

Empirical Study of Relational Learning Algorithms in the Phase Transition Framework

Erick Alphonse and Aomar Osmani

LIPN-CNRS UMR 7030, Université Paris 13, France
erick.alphonse@lipn.univ-paris13.fr,
aomar.osmani@lipn.univ-paris13.fr

Abstract. Relational Learning (RL) has aroused interest to fill the gap between efficient attribute-value learners and growing applications stored in multi-relational databases. However, current systems use general-purpose problem solvers that do not scale-up well. This is in contrast with the past decade of success in combinatorics communities where studies of random problems, in the phase transition framework, allowed to evaluate and develop better specialised algorithms able to solve real-world applications up to millions of variables. A number of studies have been proposed in RL, like the analysis of the phase transition of a NP-complete sub-problem, the subsumption test, but none has directly studied the phase transition of RL. As RL, in general, is Σ_2 – *hard*, we propose a first random problem generator, which exhibits the phase transition of its decision version, beyond NP. We study the learning cost of several learners on inherently easy and hard instances, and conclude on expected benefits of this new benchmarking tool for RL.

1 Introduction

Even though the expressiveness of (supervised) Relational Learning (RL), also known as Inductive Logic Programming (ILP), is attractive for many modern applications¹, such as life sciences, environmental sciences, engineering, natural language processing or arts (see also [1]), RL has to face the well-known trade-off between expressivity and efficiency.

From the efficiency perspective, one of the major obstacles is search efficiency, and several authors have acknowledged that a step forward would be in the design of novel search techniques (e.g. [2,1]). RL, as a sub-domain of symbolic learning, has been cast more than 25 years ago as search into a state space[3]: given a hypothesis space defined a priori, identified by its representation language, find a hypothesis consistent with the learning data. This seminal paper, relating symbolic learning to search in a state space, has enabled machine learning to integrate techniques from problem solving, operational research and combinatorics: greedy search in FOIL, beam search in ICL, breadth-first search in Aleph, 'A' search in PROGOL, IDA (Iterative-Deepening A) search in MIO

¹ <http://www-ai.ijs.si/~ilpnet2/apps/index.html>

to name a few systems². Besides very few exceptions, all systems are rooted in the generate-and-test paradigm [4] and therefore rely on *general-purpose* search strategies.

This approach has to be contrasted with the important progress, made during the past few years, in the performance of *specialised* SAT or CSP solvers, which can deal with problems' sizes that are orders of magnitude larger than those research communities would expect to solve only a decade ago. This progress has been driven by studies of randomly generated problem instances, in the phase transition framework, where hard to solve instances can be reliably generated, independently from the solver used. This framework relies on the conjecture that a phase transition in the probability of solubility of NP-complete problems can be exhibited along so-called order parameters and that the phase transition region contains the most difficult problem instances, those on which the computational complexity shows an exponential increase in the problem size [5,6,7]. 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” complexity resolution pattern, standard nowadays [8,9,10]. Since then, it has been strongly developed in many combinatorics domains and has changed the way search algorithms are empirically evaluated. This has led to new designs of search algorithms, from incomplete to complete solvers and from deterministic to randomised solvers (see e.g. [11]).

We think that RL, as a combinatorics field, must follow the same path and study learning as a decision problem in the phase transition (PT) framework in order to re-new and improve its algorithmic approach to answer the new challenges of modern applications. As the consistency problem is at the core of the Statistical Learning Theory, notably studied in the PAC framework (see [12,13] for details), the PT framework will be able to go beyond the worst-case complexity analysis and allow to study the behaviour of learning algorithms in the “average” or “typical” complexity case. Most importantly, it is also at the core of learners' optimisation algorithms, typical of real-world systems: optimisation procedures are obtained by adding branch-and-bound techniques, or treated as subsequent decision problems³. This *a fortiori* is true in RL where almost all noise-resistant learners are relaxation of this problem [15], therefore studying this problem will benefit search strategies for learning.

Machine Learning has known several developments in the study of phase transition phenomena since the early 90s. The first works were in neural networks [16,17,18,19] and studied the phase transition of the generalization error, where the number of examples are shown as order parameters. Other works studied the impact of the phase transition of the solubility probability of the NP-complete subsumption test, a sub-problem of Relational Learning, on the generalisation

² Relevant information on these systems can be found at <http://www-ai.ijs.si/~ilpnet2/systems>

³ This latter strategy, for instance, is used to solve pseudo-boolean constraints, an optimisation version of SAT: one of the best pseudo-boolean solver, Minisat+ is based on the SAT solver Minisat [14].

error [20] and heuristic search [21]. However, as surprising as it may seem, the phase transition framework strongly developed in combinatorics has almost not been imported to the realm of symbolic learning. To the best of our knowledge, the only work along this line has been done by Ruckert et al. [22] who exhibited the phase transition of the probability of solubility of a *learning problem*, the k -term DNF consistency problem, a well-known NP-complete problem [13], showing that the number of variables, the number of positive and negative examples were order parameters.

RL is arguably harder than attribute-value learning, like k -term DNF learning, which has been formalised by Gottlob et al. [23] who showed that the simple bounded ILP consistency problem, which will be discussed in later, is Σ_2 -complete. This is one class higher in the polynomial hierarchy than NP-complete (or Σ_1 -complete) problems. Some authors [24,25], have conjectured that a phase transition could be exhibited further up the polynomial hierarchy and therefore that this framework could be useful to other PSPACE problems. This was supported by results on planning and QBF-2 (Quantified Boolean Formulas with two alternating quantifiers, see also [26]).

In this paper, we show that this also holds true for the bounded ILP consistency problem and we present two main results. First, we exhibit the phase transition of the probability of solubility of the bounded ILP consistency problem, with the number of positive and negative examples as order parameters. Second, we exploit this framework as a benchmarking tool to evaluate for the first time learning algorithms on inherently hard and inherently easy problems, from a complexity point of view. We evaluate classical complete relational learners found in the learning systems Aleph [27], Progol [28] and Propal [29]: 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). We also use a lgg-based learner: Depth-First Bottom-up Data-Driven (DF-BDD).

The expected benefit is the same as for other combinatorics domains: (1) pointing out particularly 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 Relational Learning (see e.g. [2]) by studying their behaviour on inherent hard instances in the phase transition; and last (5) finding ways to generate hard instances to understand the problem's complexity and to provide challenging benchmark datasets [30,31].

The paper is organised in the following manner. Section 2 presents background information about relational learning and search strategies, as well as main results on the phase transition framework in combinatorics and the “easy-hard-easy” pattern. Section 3 describes the random problem instance generator, which has been already proposed to study the bounded ILP consistency problem in [21], although they did not study its PT as they did not investigate the relevant parameters for

this work. Next, section 4 shows that RL exhibits a phase transition of the probability of solubility and therefore can be used to reliably assess inherent complexity of learning problems. This is used in section 5 to evaluate and discuss the behaviour of popular complete learners on inherently hard and inherently easy problems. Finally, section 6 discusses the expected benefit of the development of the phase transition framework in RL and conclude on future works.

2 Background

2.1 Relational Learning

In machine learning, we are 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 named subsumption test \geq which relates \mathcal{L}_e and \mathcal{L}_h and partially order the hypotheses. After [3], (symbolic) learning is defined as search in \mathcal{L}_h . The problem, known as 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: $\forall e^- \in E^-, h \not\geq e^-$.

In this article, we study a typical instance of this problem in relational learning, known as the ILP consistency problem for function-free Horn clauses [23]: 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 l polynomial in $|E^+ \cup E^-|$, does there exist $h \in \mathcal{L}_h$ with no more than l literals such that h logically implies each element in E^+ and none element in E^- . In such hypothesis space, the logical implication is equivalent to θ -subsumption⁴ which is NP-complete and therefore decidable [32]. This result implies that relational learning is higher than attribute-value learning in the polynomial hierarchy. [23] proved that this problem is Σ_2^P -complete (or equivalently NP^{NP}): the search is NP-complete and it is guided by the subsumption test which is NP-complete.

A central idea in symbolic learning is the use of a generality partial order between hypotheses to guide the resolution of the consistency problem [3]. Mitchell 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 (see [33] for details). 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

⁴ Let C, D two clauses. C θ -subsumes D , noted $C \geq_\theta D$ iff there exists a substitution θ such that $C\theta \subseteq D$.

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 approach can therefore compensate for a poor evaluation function by using the learning data [29]. Moreover, some TDD strategies make the most of the negative instances in order to select informative negative examples, e.g. *near-misses* according to Winston. 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, noted δ 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 [34], who made the link between generalisation in learning and lowest-upper bound (lub) in lattice theory. Such an operator, also known as least-general generalisation (lgg) or most-specific generalisation (msg), has known several theoretic developments [35,12,36] but has been seldom used in learning systems, whose the best-known system is probably GOLEM [37].

2.2 Phase Transition and “Easy-Hard-Easy“ Pattern

Phase transition is a term originally used in physics to describe the changes of state of matter [38]. Even though originally referring to gas, liquid, or solid, within the framework of thermodynamics, it is used, by extension, to describe an abrupt and suddenly change in one of the parameters describing the state (in thermodynamic sense) of an arbitrary system. Thus, the transition from ferromagnetic to paramagnetic state, the emergence of super-fluidity, changing the type of crystal (with broken symmetry), or denaturation transition of DNA are characterised as phase transitions. A well-known example in everyday life is water, which is boiling (at normal pressure) at 100 °C. At the transition point there is a coexistence of liquid water and vapor (a “first order” phase transition). When plotting the density as a function of the temperature, a jump at the transition temperature can be observed. In this example, using physics terminology, density corresponds to the order parameter, whereas temperature corresponds to the external parameter of the phase transition. Notice that, in computer sciences, the term order parameter is used instead of external parameter, and we will keep this terminology in the following.

The phase transition of the probability of NP-complete decision problems, and beyond, certainly is the most studied phase transition framework in Computer Sciences as it has important consequences in practice on the average search complexity [5,39,7,8,9,40,41,25,42,43]. The so-called order parameters allow to wander from an under-constrained region, named the “yes” region, where this is almost surely a solution, to an over-constrained region, named the “no” region, where there is almost surely no solution. In between, the phase transition region contains the most difficult problem instances, those on which the computational complexity shows an exponential increase in the problem size, independently of the solver used [5,6,41,7]. The under-constrained problems from the “yes” region

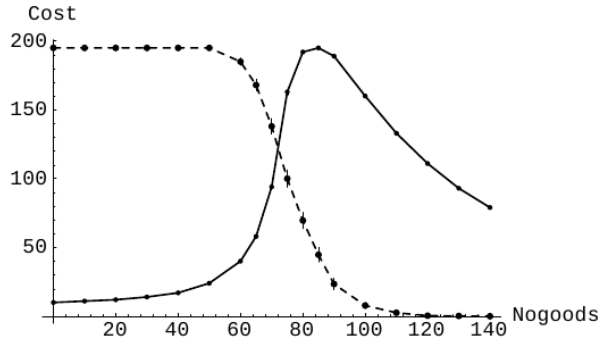


Fig. 1. Typical “easy-hard-easy” pattern [10]. Solid line shows median solution cost for dynamic backtracking and dashed line shows probability of a solution as a function of assignments to a pair of variables in random CSP that are considered to be inconsistent (number of no-goods).

appear to be easily soluble, as there are many solutions. This is the same for over-constrained 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 phase transition 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 phase transition region for certain types of algorithms [9,41,40]. 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. Such a typical pattern is shown in figure 1, from [10]. The search cost varies as a function of a given order parameter (referred as nogoods) for a class of problems, independent of particular search algorithms.

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 understand the complexity of the problem [43,30]. We present in the next section the model RLPG for the bounded ILP consistency problem that we will use to exhibit the PT.

3 Random Generator for Relational Learning Problem

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. The first four parameters are defined in order to ensure that a subsumption test between a hypothesis and an example during search encode a valid CSP problem following the model RB for random CSP [43]. We recall their meaning and

Table 1. Example of a random learning problem generated with RLPG, with no solution

$$\begin{array}{l} \perp | p0(A) \leftarrow p1(A, B, C), p2(A, B, D), p3(A, C, D) \\ + | p0(e1) \leftarrow p1(e1, b, c), p2(e1, c, d), p3(e1, e, f) \\ - | p0(e2) \leftarrow p1(e2, c, f), p2(e2, d, e), p3(e2, d, c) \end{array}$$

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

$$\begin{array}{l} \perp | p0(A) \leftarrow p1(A, B, C), p2(A, B, D), p3(A, C, D) \\ + | p0(e1) \leftarrow p1(e1, b, c), p2(e1, d, e), p3(e1, e, e) \\ - | p0(e2) \leftarrow p1(e2, b, b), p2(e2, e, e), p3(e2, e, c) \end{array}$$

focus on the last two parameters, which were not studied before and which will be shown to be order parameters of the phase transition of the ILP consistency problem.

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

4 Number of Positive and Negative Examples as Order Parameters

In this section, we study the effect of the number of positive and negative examples on the solubility probability of the ILP consistency problem. If we refer to the previous section, $RLPG$ is parametrised with 6 parameters but we only

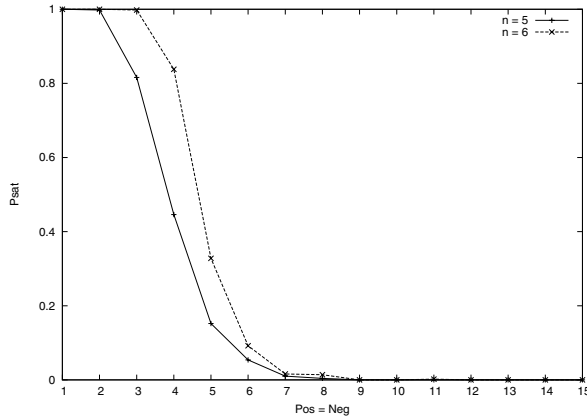


Fig. 2. Probability of satisfiability according to the number of learning examples (= Pos = Neg), with $n = 5$ and $n = 6$

study the last two, Pos and Neg , as the effect of the other parameters have already been studied in [21] for constant number of positive and negative examples. Here, we focus on few settings for these parameters, with $k = 2$, $n = 5$ and $n = 6$, to study different problem sizes, $\alpha = 1.4$ and $N = 10$. The choice of these parameters ensures that we do not generate trivially insoluble problems (see [25] for details), but also various experiments, not shown here, indicated that they were representative of the phase transition behaviour of the ILP consistency problem. In all experiments below, statistics were computed from a sample of 500 learning problems, solved with a complete learner.

We start by varying both Pos and Neg . Figure 2 shows the solubility probability of the ILP consistency problem when $Pos = Neg$ are varied from 1 to 15, for $n = 5$ and $n = 6$. As we can see, when the number of examples is small, there is almost surely a consistent hypothesis, and when the number is large, it is almost surely impossible to find a consistent hypothesis. The cross-over point, where the probability of solubility is about 0.5, is around 4 for $n = 5$ and 5 with $n = 6$. It is not surprising that it increases with bigger problems. For $n = 5$, the hypothesis space size is 2^{10} and 2^{15} for $n = 6$. We could not conduct experiments for larger values of n as the hypothesis space grows too fast in *RLPG*. For instance, $n = 7$ sets a hypothesis space of size 2^{21} , which cannot be handled by our complete solver. In the future, it would be interesting to modify *RLPG* to specify the size of the bottom clause and then draw the number of variables accordingly.

We study now the phase transition along the number of positive examples, for constant values of Neg . Figures 3 and 4 show the phase transition when Pos varies from 1 to 25, for $n = 5$ and $n = 6$ respectively. With no positive examples, the bottom element of the search space is solution, but as Pos increases, complete hypotheses get more general and eventually subsume a negative example. The transition becomes sharper as Neg increases, which is not surprising as the subset of correct hypotheses shrinks as Neg increases. The second order parameter is

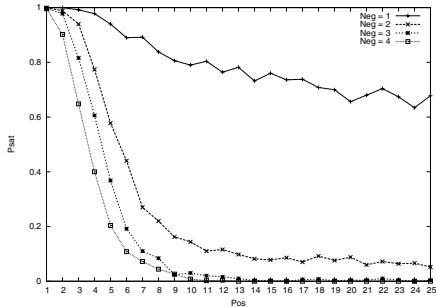


Fig. 3. Probability of satisfiability according to the number of positive examples with $n = 5$, for $Neg = 1, 2, 3, 4$

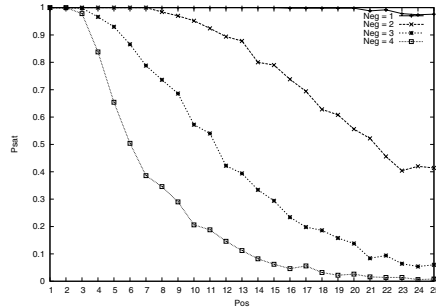


Fig. 4. Probability of satisfiability according to the number of positive examples with $n = 6$, for $Neg = 1, 2, 3, 4$

the number of negative examples Neg but it is not shown here because of the space requirements. The plots essentially exhibit the same profile.

5 Evaluating Complete Learners

We evaluate complete learners representative of the search strategies described in section 2.1. 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 as 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 [27,28] 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, and we refer to [36] for implementation details. Briefly, starting from the bottom element, the algorithm generalises the current hypothesis to subsume each positive example in turn, until it outputs a consistent hypothesis,

or until it proves that no correct hypothesis subsumes all positive examples. Note that as the hypothesis space is not a lattice, which is the case here under θ -subsumption as the hypothesis space is finite, the lgg operator outputs all possible generalisations on subsequent backtracks.

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 (see section 2.2). As the consistency problem in RL is Σ_2^P -complete (see section 2.1), 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 [29] 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 5 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 solubility probability is no longer 1 (when both the number of positive and negative examples are greater than 3). This 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 increases, 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 of those extra examples.

For various percentiles, figure 6 shows that BF-TGT, as a non-informed search strategy, is costly very early in the “yes” region. However, after the PT, A-TGT

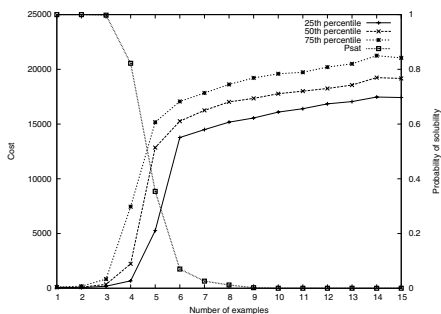


Fig. 5. Backtracking cost using A-TGT strategy for various percentiles and probability of solubility, for $n = 6$

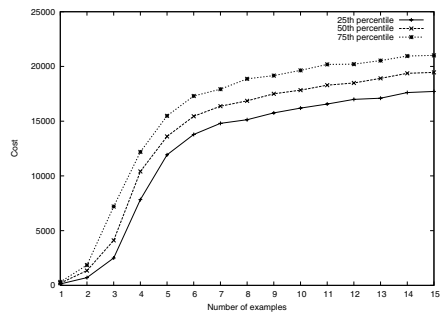


Fig. 6. Backtracking cost using BF-TGT strategy for various percentiles, for $n = 6$

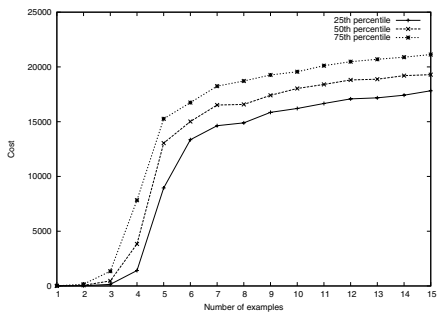


Fig. 7. Backtracking cost using DF-TGT strategy for various percentiles, for $n = 6$

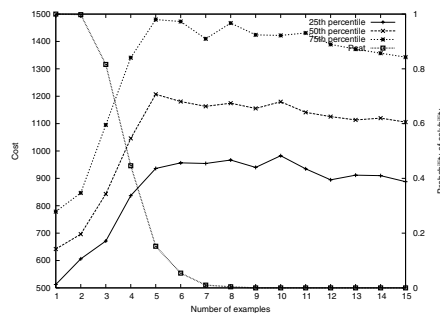


Fig. 8. Backtracking cost using DF-BDD strategy for various percentiles for $n = 5$

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, to keep 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 8, we show results for the data-driven search, DF-BDD, first on problems of 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

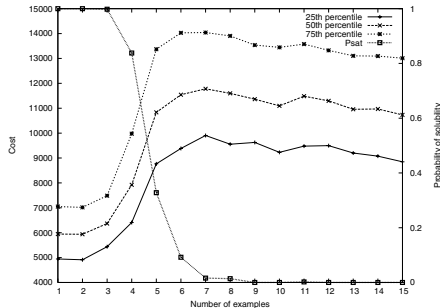


Fig. 9. Backtracking cost using DF-BDD strategy for various percentiles, for $n = 6$

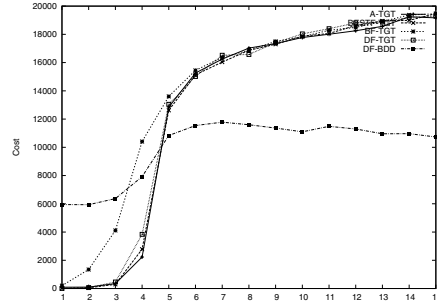


Fig. 10. Median (50th percentile) backtracking cost for A-TGT, DF-TGT, BF-TGT, BESTF-TGT and DF-BDD strategies, for $n = 6$

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, although some improvements can be done.

Figure 9 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.

Figure 10 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 many 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 increases.

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, such as for BF-TDD (see above). We then complete it by plotting the resolution time of BF-TDD in figure 11. 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

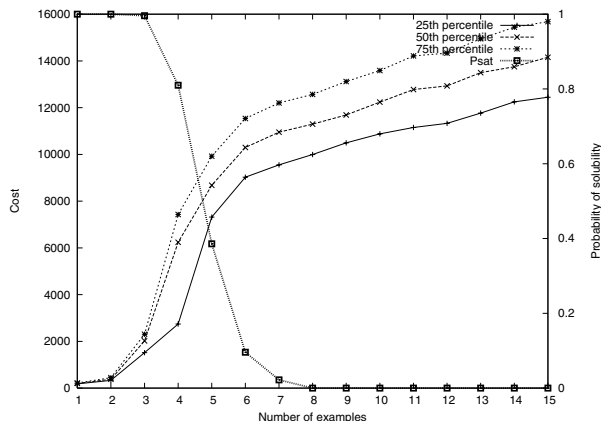


Fig. 11. Time cost using BF-TDD strategy for various percentiles, for $n = 6$

the quality of the near-miss and the reduction of the search space has to be evaluated.

6 Conclusion

Although Relational Learning has been cast, more than 25 years ago, as search, it has known very few developments from the search strategy point of view and most learners rely on general-purpose solvers. This is a strong limitation to its applicability on many modern applications, as it prevents RL to scale-up well. On the other hand, important progress has been made in other combinatorics communities, such as SAT and CSP, in the development of efficient specialised solvers, through the study of random NP-complete problem generators in the phase transition framework. RL has a higher complexity, being Σ_2 -hard in the general case. However, we argue that this framework will benefit RL, based on the conjecture that the phase transition can be exhibited further up the polynomial hierarchy. We show that this conjecture holds true with the bounded ILP consistency problem, a Σ_2 -complete problem, representative of RL problems. We propose a first simple random generator that exhibits a phase transition in the problem’s solubility, with the number of positive and negative examples as order parameters. 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 tend 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 phase transition 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. It is clear that this study does not take into account all the dimensions of learning problems: optimization instead of consistency, presence of noise, etc. However, the first idea is to understand the complexity landscape of learning problems and to define order parameters to control this complexity. The most important advantage of the proposed approach to evaluate algorithm complexity is that contrary to results obtained directly on real-world applications, which hardly transpose when the size of the problems change of scale, the phenomena observed with few variables are the same as those observed with thousands of variables. 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. For instance, we plan to investigate lgg-based learning algorithms, which have been seldom used in learning systems but seem to be efficient solvers.

References

1. Domingos, P.: Prospects and challenges for multi-relational data mining. *SIGKDD Explorations* 5(1), 80–83 (2003)
2. Page, D., Srinivasan, A.: Ilp: a short look back and a longer look forward. *J. Mach. Learn. Res.* 4, 415–430 (2003)
3. Mitchell, T.M.: Generalization as search. *Artificial Intelligence* 18, 203–226 (1982)
4. Newell, A., Simon, H.A.: *Human Problem Solving*. Prentice-Hall, Englewood Cliffs (1972)
5. Cheeseman, P., Kanefsky, B., Taylor, W.: Where the really hard problems are. In: *Proc. of the 12th International Joint Conference on Artificial Intelligence*, pp. 331–340. Morgan Kaufmann, San Francisco (1991)
6. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distribution of SAT problems. In: *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 440–446 (1992)
7. Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L.: Determining computational complexity from characteristic ‘phase transitions’. *Nature* 400, 133–137 (1999)
8. Smith, B.M., Dyer, M.E.: Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence* 81(1-2), 155–181 (1996)
9. Hogg, T., Williams, C.: The hardest constraint problems: A double phase transition. *Artificial Intelligence* 69(1-2), 359–377 (1994)
10. Mammen, D.L., Hogg, T.: A new look at the easy-hard-easy pattern of combinatorial search difficulty. *Journal of Artificial Intelligence Research* 7, 47–66 (1997)
11. Gomes, C., Heny Kautz, A.S., Selman, B.: Satisfiability solvers. In: *Handbook of Knowledge Representation* (2007)
12. Haussler, D.: Learning conjunctive concepts in structural domains. *Machine Learning* 4(1), 7–40 (1989)
13. Kearns, M.J., Vazirani, U.V.: *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge (1994)
14. En, N., Srensson, N.: Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation* 2, 1–26 (2006)

15. Fürnkranz, J.: Pruning algorithms for rule learning. *Mach. Learn.* 27(2), 139–172 (1997)
16. Shim, G.M., Choi, M.Y., Kim, D.: Phase transitions in a dynamic model of neural networks. *Physics Review A* 43, 1079–1089 (1991)
17. Nagashino, H., Kelso, J.A.: Phase transitions in oscillatory neural networks. In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 1710, pp. 279–287 (1992)
18. Schottky, B.: Phase transitions in the generalization behaviour of multilayer neural networks. *Journal of Physics A Mathematical General* 28, 4515–4531 (1995)
19. Biehl, M., Ahr, M., Schusser, E.: Statistical physics of learning: Phase transitions in multilayered neural networks. *Advances in Solid State Physics* 40/2000, 819–826 (2000)
20. Botta, M., Giordana, A., Saitta, L., Sebag, M.: Relational learning as search in a critical region. *Journal of Machine Learning Research* 4, 431–463 (2003)
21. Alphonse, E., Osmani, A.: A model to study phase transition and plateaus in relational learning. In: *Proc. of Conf. on Inductive Logic Programming*, pp. 6–23 (2008)
22. Rückert, U., Kramer, S., Raedt, L.D.: Phase transitions and stochastic local search in k -term DNF learning. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) *ECML 2002. LNCS (LNAI)*, vol. 2430, pp. 405–417. Springer, Heidelberg (2002)
23. Gottlob, G., Leone, N., Scarcello, F.: On the complexity of some inductive logic programming problems. In: Džeroski, S., Lavrač, N. (eds.) *ILP 1997. LNCS*, vol. 1297, pp. 17–32. Springer, Heidelberg (1997)
24. Bylander, T.: A probabilistic analysis of propositional strips planning. *Artificial Intelligence* 81(1-2), 241–271 (1996)
25. Gent, I.P., Walsh, T.: Beyond NP: the QSAT phase transition. In: *AAAI 1999/IAAI 1999: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference*, pp. 648–653 (1999)
26. Chen, H., Interian, Y.: A model for generating random quantified boolean formulas. In: Kaelbling, L.P., Saffioti, A. (eds.) *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 66–71. Professional Book Center (2005)
27. Srinivasan, A.: A learning engine for proposing hypotheses (Aleph) (1999), <http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph>
28. Muggleton, S.: Inverse entailment and PROGOL. *New Generation Computing* 13, 245–286 (1995)
29. Alphonse, É., Rouveirol, C.: Extension of the top-down data-driven strategy to ILP. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) *ILP 2006. LNCS (LNAI)*, vol. 4455, pp. 49–63. Springer, Heidelberg (2007)
30. Cook, S.A., Mitchell, D.G.: Finding hard instances of the satisfiability problem: A survey. In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 1–17. American Mathematical Society (1997)
31. Xu, K., Li, W.: Many hard examples in exact phase transitions. *Theor. Comput. Sci.* 355(3), 291–302 (2006)
32. Gottlob, G.: Subsumption and implication. *Information Processing Letters* 24(2), 109–111 (1987)
33. Fürnkranz, J.: A pathology of bottom-up hill-climbing in inductive rule learning. In: Cesa-Bianchi, N., Numao, M., Reischuk, R. (eds.) *ALT 2002. LNCS (LNAI)*, vol. 2533, pp. 263–277. Springer, Heidelberg (2002)

34. Plotkin, G.: A note on inductive generalization. In: *Machine Intelligence*, pp. 153–163. Edinburgh University Press (1970)
35. Valiant, L.G.: A theory of the learnable. In: *ACM Symposium on Theory of Computing (STOC 1984)*, Baltimore, USA, pp. 436–445. ACM Press, New York (1984)
36. Kietz, J.U.: A comparative study of structural most specific generalisations used in machine learning. In: *Proc. Third Workshop on ILP*, pp. 149–164 (1993)
37. Muggleton, S., Feng, C.: Efficient induction of logic programs. In: *Proc. of the 1st Conference on Algorithmic Learning Theory*, Ohmsma, Tokyo, Japan, pp. 368–381 (1990)
38. Paskal, Y.I.: The meaning of the terms phase and phase transition. *Russian Physics Journal* 31(8), 664–666 (1988)
39. Selman, B., Levesque, H.J., Mitchell, D.: A new method for solving hard satisfiability problems. In: *Proc. of the Tenth National Conference on Artificial Intelligence*, Menlo Park, California, pp. 440–446 (1992)
40. Gent, I.P., Walsh, T.: Easy problems are sometimes hard. *Artificial Intelligence* 70(1–2), 335–345 (1994)
41. Davenport, A.: A comparison of complete and incomplete algorithms in the easy and hard regions. In: *Workshop on Studying and Solving Really Hard Problems*, CP 1995, pp. 43–51 (1995)
42. Smith, B.M.: Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *Theoretical Computer Science* 265(1–2), 265–283 (2001)
43. Xu, K., Boussemart, F., Hemery, F., Lecoutre, C.: Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artif. Intell.* 171(8–9), 514–534 (2007)