

Analyse en Moyenne de la Complexité des Algorithmes d'Apprentissage Relationnel

Erick ALPHONSE, Aomar OSMANI

Laboratoire LIPN- UMR CNRS 7030, Université Paris 13

Résumé : Ce papier présente pour la première fois une analyse empirique de la complexité des algorithmes d'apprentissage relationnel sur tout le paysage de la complexité des problèmes (en particulier dans le cadre du motif standard “easy-hard-easy”). Cette analyse est rendue possible grâce aux récents travaux en transition de phase où un générateur de problèmes aléatoires a été proposé, dont les paramètres d'ordre permettent de traverser tout le spectre de la complexité. Dans le cas que nous avons considéré, le paramètre d'ordre utilisé est le nombre d'exemples positifs et négatifs.

Nous avons analysé le comportement des algorithmes d'apprentissage relationnel les plus utilisés. Les résultats obtenus montrent qu'il reste encore du chemin à parcourir pour obtenir des algorithmes efficaces en apprentissage. En particulier, la courbe de la complexité des algorithmes testés montrent un profil “easy-hard-hard” – sauf pour le cas de l'approche ascendante guidée par les données (lgg) – alors que la courbe théorique possède un profil “easy-hard-easy”. Cette étude ouvre la voie vers une analyse en profondeur des comportements des algorithmes et la proposition d'algorithmes se rapprochant le plus du référentiel théorique désormais disponible.

Mots-clés : Apprentissage relationnel, transition de phase, complexité des algorithmes

1 Introduction

Symbolic learning has been known for more than a quarter of a century as search into a state space (Mitchell, 1982). Worst-case complexity analysis, as conducted in the PAC framework (Valiant, 1984), showed that search in languages of interest, as those typically considered in Relation Learning (RL), is NP-hard (e.g. (Haussler, 1989)) and even NP^{NP} -hard (or Σ_2^P -hard) in the richer settings (Gottlob *et al.*, 1997). However, as far as we know, no works have studied the behaviour of learning algorithms in the “average” or “typical” complexity case, more representative of real-world applications. This is despite the important recent progress realised in various combinatorics domains, such as SAT or CSP, which have not been imported to the realm of symbolic learning. Particularly since the seminal works of (Cheeseman *et al.*, 1991), the way search algorithms are empirically evaluated has indeed drastically changed. Search algorithms are evaluated and compared in the *phase transition framework* on random problem instances where the inherent resolution complexity is determined by order parameters. This tool allows to design benchmark datasets to point out and filter “bad” algorithms, and to promote “good” algorithms which are close to the “easy-hard-easy” pattern, standard nowadays (Smith & Dyer, 1996).

This paper presents, for the first time, a study of learning algorithms on inherent hard problems in RL. The most popular algorithms are compared and the results are clear : the top-down approaches, either rooted in the generate-and-test or the data-driven paradigms are not good. We show that they exhibit a “easy-hard-hard” pattern, although the theoretic pattern is “easy-hard-easy”. Only the bottom-up data-driven approach manages to efficiently solve the insoluble problems and then exhibits the desired pattern. Those results are obtained using classical complete relational learners found in the learning systems Aleph (Srinivasan, 1999), Progol (Muggleton, 1995) and Propal (Alphonse & Rouveirol, 2006). Informed search algorithms such as Best-First Top-down Generate-and-Test (BESTF-TGT), A-search Top-down Generate-and-Test (A-TGT), and non-informed ones, such as Breadth-First Top-down Generate-and-Test and Data-Driven (BF-TGT and BF-TDD), Depth-First Top-Down Generate-and-Test (DF-TGT) and Depth-First Bottom-up Data-Driven (DF-BDD) are tested.

Learning algorithms have never been studied on reliable hard instances, and the expected benefits of the proposed tool are the same as obtained in other combinatorics domains : (1) pointing out and filter bad algorithms ; (2) importing and adapting the best search strategies developed in other domains ; (3) developing a unified framework for the empirical evaluation of learners, not only based on real-world applications (as this is always necessary), but also on problems with controlled complexity ; (4) understanding scaling-up problems acknowledged in RL (see e.g. (Page & Srinivasan, 2003)) by studying their behaviour on inherent hard instances in the phase transition.

It has to be noted that (Rückert *et al.*, 2002) studied the phase transition of the well-known NP-complete k-term DNF consistency problem, although they didn’t use it to evaluate complete learners as they are not popular in attribute-value learning, as opposed to relational learning which is arguably harder.

The paper is organised as follows : section 2 presents RL and search strategies background. Section 3 recalls the main results on the PT framework and the “easy-hard-easy” pattern. Section 4 describes the random problem instance generator. Section 5 presents the main results of the paper where complete learners’ behaviour is shown and analysed. Finally, section 6 concludes the paper.

2 Relational Learning

In machine learning, given a learning set $E = E^+ \cup E^-$, with positive and negative examples of the unknown target concept, drawn from an example language \mathcal{L}_e , a hypothesis language \mathcal{L}_h , a generality relation \geq which relates \mathcal{L}_e and \mathcal{L}_h and partially order the hypotheses. After (Mitchell, 1982), (symbolic) learning is defined as search in \mathcal{L}_h . The consistency problem, is to find a hypothesis $h \in \mathcal{L}_h$ such that h is consistent with the data. A given hypothesis h is consistent iff it is complete : $\forall e^+ \in E^+, h \geq e^+$ and correct : $\forall e^- \in E^-, h \not\geq e^-$. The consistency problem is fundamental in learning as it is at the core of the Statistical Learning Theory, notably studied in the PAC framework (see (Valiant, 1984; Haussler, 1989) for details). This *a fortiori* is true in RL where almost all noise-resistant learners are relaxation of this problem (Fürnkranz, 1997), therefore studying this problem will benefit search strategies for learning.

In this article, we study a typical instance of this problem in RL, known as the ILP consistency problem for function-free Horn clauses (Gottlob *et al.*, 1997) : given \mathcal{L}_e ,

a language of function-free ground Horn clauses, \mathcal{L}_h , a language of non-recursive function-free Horn clauses and an integer k polynomial in $|E^+ \cup E^-|$, does there exist $h \in \mathcal{L}_h$ with no more than k such that h logically implies each element in E^+ and h does not implies any element in E^- . In such hypothesis space, the logical implication is equivalent to θ -subsumption¹ which is NP-complete and therefore decidable (Gottlob, 1987). This result implies that RL is higher than attribute-value learning in the polynomial hierarchy. (Gottlob *et al.*, 1997) proved that this problem is Σ_2^P -complete : the search is NP-complete and it is guided by the subsumption test which is NP-complete.

The Mitchell seminal paper (Mitchell, 1982), relating 'symbolic' concept learning to search in a state space, has enabled machine learning to integrate techniques from problem solving, operational research and combinatorics : greedy search in C4.5 and FOIL, beam search in AQ and CN2, breadth-first search in Aleph (Srinivasan, 1999), 'A' search in PROGOL (Muggleton, 1995), IDA (Iterative-Deepening A) search in MIO to name a few systems.

Mitchell further refines the search strategy into the generate-and-test (GT) and data-driven (DD) strategies. Virtually all GT algorithms are top-down, as it appeared early that a bottom-up approach would start with a too specific hypothesis to be efficiently guided by a heuristic function. In this paradigm, the top-down refinement operator, noted ρ , is only based on the structure of the hypothesis space, independently of the learning data : Let $h \in \mathcal{L}_h : \rho(h) = \{h' \in \mathcal{L}_h | h \geq h'\}$.

Therefore, generate-and-test algorithms have to deal with many refinements that are not relevant with respect to the discrimination task. They only rely on the evaluation function to prune the irrelevant branches. On the contrary, the top-down DD (TDD) strategy searches the space of hypotheses that are more general than or equal to a given positive example, named the seed example, and uses negative examples to prune irrelevant branches in the refinement graph. Formally, the TDD refinement operator is defined as a binary operator : Let $h \in \mathcal{L}_h, e^- \in E^- : \rho(h, e^-) = \{h' \in \mathcal{L}_h | h \geq h' \text{ and } h' \not\geq e^-\}$. As opposed to a TGT approach, a TDD one can therefore compensate for a poor evaluation function by using the learning data (Alphonse & Rouveirol, 2006). Moreover, some TDD strategies make the most of the negative instances in order to select informative negative examples.

Dually to the TDD strategy, the Bottom-up Data Driven (BDD) strategy relies on positive examples to guide its generalisation step. Its BDD refinement operator is given as follows : Let $h \in \mathcal{L}_h, e^+ \in E^+ : \delta(h, e^+) = \{h' \in \mathcal{L}_h | h \leq h' \text{ and } h' \geq e^+\}$. This strategy has been first formalised by (Plotkin, 1970), who made the link between generalisation in learning and lowest-upper bound in lattice theory. Such an operator, also known as least-general generalisation (lgg) or most-specific generalisation (msg), has know several theoretic developments (Valiant, 1984; Haussler, 1989) but has been seldom used in learning systems.

3 Phase transition

Phase transition (PT) in physical systems is the abrupt change of its properties at certain values of the defined order parameters. The water/ice abrupt transition at particular

¹Let C, D two clauses. C θ -subsumes D , noted $C \geq_\theta D$ iff \exists a substitution θ such that $C\theta \subseteq D$.

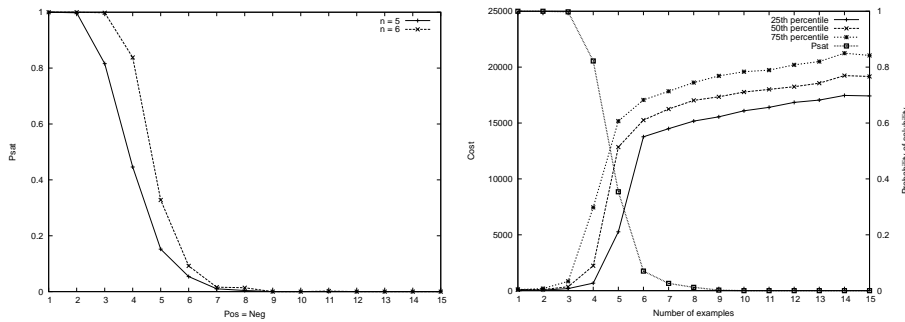


FIG. 1 – Probability of solubility according to the number of learning examples (=Pos strategy for various percentiles and probability of solubility, for $n = 6$).

temperatures and pressures is a typical example. Since two decades many works are done on PT in computer science, but as pointed in (Parkes, 1997) it seems that this area started with the remarkable observation in (Erdős & Rényi, 1960) that thresholds in properties such as connectivity emerge in large random graphs. Constraint satisfaction problem (CSP) and propositional satisfiability problem (SAT) are the communities where this framework is more studied (Cheeseman *et al.*, 1991; Selman *et al.*, 1992; Smith, 2001; Hogg & Williams, 1994; Gent & Walsh, 1999).

In the PT framework, it is conjectured that the hardest problem instances occur in the PT (see e.g. (Cheeseman *et al.*, 1991; Davenport, 1995; Gent & Walsh, 1999)). The under-constraint problems from the “yes” region appear to be easily solvable, as there are many solutions. This is the same for over-constraint problems from the “no” region as it is easy to prove that they are insoluble. These findings have been corroborated on several problems, with different types of algorithms, and it is considered that the problem instances appearing in the PT are inherently hard, independently of the algorithms used. In the “yes” and “no” regions, the easy ones, the complexity appears to be very dependent of the algorithm. There are, in these regions, some problems exceptionally hard, whose complexity dominates the complexity of instance problems in the PT region for certain types of algorithms (Hogg & Williams, 1994; Davenport, 1995; Gent & Walsh, 1994). In other words, a “good” algorithm when studied along the three different regions has to exhibit an average complexity following the so-called “easy-hard-easy” pattern.

The interest of the framework is two-fold as it gives a way to empirically assess the efficiency of algorithms on classes of problems whose inherent complexity is controlled by order parameters, and as finding ways to generate hard instances for a problem is important to understanding the complexity of the problem (Xu & Li, 2006; Cook & Mitchell, 1997). Such framework has been developed recently in RL, whose complexity class is beyond NP and is presented in the next section.

4 Problem Generator RLPG

We reuse the random problem instance generator proposed by (Alphonse & Osmani, 2008; Alphonse, 2009) to study the bounded ILP consistency problem. They used this

generator, named model RLPG (Relational Learning Problem Generator), to exhibit the PT of learning, as well as the NP-complete subsumption test, but did not investigate this generator as a benchmarking tool for learners.

A learning problem instance in this model is denoted $RLPG(k, n, \alpha, N, Pos, Neg)$. The parameters k, n, α, N are related to the definition of the hypothesis and example spaces. Pos and Neg are the number of positive and negative examples respectively. $k \geq 2$ denotes the arity of each predicate present in the learning language, $n \geq 2$ the number of variables in the hypothesis space, α the domain size for all variables as being equal to n^α , and N the number of literals in the examples built on a given predicate symbol. Given k and n , the size of the bottom clause of the hypothesis space \mathcal{L}_h is $\binom{n}{k}$, and encodes the largest constraint network of the underlying CSP model. Each constraint between variables is encoded by a literal built on a unique predicate symbol. \mathcal{L}_h is then defined as the power set of the bottom clause, which is isomorphic to a boolean lattice. Its size is $2^{\binom{n}{k}}$.

Learning examples are randomly drawn, independently and identically distributed, given k, n, α and N . Their size is $N \cdot \binom{n}{k}$. Each example defines N literals for each predicate symbol. The N tuples of constants used to define those literals are drawn uniformly and without replacement from the possible set of $\binom{n^\alpha}{k}$ tuples.

As an illustration, table 1 shows a random $RLPG(2, 3, \alpha, 1, 1, 1)$ problem, with α such that $n^\alpha = 5$. The first line shows the bottom-most element of the hypothesis space, which encodes all binary constraints between 3 variables. The next two lines show the positive and the negative example, respectively, allowing only one matching of a given predicate symbol (as $N = 1$). The search space is of size 2^3 and consists of all hypotheses built with the same head as the bottom clause, and with a subset of its body as body. In such a space, it is easy to see that there is no solution, given that no hypothesis subsumes the positive example without subsuming the negative example.

\perp	$p0(A) \leftarrow p1(A, B, C), p2(A, B, D), p3(A, C, D)$
+	$p0(e1) \leftarrow p1(e1, b, c), p2(e1, c, d), p3(e1, e, f)$
-	$p0(e2) \leftarrow p1(e2, c, f), p2(e2, d, e), p3(e2, d, c)$

TAB. 1 – Example of a random learning problem generated with RLPG, without solutions.

Whereas the problem illustrated in table 2 accepts the following clause as solution : $p0(A) \leftarrow p2(A, B, D), p3(A, C, D)$.

\perp	$p0(A) \leftarrow p1(A, B, C), p2(A, B, D), p3(A, C, D)$
+	$p0(e1) \leftarrow p1(e1, b, c), p2(e1, d, e), p3(e1, e, e)$
-	$p0(e2) \leftarrow p1(e2, b, b), p2(e2, e, e), p3(e2, e, c)$

TAB. 2 – Example of a random learning problem generated with RLPG, with a solution.

(Alphonse, 2009) showed that the number of positive and negative examples were order parameters of this Σ_2^P -complete problem. For instance, if one keeps the number of negative examples constant and increases the number of positive examples, one wanders from a under-constraint region, where there is almost surely a solution, to an over-constrained region, where there is no solution, as the solution has to be more and more general to cover all positive examples but ends up too general to stay correct. The same observation is made if one keeps the number of positive examples constant and varies the number of negative ones.

In the next section, we investigate the case where the number of positive and negative examples are the same and vary to exhibit the PT in problem solubility, as showed in figure 1, for two problem sizes, $n = 5$ and $n = 6$, which define a hypothesis space of size 2^{10} and 2^{15} , respectively.

5 Evaluating Complete learners

We evaluate complete learners representative of the search strategies described in section 2. As non-informed searches, we use the Breadth-First TGT search (BF-TGT) and the Depth-First TGT search (DF-TGT). As informed searches, we use the A TGT search (A-TGT) and the Best-First TGT search (BESTF-TGT). Informed search makes use of an evaluation function to minimise, whose general form is $f = g + h$. g is defined as the cost from the start to the current hypothesis and h as an estimation of the distance from the current hypothesis to the goal. We define A-TGT according to the Progol system : g is defined as the length of the current hypothesis and h has the difference between the number of negative examples and the number of positive examples. In our context, as all positive examples must be subsumed, it simplifies to the number of negative examples. BESTF-TGT is not biased towards shorter hypotheses and defines $g = 0$. We refer to (Srinivasan, 1999; Muggleton, 1995) for details about their implementations.

The next learning strategy we study is the one used in the TDD learner Propal. This is an incomplete learner as it performs a beam search guided by the Laplace function. So we set Propal with a beam of unlimited size, which basically turns down to a non-informed Breadth-First search (BF-TDD). The only difference is that when the solution is reached at a level of the search, it will be the first picked up at the next level. Note also that, as an incomplete learner, it does not have an optimal refinement operator, like the other learners, and may evaluate the same hypothesis several times.

The last learning strategy is Depth-First BDD (DF-BDD), based on Plotkin's lgg operator, following the implementation of (Alphonse, 2009). Note that, as the hypothesis space is finite, it is not a lattice under θ -subsumption, and the refinement operator outputs the set of all least-general generalisations. This operator is applied depth first. The computation of lgg is done with depth-first search into possible subsets of the hypothesis and it outputs the largest subsets that subsume the example.

We evaluate complete RL learners on random problem instances whose inherent complexity is controlled by the order parameter of the PT. We plot their search cost as a function of the order parameter to compare their complexity pattern to the standard "easy-hard-easy" pattern, as it is an indication of search efficiency. As the consistency problem in RL is Σ_2^P -complete (see section 2), the search cost measurement has to take into account both the cost of the exploration of the hypothesis space and the cost of the consistency check. We propose to measure both the number of backtracks of the subsumption procedure and the time in milliseconds needed to solve a learning problem. The former measure is relevant for GT approaches, as the cost of the refinement operator is negligible compared to the subsumption cost, and it reflects the number of evaluated hypotheses. This is also the case for DF-BDD, as the lgg operator uses the subsumption test to find the common generalisations of two given clauses. However, it is not appropriate for BF-TDD which is based on the Propal system. Propal delegates the computation of refinements to a Weighted CSP solver (Alphonse & Rouveirol, 2006)

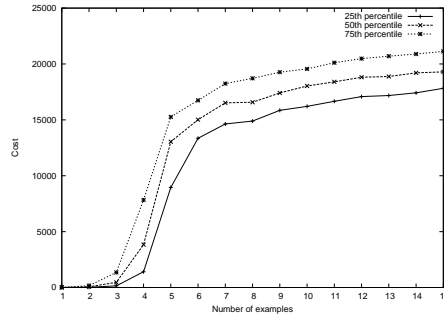
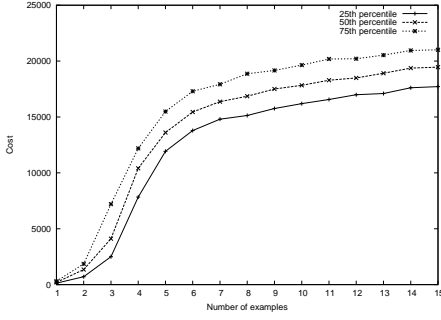


FIG. 3 – Backtracking cost using BF-TGT strategy for various percentiles, for $n = 6$. FIG. 4 – Backtracking cost using DF-TGT strategy for various percentiles, for $n = 6$.

whose cost does not translate into backtracks of the subsumption test. It would be interesting to propose a relevant cost measure for all RL learners, independently on the implementation but we leave it for future research. Thus, we use the resolution time as cost measure for this strategy, and although it does not allow a direct comparison with other approaches, it is still relevant to study its expected cost pattern.

All experiments are done using instances from $RLPG(k, n, \alpha, N, Pos, Neg)$, with $k = 2$, $n = 5$ and $n = 6$ to study different problem sizes, $\alpha = 1.4$ and $N = 10$. Additional experiments using different parameter values (not shown here) have been conducted and result in similar findings. In the following figures every plot is averaged over 500 randomly drawn learning problems.

Figure 2 shows the results obtained with A-TGT, for $n = 6$. We can see that the easy problems resolution from the “yes” region follows the standard pattern. The superposition of the solubility probability plot shows the PT region. The cost sharply increases as soon as the probability solubility is no longer 1 (when both the number of positive and negative examples are greater than 3). This exponential increase stops when the probability gets close to 0. However, the plot does not reach a maximum right after the PT. This is indicative of a bad search algorithm, as the backtracking cost keeps increasing, as the number of examples increase, in the region theoretically easy, dominating then the cost in the PT. We are going to see that this behaviour is typical of the top-down approaches : interestingly, in the “no” region, extra examples do not help enough pruning the hypothesis space to compensate the increase in subsumption cost.

For various percentiles, figure 3 shows that BF-TGT, as a non-informed search strategy, is costly very early in the “yes” region. However, after the PT, A-TGT and BF-TGT are about equivalent : they cannot cope with an increasing number of examples, and the cost in the “no” region dominates the cost in the PT region.

In the “yes” region, DF-TGT behaves better than BF-TGT. This is particularly true in the “yes” region where there are a lot of solutions. In that case, keeping specialising a complete hypothesis leads almost surely to a consistent hypothesis. DF-TGT is as good as A-TGT in this region, but when they get closer to the PT, A-TGT performs better. Its heuristic function prioritizes hypotheses which discriminate negative examples the most and this seems to lead to consistent hypotheses faster. In the “no” region, we see again that DF-TGT degenerates as A-TGT and BF-TGT.

In figure 5, we show results for the data-driven search, DF-BDD, first on problems of

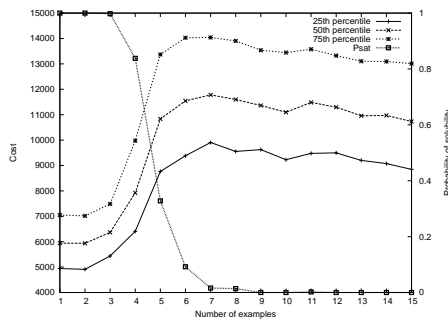
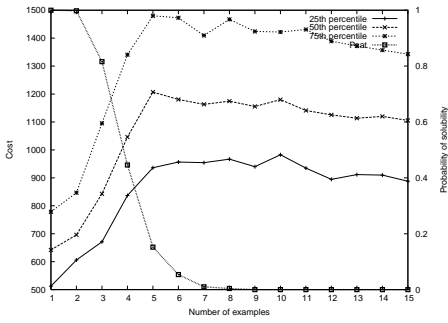


FIG. 5 – Backtracking cost using DF-BDD strategy for various percentiles, for $n = 5$. FIG. 6 – Backtracking cost using DF-BDD strategy for various percentiles, for $n = 6$.

size $n = 5$. It gets close to the standard pattern for the “yes” region problems. We note however that for higher percentiles (e.g. the median) the algorithm has a non negligible cost even for 1 positive and 1 negative example. Moreover, the superposition of the solubility plot shows the cross-over point of the PT between 4 and 5 examples and that the complexity peak is slightly shifted to the right with respect to this point, which indicates that DF-BDD’s cost pattern is close to the “easy-hard-easy” pattern. Also, we see that for all percentiles, the cost slowly decreases after the PT. We can say that this algorithm is a good search algorithm, also some improvements can be done.

Figure 6 shows results for the same algorithm, but on larger problems, with $n = 6$. The cross-over point is now around 5, and DF-BDD’s behaviour gets closer to the standard pattern in the “yes” region. Although there is a minimum cost (6000 backtracks as median cost), certainly due to the naive implementation of the lgg refinement operator, this cost does not vary much in the “yes” region. Among all tested algorithms, it is the only one exhibiting the “easy-hard-easy” pattern.

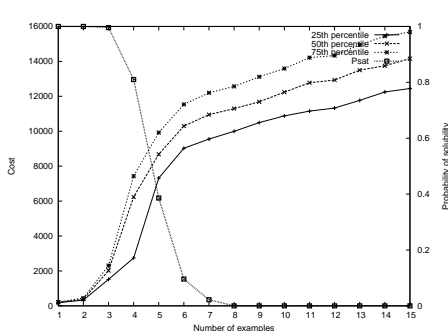
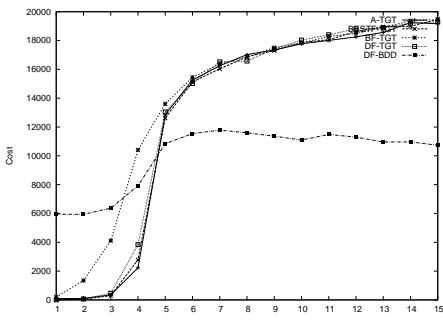


FIG. 7 – Median backtracking cost for A-TGT, BF-TGT, DF-TGT, BESTF-TGT and for various percentiles, for $n = 6$. FIG. 8 – Time cost using BF-TDD strategy for various percentiles, for $n = 6$.

Figure 7 summarises the backtracking cost of the search algorithms discussed above, with the addition of BESTF-TGT. The results are clear : all GT approaches are interesting for problems with a lot of solutions but are particularly bad when there are few or no solutions. Moreover, either informed or non-informed search strategies, they all have the same profile in this latter case, which is an interesting point to detail in the future.

Conversely, DF-BDD, although penalised in the “yes” region, is more efficient in those problems with few or no solutions, with a decrease in cost as the number of examples increase.

The complexity analysis limited to the number of backtracks of the subsumption test is not enough for this study because it does not take into account the cost of the refinement operators for all approaches, as for BF-TDD (see above). We then complete it by plotting the resolution time of BF-TDD in figure 8. Although the search cost cannot be directly compared, we see that it behaves similarly to the other top-down approaches. In the “yes” region, the TDD operator cannot compensate the breadth-first search with its smaller branching factor, and therefore behaves like BF-TGT. After the exponential increase in cost on the inherent hard instances, the cost keeps increasing as the number of examples grows in the “no” region. The penalty here is that the number of calls to the Weighted CSP solver to compute a near-miss is proportional to the number of negative examples. This is clearly too costly and the trade-off between the quality of the near-miss and the reduction of the search space has to be evaluated.

6 Conclusion

This paper presents for the first time an empirical evaluation of popular RL algorithms in the PT framework. This framework has been introduced only recently in RL, as most works have been done on NP-complete problems, and RL is more difficult, being one class higher in the polynomial hierarchy than NP.

We used this framework to generate benchmark datasets with controlled complexity, based on conjectures linking the probability of problem solubility with inherent problem hardness. First, this study shows that all well-known top-down relational algorithms, rooted either in the generate-and-test or the data-driven paradigm, are bad as they fail to exhibit the standard “easy-hard-easy” pattern. Their complexity tends to increase with the number of examples, although the extra examples do not change the solubility of the problem, and therefore they exhibit an “easy-hard-hard” pattern. This has to be contrasted with DF-BDD, a lgg-based learner, which does not perform as well on the easy problems in the “yes” region, but well on the easy problems of the “no” region, as well as in the PT compared to the other algorithms. This study shows that search strategies standard in RL lag behind what is considered state of the art in other combinatorics communities. We hope that it will enable RL and ILP to import and/or develop better search algorithms, to eventually benefit to better scaling relational learners.

Références

- ALPHONSE E. (2009). Étude empirique de la transition de phase en apprentissage relationnel. In *Actes de la Conférence d'Apprentissage (CAP'09)*, p. 173–184.
- ALPHONSE E. & OSMANI A. (2008). A model to study phase transition and plateaus in relational learning. In *Proc. of Conf. on ILP*, p. 6–23.
- ALPHONSE E. & ROUVEIROL C. (2006). Extension of the top-down data-driven strategy to ILP. In *Proc. of Conf. on ILP* : Springer Verlag.
- CHEESEMAN P., KANEFISKY B. & TAYLOR W. (1991). Where the really hard problems are. In R. MYOPOULOS, JOHN ; REITER, Ed., *Proc. of the 12th IJCAI*, p. 331–340 : Morgan Kaufmann.

- COOK S. A. & MITCHELL D. G. (1997). Finding hard instances of the satisfiability problem : A survey. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, p. 1–17 : American Mathematical Society.
- DAVENPORT A. (1995). A comparison of complete and incomplete algorithms in the easy and hard regions. In *Workshop on Studying and Solving Really Hard Problems, CP-95*, p. 43–51.
- ERDŐS P. & RÉNYI A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, **5**, 17–61.
- FÜRNKRANZ J. (1997). Pruning algorithms for rule learning. *Mach. Learn.*, **27**(2), 139–172.
- GENT I. P. & WALSH T. (1994). Easy problems are sometimes hard. *Artificial Intelligence*, **70**(1–2), 335–345.
- GENT I. P. & WALSH T. (1999). 'beyond np : the qsat phase transition'. In *AAAI '99/IAAI '99*, p. 648–653.
- GOTTLÖB G. (1987). Subsumption and implication. *Information Processing Letters*, **24**(2), 109–111.
- GOTTLÖB G., LEONE N. & SCARCELLO F. (1997). On the complexity of some inductive logic programming problems. In *Proc. of the 7th ILP*, p. 17–32 : Springer.
- HAUSSLER D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning*, **4**(1), 7–40.
- HOGG T. & WILLIAMS C. (1994). The hardest constraint problems : A double phase transition. *Artificial Intelligence*, **69**(1–2), 359–377.
- MITCHELL T. M. (1982). Generalization as search. *Artificial Intelligence*, **18**, 203–226.
- MUGGLETON S. (1995). Inverse entailment and PROGOL. *New Generation Computing*, **13**, 245–286.
- PAGE D. & SRINIVASAN A. (2003). Iip : a short look back and a longer look forward. *J. Mach. Learn. Res.*, **4**, 415–430.
- PARKES A. J. (1997). Clustering at the phase transition. In *In Proc. of the 14th Nat. Conf. on AI*, p. 340–345 : AAAI Press / The MIT Press.
- PLOTKIN G. (1970). A note on inductive generalization. In *Machine Intelligence*, p. 153–163. Edinburgh University Press.
- RÜCKERT U., KRAMER S. & RAEDT L. D. (2002). Phase transitions and stochastic local search in k -term DNF learning. *Lecture Notes in Computer Science*, **2430**, 405–417.
- SELMAN B., LEVESQUE H. J. & MITCHELL D. (1992). A new method for solving hard satisfiability problems. In P. ROSENBLOOM & P. SZOLOVITS, Eds., *Proc. of the 10th AAAI*, p. 440–446, Menlo Park, California.
- SMITH B. M. (2001). Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *Theo.Com.Scie.*, **265**(1–2), 265–283.
- SMITH B. M. & DYER M. E. (1996). Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, **81**(1–2), 155–181.
- SRINIVASAN A. (1999). A learning engine for proposing hypotheses (Aleph). <http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph>.
- VALIANT L. G. (1984). A theory of the learnable. In *ACM Symposium on Theory of Computing (STOC '84)*, p. 436–445, Baltimore, USA : ACM Press.
- XU K. & LI W. (2006). Many hard examples in exact phase transitions. *Theor. Comput. Sci.*, **355**(3), 291–302.