by O in Section 2.1) are successfully reasoned by at least one reasoner (without errors and within the one-hour time limit). In O, 2,407 ontologies are successfully reasoned by all the six component reasoners, while the others encountered processing errors by at least one reasoner. These 2,407 common ontologies constitute the dataset O_c.

A 10-fold cross validation is employed to adequately assess the performance of R_2O_2. In each fold, O is split according to Section 2.1. In the experiment, each of O_{p_i}, O_r and O_t is approximately 40%, 50% and 10% of the size of O respectively. The performance evaluation results presented in the rest of this section is the average across the 10 folds.[4]

Note that TrOWL is an approximate reasoner, hence it is sound but incomplete. All the other five reasoners are sound and complete. As an approximate reasoner, TrOWL gains efficiency by sacrificing completeness. This results in significant performance gain, as can be seen in Table 1. Hence, to assess the impact of TrOWL's inclusion on R_2O_2's performance, we conduct two sets of experiments, one with TrOWL included and one without.

3.2 Training R_2O_2

Training the prediction models. Using the 91 ontology metrics proposed previously [32,17] and dataset O_p, a performance prediction model is trained for each reasoner. Specifically, one Random Forest-based *regression model* (i.e., those in $\hat{R}$) is trained for each of the six reasoners (i.e., those in R). All the regression models in $\hat{R}$ are shown to be highly accurate, achieving high R^2 values that range from 0.73 (Konclude) to 0.91 (TrOWL).

Training the ranking models. Using the predicted reasoning time of the ontologies in O_r obtained from $\hat{R}$ as features, we trained five rankers (i.e., those in RM) specified in Section 2.2. The performance of each ranker is evaluated in terms of *precision at 1* (P@1). For a ranker, P@1 measures the proportion of the prediction model correctly ranked as the fastest. All rankers show high performance, achieving a P@1 between 88.7% and 90.6%.

3.3 Performance Evaluation of R_2O_2

We also employ P@1 as a performance metric for evaluating R_2O_2. For R_2O_2, P@1 measures the proportion of the component reasoner selected being the most efficient. We choose P@1 as we are only interested in evaluating how accurately R_2O_2 is able

[4] Data associated with the evaluation can be found at `http://www.csse.monash.edu.au/~yli/r2o2/`.

to recommend the single most efficient component reasoner, but not its ability to generate a total ranked list of the reasoners.

Table 1. Performance comparison, with TrOWL included as a component reasoner, between reasoners for O_t on: (1) average reasoning time per ontology (in seconds) for O_t, (2) average percentage of ontology being the fatest (P@1), and (3) average number of timeout ontologies per fold. The best performance values (lower is better) are typeset in **bold**. For comparison purposes, performance figures for the virtual best reasoner (VBR) are also shown.

Reasoner	Runtime in seconds (rank)	% P@1 (rank)	No. timeout
FaCT++	108.05 (7)	5.49 (4)	6.4
HermiT	75.31 (6)	0.00 (8)	3.9
JFact	239.93 (8)	0.07 (7)	12.4
Konclude	26.00 (4)	90.85 (2)	1.5
MORe	39.01 (5)	2.32 (5)	2.2
TrOWL	21.50 (2)	1.37 (6)	**0.8**
PR	24.24 (3)	90.00 (3)	1.5
R_2O_2	**17.52** (**1**)	**91.30** (**1**)	1.0
VBR	2.94 (–)	–	0.0

Table 2. Performance comparison, excluding TrOWL, between reasoners for O_t on: (1) average reasoning time per ontology (in seconds) for O_t, (2) average percentage of ontology being the fatest (P@1), and (3) average number of timeout ontologies per fold. The best performance values (lower is better) are typeset in **bold**. For comparison purposes, performance figures for the virtual best reasoner (VBR) are also shown.

Reasoner	Runtime in seconds (rank)	% P@1 (rank)	No. timeout
FaCT++	116.32 (6)	6.16 (4)	6.9
HermiT	79.39 (5)	0.11 (7)	4.1
JFact	234.4 (7)	0.14 (6)	13.2
Konclude	24.01 (2)	91.76 (2)	1.5
MORe	48.78 (4)	2.29 (5)	3.0
PR	24.36 (3)	90.53 (3)	1.5
R_2O_2	**16.46** (**1**)	**92.25 (1)**	**0.9**
VBR	7.52 (–)	-	0.3

Besides the six component reasoners, we also compare R_2O_2 against a Portfolio-based reasoner (denoted PR) in the spirit of SATzilla [30] as well as the *virtual best*

reasoner (denoted VBR), which always selects the most efficient component reasoner for any given ontology.

Table 1 and Table 2 show and compare the performance of R_2O_2 against all the other reasoners, across the 10 folds. TrOWL is in incomplete reasoner, and it gains efficiency by ignoring/approximating certain difficult language constructs. Two sets of experiments were conducted to assess the impact on performance of including TrOWL as a component reasoner in R_2O_2, one with TrOWL (Table 1) and one without (Table 2). For each reasoner r, three values are presented: (1) the average reasoning time per ontology (the lower the better), (2) the percentage of r being the most efficient (P@1, the higher the better), and (3) the average number of ontologies timed out on r (the lower the better) per fold. For R_2O_2, average ranking time per ontology is 0.03ms, trivial compared to reasoning time. For (1) and (2) above, the rank of each reasoner is also presented (the lower the better). As can be seen, in both experiments, R_2O_2 is the closest to VBR, and is more efficient than other reasoners with a speedup of up to 14x. It can also be observed that R_2O_2 times out on the smallest number of ontologies, which shows that R_2O_2 is not only efficient, its performance is also stable (less fluctuations).

It can be observed that Konclude dominates the other component reasoners, with a significantly faster reasoning time and a much larger percentage of being the fastest. However, R_2O_2 outperforms Konclude in both experiments, with a speedup of 1.48x and 1.46x respectively. This attests to the effectiveness of R_2O_2's approach: the combination of accurate performance prediction models and rankers successfully identifies the rare cases when another reasoner is faster than Konclude, and improves overall reasoning performance.

The evaluation also demonstrates the performance disparity among the reasoners. For example, JFact has the largest average runtime. However, it is the fastest for a small portion of ontologies, hence contributing to meta-reasoning. On the other hand, HermiT has a much smaller average runtime, but with less contribution to meta-reasoning. Moreover, even with a dominant reasoner such as Konclude, the other reasoners do outperform it sometimes (approximately 10%), and the performance of R_2O_2 as well as VBR validates the value of meta-reasoning: that the combination of reasoners indeed improves reasoning performance.

The effect of TrOWL is a little surprising. The performance of both PR and R_2O_2 remain relatively unchanged with or without the inclusion of TrOWL. However, VBR is significantly faster when TrOWL is included than when it is not, and it does not timeout on any ontology. It seems to suggest that TrOWL indeed reduces reasoning time for some hard ontologies. Further investigation is required to study the impact on reasoning performance and completeness.

To better assess reasoning performance, Figure 1 below shows a boxplot depicting the distributions of reasoning performance of the nine reasoners (including TrOWL and VBR). In a boxplot, the box itself contains the middle 50% of the values, the median (resp. mean) is represented by the horizontal bar (resp. '+') inside the box. The upper (resp. lower) whisker extends to the highest (resp. lowest) value within the upper (resp. lower) quartile. The reasoning time of all ontologies (across 10 folds) are also shown as dots in the plot to show their distributions. As can be seen, besides VBR, R_2O_2 has the lowest mean as well as median values, demonstrating its efficiency as well as stability.

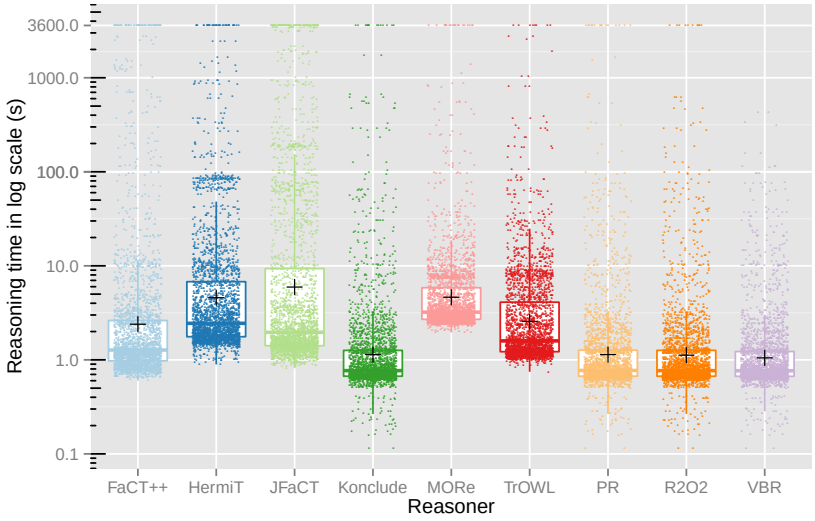

Fig. 1. A boxplot of the distributions of actual reasoning time (log-transformed) for the nine reasoners.

From Table 1 above, among the component reasoners, it seems that Konclude and TrOWL dominate the other component reasoners, as they are the fastest component reasoners and faster than the others by a large margin. It seems to make intuitive sense that they are the most efficient for most ontologies, and that R_2O_2 (and VBR) selects these two reasoners most of the time as the most efficient reasoner. However, this is not the case, as can be seen in Table 3 below. To better understand the reasoner selection based on performance, we discretise ontology reasoning time (in seconds) into four bins: 'A' (0, 1), 'B' [1, 10), 'C' [10, 100), and 'D' [100, 3,600]; and partition ontologies into these bins by their reasoning time by VBR. For brevity reasons only the figures for the experiment that includes TrOWL are shown. Note that in the training of R_2O_2, *actual*, but not discretised, reasoning is predicted.

Table 3. The percentage of each component reasoner being the most efficient reasoner (P@1=1) for each bin as well as for all the ontologies in O_t. The highest percentage in each bin is typeset in **bold**.

Reasoner	A	B	C	D	All
FaCT++	3.79	8.52	9.35	**50.00**	5.49
HermiT	0	0	0	0	0
JFaCT	0	0.25	0	0	0.07
Konclude	**96.21**	**86.22**	37.38	0.00	**90.85**
MORe	0.00	1.25	**48.60**	40.00	2.32
TrOWL	0.00	4.14	4.67	10.00	1.37

Table 3 shows the percentage each component reasoner being the most efficient (P@1=1) for ontologies in each bin and overall. A number of interesting observations can be made.

- Even though Konclude is the most efficient among the six reasoners, as can be seen in Table 1, Konclude does not dominate the other reasoners in bins 'C' and 'D', the bins for most difficult ontologies. Furthermore, even though Konclude has the fastest average runtime and overall P@1 value, it is not the fastest for any of the hardest ontologies (bin 'D').
- Even FaCT++ is only the fastest with a very small percentage overall (P@1=5.49%), it is the fastest for half of the most difficult ontologies (bin 'D').
- MORe has an even smaller percentage of being the fastest overall (P@1=2.32%), it is the fastest for 40% of the most difficult ontologies (bin 'D'). It is also the dominate reasoner for bin 'C'.

4 Related Work

Boolean satisfiability checking, or SAT, is a well-known and widely studied NP-complete decision problem. Many theoretical advances and practically useful heuristics have been developed over the past decades. However, it is recently recognised that the *empirical hardness* of NP-complete problems such as SAT [28,19] is still not well understood, and that theoretical, worst-case complexity analysis does not always provide useful insights on hardness of real-world problem instances. Hence, more research is needed to understand the sources of instance hardness and reasons why certain optimisations are effective while others are not, given a specific problem instance.

Ontology reasoning tasks such as consistency checking is a type of hard decision problems that may go beyond NP-hard. For very expressive DLs, ontology reasoning has a very high worst-case complexity of 2NExpTime-complete [4]. The research community has recognised the importance of empirical studies given the rapid development in ontology reasoning optimisation.

Ontology reasoners have been repeatedly benchmarked over the years [7,2,5,16]. It is observed from these benchmarking efforts that different reasoners have different levels of support, robustness and efficiency for ontologies with different features (e.g., language constructs used and their interactions), confirming the need for further investigation of sources of instance hardness.

Recently, the OWL Reasoner Evaluation (ORE) workshop series began an OWL reasoner competition [9,1] on a number of reasoning tasks (consistency checking, classification, and realisation) and for different profiles of the OWL language (OWL 2 DL and EL). The recent reasoner Konclude [22] is a novel parallellised OWL 2 DL reasoner. It is very efficient, significantly outperforming other reasoners in the latest ORE 2014 competition, winning 5 of the 6 categories. As part of the competition, the ontology corpus and the competition framework have been made publicly available, making it easy to reuse the data and to reproduce the results.

The empirical *robustness* of OWL reasoners was investigated [10]. A reasoner is said to be *successful* for a given ontology if it successfully loads the ontology and performs reasoning within a specified timeout cutoff (e.g., two hours). A reasoner

is *robust* if it is successful for at least 90% of a given corpus. Experiments on 4 OWL reasoners and three corpora of ontologies revealed that the *best combo*, the virtual best reasoner (or the meta-reasoner), is "extremely robust over all corpora" [10], achieving an overall robustness of over 98% for all three corpora. However, such a meta-reasoner was only identified manually and *post festum*.

Inspired by the success of empirical software engineering research, we proposed a number of metrics to measure the design complexity of ontologies [32]. These metrics measure various aspects of complexity: overall complexity of the ontology, complexity of classes and properties, as well as those characterising complex class and property expressions.

We studied the problem of estimating ontology reasoning time by applying machine learning techniques to building classifiers [15] and regression models [17] to estimate (discretised or actual) reasoning time of a given (ontology, reasoner) pair. High accuracy was achieved in both approaches, which were used to identify important features that affect performance the most, and to identify performance hotspots efficiently. An ontology is represented by a number of syntactic and structural metrics that are efficient to calculate. These metrics are used as features to train classifiers and regression models, one for each reasoner.

A different, *local*, reasoning performance prediction method was proposed in [21]. It decomposes an ontology into smaller subsets with increasing sizes, and then extrapolates their performance to the entire ontology. The local approach does not require a corpus, but instead does require repeated reasoner invocations over ontology subsets. Using the k-NN classifier as a baseline, it was observed that using only one metric, number of axioms, the local approach achieves comparable classification performance with [15]. The local prediction approach can be regarded as an *online* prediction approach as it needs to invoke reasoners on the subsets of growing sizes, whereas [15,17] as well as this work are *offline*.

Understanding empirical hardness of ontologies has also garnered attention in recent years. The identification of sources of ontologies in terms of *performance hotspots* was investigated [11,17], where hotspots are found in a number of hard biomedical ontologies. The removal of such hotspots dramatically reduces the reasoning time of the remaining ontology. However, as a result, reasoning soundness and completeness cannot be guaranteed.

Portfolio-based algorithm selection [13] has been successfully applied to combinatorial optimisation and constraint satisfaction problems. SATzilla [30], for instance, a portfolio SAT solver, has demonstrated higher efficiency over single solvers. Compared to ontology languages OWL and OWL 2, k-SAT instances are described by a simpler language, whereas OWL and OWL 2 contain many more language constructs (various class expressions, property expressions and axioms). The richness of the ontology languages make it difficult to define features to sufficiently describe characteristics of ontologies. Moreover, SATzilla does not employ a ranking component but solely relies on prediction models. Evaluation in Section 3 above shows that R_2O_2 outperforms a SATzilla-style portfolio reasoner, demonstrating the advantages of integrating a ranking component.

Recently, preference learning [6] has been shown to be effective for developing meta-learners with an aim to predicting the most efficient algorithm from a few promising ones on different types of datasets such as those represented using meta-features [23] and data streams [27]. In these studies, preference learning has been demonstrated to significantly reduce optimisation time needed for choosing a best model on a given dataset.

Often, ranking algorithms in preference learning use machine learning approaches by analysing the association information between characteristics of a given dataset (in our context, ontology metrics) and the relative performance of the available algorithms. Such ranking algorithms include the algorithms proposed in [23] such as the nearest neighbour-based approach, the pairwise binary classification approach, the regression-based approach, the ranking tree-based approach and the ranking forest-based approach. Also, the learning-to-rank approach [29] and the label ranking approach [12] have been shown to be used for preference learning. In R_2O_2, we utilise some of the ranking algorithms introduced in [23] to predict the most efficient reasoners for a set of new ontologies. The incorporation of such a ranking component demonstrably improves reasoning efficiency.

5 Conclusions

In this paper, we present R_2O_2, a novel meta-reasoner that combines component reasoners in an efficient way, by automatically selecting the reasoner that is most likely the most efficient for any given ontology. A key novel feature of our approach is the incorporation of reasoner ranking to determine the best component reasoner according to their predicted reasoning time. Another important feature is the use of prediction models for ontology reasoners for estimating reasoning time. The performance of R_2O_2 is further improved by the incorporation of the *second-order* prediction (ranking), as compared to the traditional portfolio-based meta-reasoning approach.

Our comprehensive, large-scale evaluation involving more than 4,000 ontologies and 6 state-of-the-art OWL 2 DL reasoners—including the currently dominant reasoner Konclude—demonstrates the superiority of our meta-reasoner, in both efficiency and stability. A speedup of up to 14x is achieved, with a 1.4x speedup over Konclude. We show that R_2O_2 achieves significant efficiency improvements over the 6 component reasoners, as well as a SATzilla-style portfolio reasoner with a 1.5x speedup.

In future we will investigate novel prediction models and ranking models to further improve their accuracy, as well as the performance of R_2O_2. We will study the effectiveness of incorporting additional efficient reasoners such as ELK [18] that provide efficient reasoning support of less expressive DLs. Moreover, we will also investigate sources of instance hardness by studying similarity of ontologies that have similar performance for a given reasoner.

Acknowledgment

We thank Andreas Steigmiller for his kind and timely assistance in helping us set up the ORE 2014 reasoner competition framework in our evaluation.

References

1. Samantha Bail, Birte Glimm, Ernesto Jiménez-Ruiz, Nicolas Matentzoglu, Bijan Parsia, and Andreas Steigmiller. Summary ore 2014 competition. In *Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE 2014)*, volume 1207, pages iv–vii. CEUR Workshop Proceedings, 2014.
2. Jurgen Bock, Peter Haase, Qiu Ji, and Raphael Volz. Benchmarking OWL reasoners. In *ARea2008 - Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*, June 2008.
3. Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *23rd Annual Conference on Neural Information Processing Systems (NIPS 2009).*, pages 315–323. Curran Associates, Inc., 2009.
4. Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6:309–322, November 2008.
5. Kathrin Dentler, Ronald Cornet, Annette ten Teije, and Nicolette de Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web Journal*, 2(2):71–87, 2011.
6. Johannes Frünkranz and Eyke Hüllermeier. *Preference Learning*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
7. Tom Gardiner, Dmitry Tsarkov, and Ian Horrocks. Framework for an automated comparison of description logic reasoners. In *Proceedings of the 5th International Conference on The Semantic Web*, ISWC'06, pages 654–667, 2006. Springer-Verlag.
8. Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An owl 2 reasoner. *J. Autom. Reason.*, 53(3):245–269, October 2014.
9. Rafael S. Gonçalves, Samantha Bail, Ernesto Jiménez-Ruiz, Nicolas Matentzoglu, Bijan Parsia, Birte Glimm, and Yevgeny Kazakov. OWL reasoner evaluation (ORE) workshop 2013 results: Short report. In *2nd International Workshop on OWL Reasoner Evaluation (ORE-2013)*, volume 1015 of *CEUR Workshop Proceedings*, pages 1–18. 2013.
10. Rafael S. Gonçalves, Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler. The empirical robustness of description logic classification. In *Description Logics*, volume 1014 of *CEUR Workshop Proceedings*, pages 197–208. 2013.
11. Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler. Performance heterogeneity and approximate reasoning in description logic ontologies. In *11th International Semantic Web Conference (ISWC2012)*, volume 7649 of *LNCS*, pages 82–98. Springer, 2012.
12. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, November 2008.
13. Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artif. Intell.*, 206:79–111, 2014.
14. Yong-Bin Kang, Shonali Krishnaswamy, and Yuan-Fang Li. A meta-reasoner to rule them all: Automated selection of OWL reasoners based on efficiency. In *23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM'14)*, pages 1935–1938, New York, NY, USA, 2014. ACM.
15. Yong-Bin Kang, Yuan-Fang Li, and Shonali Krishnaswamy. Predicting reasoning performance using ontology metrics. In *11th International Semantic Web Conference (ISWC2012)*, volume 7649 of *LNCS*, pages 198–214. Springer, 2012.
16. Yong-Bin Kang, Yuan-Fang Li, and Shonali Krishnaswamy. A rigorous characterization of reasoning performance – a tale of four reasoners. In *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012)*, June 2012.

17. Yong-Bin Kang, Jeff Z. Pan, Shonali Krishnaswamy, Wudhichart Sawangphol, and Yuan-Fang Li. How long will it take? accurate prediction of ontology reasoning performance. In *AAAI*, pages 80–86. AAAI Press, 2014.
18. Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reasoning*, 53(1):1–61, 2014.
19. Kevin Leyton-Brown, Holger H. Hoos, Frank Hutter, and Lin Xu. Understanding the empirical hardness of *NP*-complete problems. *Commun. ACM*, 57(5):98–107, 2014.
20. Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. More: Modular combination of owl reasoners for ontology classification. In *11th International Semantic Web Conference (ISWC2012)*, volume 7649 of *LNCS*, pages 1–16. Springer, 2012.
21. Viachaslau Sazonau, Uli Sattler, and Gavin Brown. Predicting OWL reasoners: Locally or globally? In *27th International Workshop on Description Logics*, volume 1193 of *CEUR Workshop Proceedings*, pages 713–724. 2014.
22. Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *J. Web Sem.*, 27:78–85, 2014.
23. Quan Sun and Bernhard Pfahringer. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, 93(1):141–161, 2013.
24. Edward Thomas, Jeff Z. Pan, and Yuan Ren. TrOWL: Tractable OWL 2 reasoning infrastructure. In *ESWC (2)*, pages 431–435. Springer, 2010.
25. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, pages 292–297. Springer, 2006.
26. Dmitry Tsarkov and Ignazio Palmisano. Chainsaw: a metareasoner for large ontologies. In *1st International Workshop on OWL Reasoner Evaluation (ORE-2012)*, volume 858 of *CEUR Workshop Proceedings*. 2012.
27. Jan N. van Rijn, Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. Algorithm selection on data streams. In *17th International Conference on Discovery Science (DS 2014)*, volume 8777 of *LNCS*, pages 325–336. Springer, 2014.
28. Moshe Y. Vardi. Boolean satisfiability: Theory and engineering. *Commun. ACM*, 57(3):5–5, March 2014.
29. Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *30th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.
30. Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla-07: The design and analysis of an algorithm portfolio for sat. In *CP*, volume 4741 of *LNCS*, pages 712–727. Springer, 2007.
31. Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.
32. Hongyu Zhang, Yuan-Fang Li, and Hee Beng Kuan Tan. Measuring design complexity of Semantic Web ontologies. *Journal of Systems and Software*, 83(5):803–814, 2010.