On the Two Phase Transitions of Relational Learning

Erick ALPHONSE, Aomar OSMANI

LIPN-CNRS UMR 7030, Université Paris 13, France

Abstract. It is known recently that relational learning is Σ_2 hard. This level of complexity is due to the entanglement of two NP problems: the covering test problem and the search problem in hytothesis space. This paper explains the phase transition and the complexity of relational learning by the superposition of two phase transitions associed to each of these problems. The relational learning is particularly suited to applications where data are described in a database (most corporate data), but it suffers from a problem of efficiency for the scaling-up. The objective of this work is to take another step toward understanding the sources of complexity of relational learning, drawing on work in combinatorics and statistical physics analysis averaged the intrinsic complexity of problems -rather than focusing only on the considered classical complexity in the worst case in order to propose new algorithms supporting the scaling.

1 Introduction

Most available data are stored as computer databases. Thus, it is natural to want to develop learning techniques with the same expressive level than manipulated data in order to exploit all the information given by the relationships between data. However, we note that most machine learning algorithms, using numerical representation or symbolic one, deal with data represented as vector of attributes. This representation has the advantage of having efficient algorithms. Among the drawbacks, these languages capture only part of the information provided by the data.

The relational learning (RL) or inductive logic programming is one of the best candidates to exploit corporate data since most of them are described in relational databases. However, RL has to face the well-known trade-off between expressivity and efficiency. Indeed, the relational learning has many difficulties when it addresses real-world applications [12, 31]. The benefits of expressiveness is offset by the efficiency of the algorithms. Suggest new algorithms requires, from our point of view, the understanding of the sources of complexity. The classical analytical framework of algorithms focusing on worst-case bounds gives interesting results, but they are minimally useful for the study of real cases.

The average complexity analytical framework [13, 8, 41, 30], despite the real world simplifying assumptions, provides a more relevant results about the behavior of algorithms on real applications. This framework is the most used one in combinatorics since twenty years [8, 23, 28, 15, 10, 42, 47] and it is behind the

best current algorithms [21, 7] which gave rise to type algorithms that can handle problems with millions of variables.

The main conjecture about the average complexity in combinatorial analysis and in machine learning is that the hypothesis space is structured by a control parameter which separates the problems space into three distinct regions: under-constrained region where almost all hypothesis are solutions, an over-constrained region where almost hypothesis are not solution and a region in between -a phase transition region- where almost all hard problems are concentrated (see Figure 1).

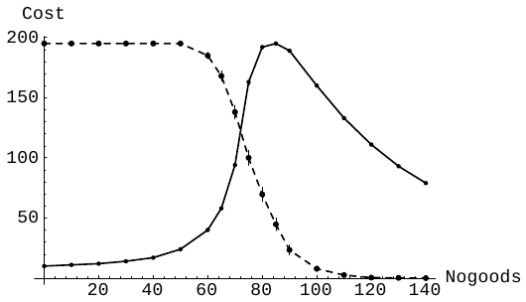

Fig. 1. This figure shows probability of a solution as a function of assignments to a pair of variables in random CSP that are considered to be inconsistent(number of no-goods)[26]. Dashed line shows the phase transition. The control parameter structure the problem space into three parts: problems with a lot of solutions, problems with few (or without) solutions and in between. Solid line shows median solution cost for dynamic backtracking and shows that hard problems are concentrated arount the phase transition.

In this paper, we continue the work arround the study of phase transition (PT) in machine learning started in the early 90s. Early work deals with neural networks[11, 41, 30], where the main problem concerns the phase transition of the generalisation error: the number of examples is shown to be a control parameter of the phase transition [5]. Phase transition receives more attention in machine learning since ten years but there are only few results. In symbolic machine learning, the first work is due to [17]. The authors propose a phase transition like approch to study the θ-subsumption test efficiency in inductive logique programming. In 2002, [37] propose the first study of the phase transition of a k-terms DNF learning problem. In 2006, using the CSP background of phase transition studies, [1] study the covering test of relational learning within the PT framework and propose the first problem generator to study this problem. In 2009, [**?**] propose a generator (RLPG) to study the PT of relation learning problems. Using this generator, they show that most of existing relational learning algorithms exhibite profiles far from the expected theoretical ones: most of existing algorithms are NP even for easy learning problems. RLPG and experimental results obtained using most existing relational learning algorithms give a first key to understand the complexity landscape of the relational learning

problem. The work, presented in this paper, goes beyond this first analysis and exhibits the phase transitions of the two independants NP problems concerned with the RL. Within this study, it will be possible, for example, to exhibit an easy to solve classes of problems for what the θ-subsumption test is NP-hard. In the other hand, using the control parameter of the PT search problem, it will be possible to find an easy problems to solve even in the plateau regions (when the evaluation function used to prioritize nodes in refinement graph is constant).

We investigated in this paper, the double phase transition of relational learning problem: the phase transition of the covering test and the phase transition of the search in the hypothesis space. We illustrate the outline of the complexity generated by each of them and we show that the independence of these two phase transitions can provide a simple control parameter characterizing the phase transition of the entire learning task. This characterization allows a better understanding of the source of learning difficulty and would guide the search for better algorithms for solving the problem of relational learning adapted to real applications.

The paper is organised as follow: section **??** presents background information about relational learning and search strategies. Section 1.2 defines the phase transition framework and shows its interest to study machine learning problems. Section **??** presents the phase transition of the covering test problem and describes the random problem instance generator. Section **??** presents the phase transition of the search problem. Section **??** shows the entanglement of two phase transitions, discusses the landscape complexity of the learning problem in relational languages and then propose the control parameter of the learning problem. Finally, section 6 discusses the expected benefit of the development of the phase transition framework in RL and conclude on future works.

1.1 Machine Learning Context

Without loss of generality, we consider, in our work, the machine learning problem as defined in [27] : 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. The learning problem, defined as search in $\mathcal{L}_h$, is to find a hypothesis $h \in \mathcal{L}_h$ such that h is consistent with the data. A given hypothesis h is consistent if and only if it is:

- complete: $\forall e^+ \in E^+, h \geq e^+$ and
- correct: $\forall e^- \in E^-, h \not\geq e^-$.

We are interested on relational learning for function-free Horn clauses [20].

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 $|E^+ \cup E^-|$,

Find $h \in \mathcal{L}_h$ with no more than l literals such that h logically implies each element in E^+ and none element in E^-.

In this space, the logical implication is equivalent to θ-subsumption which is NP-complete [19] and therefore the RL learning problem is NP^{NP}-complete [20]. Search strategies in symbolic learning follow the Mitchell definition [27]. There exists two main stategies: generate-and-test (GT) and data-driven (DD). And in each strategy, algorithms use top-down approach, buttom-up one or any combination of them.

But since twenty years, 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 [14] 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. Therefore, GT algorithms have to deal with many refinements that are not relevant with respect to the discrimination task. Formally, the refinement operator is defined as a binary operator: Let $h \in \mathcal{L}_h, e^- \in E^- : \rho(h, e^-) = \{h' \in \mathcal{L}_h | h \geq h' \; and \; h' \not\geq e^-\}$.

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 $h \in \mathcal{L}_h, e^+ \in E^+ : \delta(h, e^+) = \{h' \in \mathcal{L}_h | h \leq h' \; and \; h' \geq e^+\}$.
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) has known several theoretic developments [46, 22, 24] but has been seldom used in learning systems.

In all these strategies the search is NP-complete and it is guided by the subsumption test which is NP-complete. What we try to do, in this work, is to exhibit the phase transition of the search and the phase transition of the subsumption test. Even, the RL learning is Σ_2-complete, we will show that the everage hardness of RL problems follows a unique control parameter computed by the combination of the both control parameters. This interpretation gives a new light on understanding the complexity of relational learning.

1.2 Phase transition

Phase transition is a term originally used in physics to describe the changes of state of matter [33]. 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.

Surprising as it may be, the emergence of a phase transition is not limited to physical systems: it seems to be a rather ubiquitous phenomenon in biological networks, genetics, neural networks and in combinatorial problems. As for the latter, even a precise parallel can be established between them and physical systems composed of very large numbers of particles. In a combinatorial problem the phase transition concerns the behavior of the algorithms used to solve it. In particular, the phase transition acts as a border between regions (phases) in which algorithms behave in drastically different ways; moreover, a large increase in the algorithm's computational complexity in correspondence to the transition is usually observed.

Since two decades many works have been published on phase transitions in computer science, but, as pointed out in [32], it seems that this area started with the remarkable observation in [13] that thresholds in properties such as connectivity emerge in large random graphs. However, even if several studies appeared, concerning the link between statistical physics and graph partitioning [48], characterisation of hard instances in CNF [35], finding optimal paths in trees [36], graph colouring [45], the phase transition has empirically been discovered in combinatorial search problems only in the early 90s [8, 40]. The discovery of phase transition in NP-complete satisfiability problems [25] has raised a lot of interest in the artificial intelligence community. Constraint satisfaction problem (CSP) and propositional satisfiability problem (SAT) communities are the communities where the phase transition framework is probably the most studied [8, 40, 28, 43, 23, 15]

CSPs and k-SAT ($k \geq 3$) are NP-complete problems, as well as many other combinatorial ones. The notion of computational complexity, based on worst-case analysis, used to build up the classical polynomial hierarchy, may not be very useful in practice, In fact, many instances of NP-complete problems are actually easy to solve [8]. The phase transition framework offers a way to study problems in the "typical" complexity case, more representative of real-world applications, and it also contributes to the design of efficient algorithms. A key notion, in the study of phase transitions, is that of *problem ensemble*, which grounds the link between statistical physical systems and combinatorial problems. All the properties derived from the phase transition framework are valid for sets of *random* problems, whose generative model is precisely specified. Then, an important aspect to be discussed, inside the framework, is what model to use, and how this model can possibly cover problems faced in the real world.

In Machine Learning, two directions of research are interested in phase transitions: the first one investigates the emergence of phase transitions of the generalization error (essentially in multilayer neural networks) [41, 30, 39, 6]. Most of these works are interested in the phase transition of the generalization error, where the number of examples are shown as order parameters. For instance, [6], using a model inspired from statistical physics of disordered systems, exhibited a discontinuous dependence between the generalization error and the number of examples. The second one, instead, involves discrete spaces (symbolic learning), where learning is, in its essence, a combinatorial problem. In 2000, [18] studied

the phase transition like phenomena of the θ-subsumption test in ILP. This subproblem is known to be NP-complete and they showed that its phase transition could be exhibited with relevant parameters for learning: the size of the hypothesis and the number of constants in the example language. They proposed to use a random problem generator inspired from model B, a well-known generator in CSP [43]. Ruckert et al. [38] proposed to study k-term DNF learning in the phase transition framework in combinatorics. This NP-complete problem exhibits a phase transition in its solubility depending on the number of variables. They also characterized this phase transition as a function of k and the number of positive and negative examples. [4] studies the connection between the phase transition of the covering test and occurrences of plateaus during heuristic search. Specifically, they propose in [3] a consistency problem generator for relational learning, RLPG, that guarantees the existence of plateaus during search, based on model RB proposed for CSP[47]. Using RB properties, it is proved that the current hypothesis' size evaluated during learning -the control parameter of heuristic search in learning- is an order parameter of the phase transition of the subsumption test. It is proposed as a benchmarking tool to evaluate learning search strategies on plateaus. Finally, [**?**] propose an empiritical study of relational learning algorithms in the phase transition framework.

With the past decade, the use of phase transition framework in combinatorics communities gives a real success to solve real-world applications up to millions of variables. This phenomenon, viewed from the angle of learning complexity, both in terms of generalization error and time complexity, may bring new perspectives to learning algorithms. The interest of this framework, in this paper, 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 [47, 9].

2 Phase Transition of the Covering Test

[4] summarizes a decade of work an the covering test in machine learning using phase transition framework and concludes that the localisation of the target concept with the respect to this phase transition alone is not a reliable indication of the learning difficulty as previously thought.

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 [47]. We recall their meaning and 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 $N.\binom{n}{k}$. 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

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

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

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

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

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

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 study the last two, Pos and Neg, as the effect of the other parameters have already been studied in [3] 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 [16] 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.

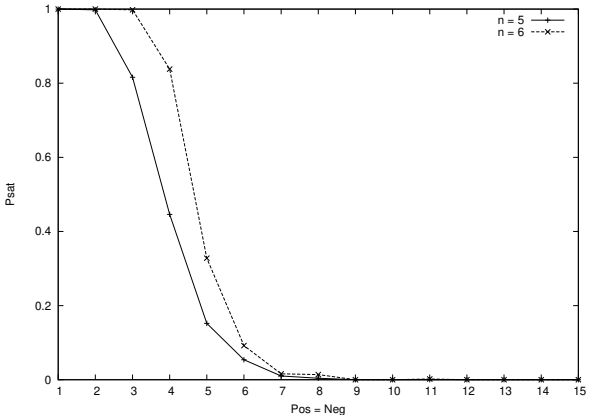

Fig. 2. Probability of satisfiability according to the number of learning examples (= Pos = Neg), with $n = 5$ and $n = 6$

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.

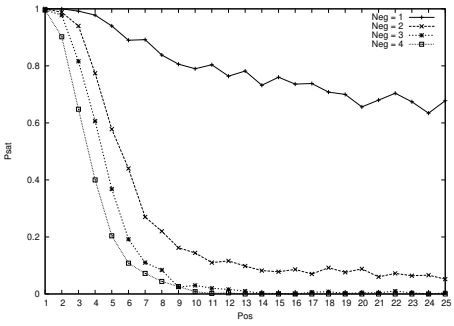

Fig. 3. Probability of satisfiability according to the number of positive examples with $n = 5$, for $Neg = 1, 2, 3, 4$

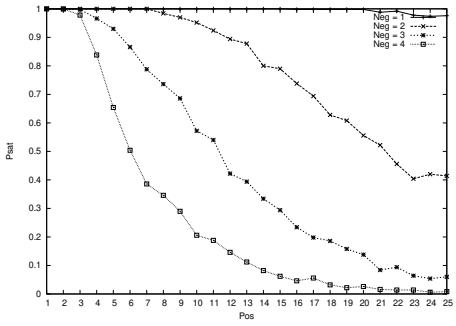

Fig. 4. Probability of satisfiability according to the number of positive examples with $n = 6$, for $Neg = 1, 2, 3, 4$

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 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 1.1. 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 [44, 29] 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 [24] 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 1.2). As the consistency problem in RL is Σ_2^p-complete (see section 1.1), 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 [2] 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.

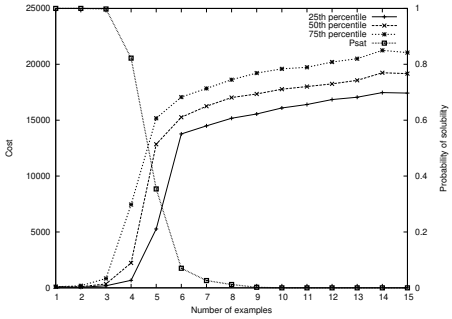

Fig. 5. Backtracking cost using A-TGT strategy for various percentiles and probability of solubility, for $n = 6$.

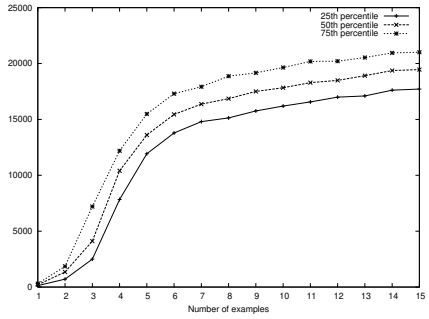

Fig. 6. Backtracking cost using BF-TGT strategy for various percentiles, for $n = 6$.

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 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.

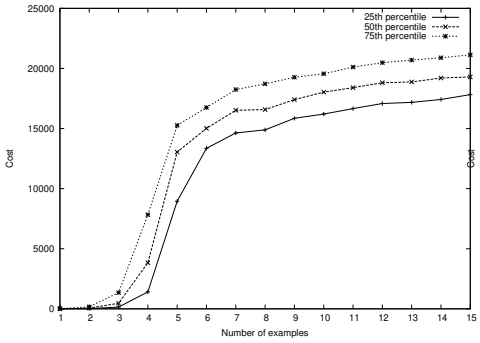

Fig. 7. Backtracking cost using DF-TGT strategy for various percentiles, for $n = 6$.

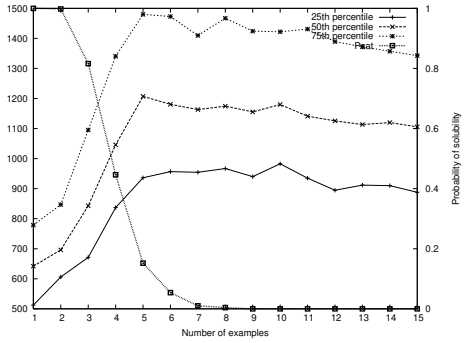

Fig. 8. Backtracking cost using DF-BDD strategy for various percentiles for $n = 5$.

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 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.

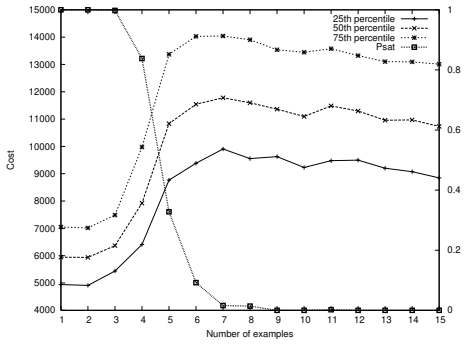

Fig. 9. Backtracking cost using DF-BDD strategy for various percentiles, for $n = 6$.

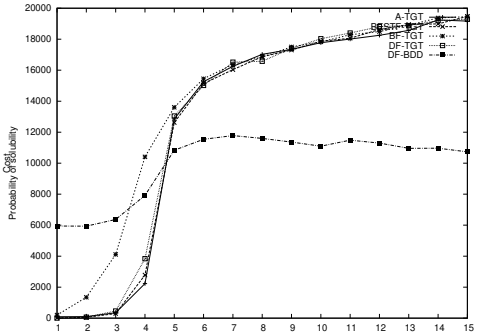

Fig. 10. Median (50th percentile) backtracking cost for A-TGT, DF-TGT, BF-TGT, BESTF-TGT and DF-BDD strategies, for $n = 6$.

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 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

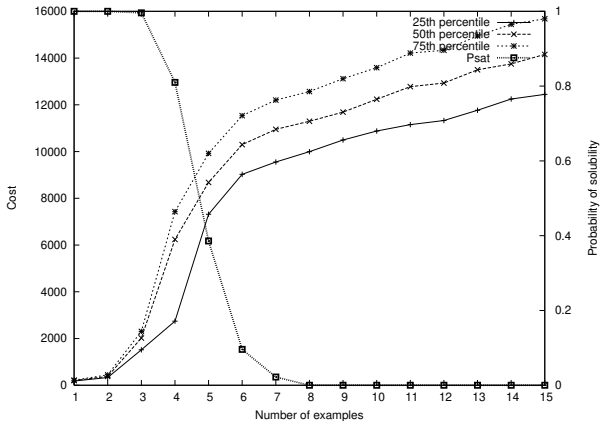

Fig. 11. Time cost using BF-TDD strategy for various percentiles, for $n = 6$.

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. E. Alphonse and A. Osmani. On the connection between the phase transition of the covering test and the learning success rate. In *Proc. of Conf. on Inductive Logic Programming*, 2006.
2. E. Alphonse and C. Rouveirol. Extension of the top-down data-driven strategy to ILP. In *Proc. of Conf. on Inductive Logic Programming*, Santiago de Compostela, Spain, 2006. Springer Verlag.
3. Erick Alphonse and Aomar Osmani. A model to study phase transition and plateaus in relational learning. In *Proc. of Conf. on Inductive Logic Programming*, pages 6–23, 2008.
4. Erick Alphonse and Aomar Osmani. On the connection between the phase transition of the covering test and the learning success rate. *Machine Learning Journal*, 70(2-3):135–150, 2008.
5. Michael Biehl, Martin Ahr, and Enno Schlösser. Statistical physics of learning: Phase transitions in multilayered neural networks. *Advances in Solid State Physics*, 40/2000:819–826, 2000.
6. Michael Biehl, Martin Ahr, and Enno Schlösser. Statistical physics of learning: Phase transitions in multilayered neural networks. *Advances in Solid State Physics*, 40/2000:819–826, 2000.
7. Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: an algorithm for satisfiability. *CoRR*, cs.CC/0212002, 2002.
8. P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. In *Proc. of the 12th International Joint Conference on Artificial Intelligence*, pages 331–340. Morgan Kaufmann, 1991.
9. Stephen A. Cook and David G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–17. American Mathematical Society, 1997.
10. A. Davenport. A comparison of complete and incomplete algorithms in the easy and hard regions. In *Workshop on Studying and Solving Really Hard Problems, CP-95*, pages 43–51, 1995.
11. Jacques Demongeot, Christelle Jquel, and Sylvain SenBoundary conditions and phase transitions in neural networks. theoretical results. *Neural Networks*, 21(7):971 – 979, 2008.
12. Pedro Domingos. Prospects and challenges for multi-relational data mining. *SIGKDD Explor. Newsl.*, 5(1):80–83, 2003.
13. P. Erdös and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.
14. J. Fürnkranz. A pathology of bottom-up hill-climbing in inductive rule learning. In *Proc. of the 13th International Conference on Algorithmic Learning Theory*, Germany, 2002. Springer-Verlag.
15. Ian P. Gent and Toby Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, 70(1–2):335–345, 1994.
16. Ian P. Gent and Toby Walsh. Beyond np: the qsat phase transition. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 648–653, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.
17. A. Giordana, M. Botta, and L. Saitta. An experimental study of phase transitions in matching. In Dean Thomas, editor, *Proc. of the 16th International Joint*

Conference on Artificial Intelligence (IJCAI-99-Vol2), pages 1198–1203. Morgan Kaufmann Publishers, July31–August6 1999.
18. A. Giordana and L. Saitta. Phase transitions in learning relations. *Machine Learning*, 41:217–25, 2000.
19. G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.
20. G. Gottlob, N. Leone, and F. Scarcello. On the complexity of some inductive logic programming problems. In Nada Lavrač and Sašo Džeroski, editors, *Proc. of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *LNAI*, pages 17–32, Berlin, 1997. Springer.
21. Shai Haim and Toby Walsh. Restart strategy selection using machine learning techniques. *CoRR*, abs/0907.5032, 2009.
22. D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4(1):7–40, 1989.
23. T. Hogg and C.P. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69(1–2):359–377, 1994.
24. J.-U. Kietz. A comparative study of structural most specific generalisations used in machine learning. In *Proc. Third Workshop on ILP*, pages 149–164, 1993.
25. Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.
26. Dorothy L. Mammen and Tad Hogg. A new look at the easy-hard-easy pattern of combinatorial search difficulty. *Journal of Artificial Intelligence Research*, 7:47–66, 1997.
27. Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
28. Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic 'phase transitions. *Nature*, 400:133–137, 1999.
29. S. Muggleton. Inverse entailment and PROGOL. *New Generation Computing*, 13:245–286, 1995.
30. H. Nagashino and J. A. Kelso. Phase transitions in oscillatory neural networks. In D. W. Ruck, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 1710 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 279–287, July 1992.
31. David Page and Ashwin Srinivasan. Ilp: a short look back and a longer look forward. *J. Mach. Learn. Res.*, 4:415–430, 2003.
32. Andrew J. Parkes. Clustering at the phase transition. In *In Proc. of the 14th Nat. Conf. on AI*, pages 340–345. AAAI Press / The MIT Press, 1997.
33. Yu. I. Paskal. The meaning of the terms "phase" and "phase transition. *Russian Physics Journal*, 31(8):664–666, 1988.
34. G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, pages 153–163. Edinburgh University Press, 1970.
35. Jr.P.W. Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 21(1-2):117–133, 1983.
36. J. Pearl R.M. Karp. Searching for an optimal path in a tree with random costs. *Artificial Intelligence*, 21:99–117, 1983.
37. U. Rückert. Machine learning in the phase transitions framework. *Thesis*, 2430:1–60, 2002.
38. U. Rückert, S. Kramer, and L. De Raedt. Phase transitions and stochastic local search in k-term DNF learning. *LNCS*, 2430:405–417, 2002.

39. B. Schottky. Phase transitions in the generalization behaviour of multilayer neural networks. *Journal of Physics A Mathematical General*, 28:4515–4531, August 1995.
40. Bart Selman, Hector J. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California, 1992.
41. G. M. Shim, M. Y. Choi, and D. Kim. Phase transitions in a dynamic model of neural networks. *Physics Review A*, 43:1079–1089, January 1991.
42. Barbara M. Smith. Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *Theoretical Computer Science*, 265(1–2):265–283, 2001.
43. Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81(1-2):155–181, 1996.
44. A. Srinivasan. A learning engine for proposing hypotheses (Aleph). http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph, 1999.
45. Jonathan S.Turner. Almost all k-colorable graphs are easy to color. *J. Algorithms*, 9(1):63–82, 1988.
46. L. G. Valiant. A theory of the learnable. In *ACM Symposium on Theory of Computing (STOC '84)*, pages 436–445, Baltimore, USA, April 1984. ACM Press.
47. Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artif. Intell.*, 171(8-9):514–534, 2007.
48. P.W. Anderson Y. Fu. Application of statistical mechanics to np-complete problems in combinatorial optimization. *Journal of Physics*, 19:1605–1620, 1986.