Journal of Machine Learning Research 4 (2003) 431-463 Submitted 5/01; Published 8/03

Relational Learning as Search in a Critical Region

Marco Botta BOTTA@DI.UNITO.IT
Attilio Giordana ATTILIO@MFN.UNIPMN.IT
Dipartimento di Informatica
Università di Torino
10149, Torino, Italy

Lorenza Saitta SAITTA@MFN.UNIPMN.IT
Dipartimento di Informatica, Università del Piemonte Orientale
15100 Alessandria, Italy

Michèle Sebag MICHELE.SEBAG@LRI.FR
Equipe Inference et Apprentissage, Lab. de Recherche en Informatique
Université Paris-Sud Orsay
91405 Orsay Cedex, France

Editors: James Cussens and Alan M. Frisch

Abstract

Machine learning strongly relies on the covering test to assess whether a candidate hypothesis covers training examples. The present paper investigates learning relational concepts from examples, termed *relational learning* or *inductive logic programming*. In particular, it investigates the chances of success and the computational cost of relational learning, which appears to be severely affected by the presence of a phase transition in the covering test. To this aim, three up-to-date relational learners have been applied to a wide range of artificial, fully relational learning problems. A first experimental observation is that the phase transition behaves as an attractor for relational learning; no matter which region the learning problem belongs to, all three learners produce hypotheses lying within or close to the phase transition region. Second, a *failure region* appears. All three learners fail to learn any accurate hypothesis in this region. Quite surprisingly, the probability of failure does not systematically increase with the size of the underlying target concept: under some circumstances, longer concepts may be easier to accurately approximate than shorter ones. Some interpretations for these findings are proposed and discussed.

1. Introduction

In a seminal paper, Mitchell (1982) characterized machine learning (ML) as a search problem. Indeed, every learner basically explores some solution space until it finds a (nearly) optimal or sufficiently good solution. Since then much attention has been devoted to every component of a search problem: the search space, the search goal, and the search engine.

The search space reflects the language chosen to express the target knowledge, called the *hypothesis language*. Although most works in ML consider attribute-value languages (Mitchell, 1982; Quinlan, 1986), first-order logic languages, or a subset thereof (Michalski, 1983), are required to deal with structured application domains such as chemistry or natural language processing. This paper focuses on learning in first-order logic languages, referred to as *relational learning* (Quinlan, 1990) or *inductive logic programming* (Muggleton and De Raedt, 1994). This task has been tackled by many authors since the early 1990s, following Bergadano et al. (1991), Pazzani and Kibler (1992) and Muggleton and Feng (1992).

1.1 Relational Learning Complexity

In machine learning, assessing the quality of any hypothesis relies heavily on the *covering test*, which verifies whether a given hypothesis h is satisfied by an example E. In the logic-based approach to learning taken by inductive logic programming (ILP), the covering test most commonly used is θ-subsumption (Plotkin, 1970; Muggleton, 1992; Nienhuys-Cheng and de Wolf, 1997), which has been proved to be an NP-hard problem (Nienhuys-Cheng and de Wolf, 1997). Then, a major concern facing relational learning is computational complexity. Most studies in computational learning are rooted in the PAC-learning framework proposed by Valiant (1984)[1], and the literature offers many results regarding the worst-case and average-case complexity of learning various classes of relational concepts (Cohen, 1993; Džeroski et al., 1992; Kietz and Morik, 1994; Khardon, 1998).

However, another emerging framework offers a more detailed perspective on complexity than just worst- and average-case analysis. This framework, initially developed in relation to search problems, is referred to as the *phase transition (PT) framework* (Hogg et al., 1996a), and it considers computational complexity as a random variable that depends on some *order parameters* of the problem class at hand. Computational complexity is thus modeled as a distribution over problem instances (statistical complexity).

The advantage of the statistical complexity paradigm is twofold. First, it accounts for the fact that, despite the exponential worst-case complexity of an NP-hard class of problems, most problem instances are actually easy to solve; second, it shows that many very hard problem instances concentrate in a narrow region, termed the *mushy* region (Hogg et al., 1996a). Most problem instances, in fact, are either under-constrained (they admit many solutions, and finding one of them is easy), or over-constrained (to such an extent that it is easy to show that no solution exists). Thus, the computational complexity landscape of a problem class appears to be divided into three regions: the "YES" region, including the under-constrained problem instances, where the probability of any of them being solvable is close to one; the "NO" region, including the over-constrained problem instances, where the probability of any of them being solvable is close to zero; and the narrow mushy region in between, across which the probability of a random problem instance being solvable abruptly drops from almost one to almost zero. The mushy region includes the phase transition, defined as the locus, in the parameter space, where the probability for a random problem instance being solvable is 0.5. Interestingly enough, it has been observed that a large peak in computational complexity usually appears in correspondence to the phase transition (Cheeseman et al., 1991; Williams and Hogg, 1994; Hogg et al., 1996b; Walsh, 1998).[2]

To sum up, the phase transition region is, on average, the hardest part of the parameter space. This fact has many theoretical and practical implications. One such implication is that any complexity analysis should try to locate the PT region; to this aim, one must identify the order parameters for the problem class at hand, and the critical values for these parameters corresponding to the location of the phase transition. Second, any new algorithm must be evaluated on problem instances lying in the PT region. Complexity results obtained on problem instances belonging to the YES or NO

1. For a detailed presentation see Kearns and Vazirani (1994).
2. In this paper we will use the terms "mushy region" and "PT region" as synonyms.

regions are likely to be irrelevant for characterizing the class complexity, as pointed out by Hogg et al. (1996a).

1.2 Phase Transition and Relational Learning

Previous experiments (Botta, Giordana, and Saitta, 1999; Botta, Giordana, Saitta, and Sebag, 2000; Giordana and Saitta, 2000) have shown the presence of a phase transition in the covering test, both in artificial and real-world learning problems. Starting from the obtained results, this paper offers a deeper insight into the observed effects of the phase transition on the quality and complexity of learning. The initial focus in the work by Botta, Giordana, and Saitta (1999); Botta, Giordana, Saitta, and Sebag (2000) and by Giordana and Saitta (2000) was on complexity issues, especially on finding ways around the dramatic complexity increase. However, this paper shows that the existence of a phase transition in the covering test has much deeper and far reaching effects on the feasibility of relational learning than just this increase in computational cost. Preliminary results in this new direction have been presented by Giordana, Saitta, Sebag, and Botta (2000). The emergence of a phase transition impacts machine learning in at least three respects:

- A region of significant size around the phase transition appears to contain extremely difficult learning problems.
- Popular search heuristics in machine learning, such as *information gain* (Quinlan, 1990) or *minimum description length* (Rissanen, 1978; Muggleton, 1995) are not reliable in the early stages of top-down hypothesis generation. These heuristics, in fact, provide reliable information only when the search enters the phase transition region.
- A low generalization error of a learned hypothesis does not imply that the "true" target concept has been captured. This is particularly important for automated knowledge discovery, where a major issue is to provide experts with new relevant insights into the domain under analysis. As a matter of fact, good approximations of a complex concept can only be found near the phase transition, irrespectively of the location of the concept. Then, discovering the "true" concept is extremely unlikely if it lies outside the phase transition region.

The above mentioned experiments have been performed using a problem generator originally designed to check the existence of a phase transition (Botta et al., 2000; Giordana and Saitta, 2000), and later extended to generate artificial, fully relational learning problems *with known target concept*. A suite of some five hundred problems has been constructed, sampling the YES, NO and PT regions. A popular relational learner, FOIL 6.4 (Quinlan, 1990), has been systematically applied to these problems. Moreover, two other learners, SMART+ (Botta and Giordana, 1993) and G-Net (Anglano et al., 1998), have been independently applied on a subset of the problems, selected in order provide a comparison with different search strategies.

These systematic experiments shed some light on the behavior, capabilities and limitations of existing relational learners. First of all, for all three relational learners considered, the PT region is actually an attractor in the sense that whatever the location of the learning problem, with respect to the phase transition, in most cases the learners conclude their search in the PT region. Second, a *failure region* appears, where all three learners fail to discover either the target concept, or any acceptable approximation thereof. The failure region largely overlaps the PT region; learning problems close to the phase transition appear to be much harder than others. Unexpectedly, very long concepts happen to be easier to learn than shorter ones under some circumstances. Interpretations for these findings are proposed, and their implications regarding relational learning are discussed.

The rest of the paper is organized as follows. To be self-contained, Section 2 summarizes previous results regarding the existence and location of the phase transition for the covering test (the reader is referred to Giordana and Saitta, 2000, for a detailed presentation). Section 3 describes the experimental setting used to estimate the impact of the phase transition on relational learning: learning problems sampling the three regions (under-constrained or YES-region, PT or mushy region, over-constrained or NO-region) are constructed. Section 4 describes the empirical results obtained on these problems by FOIL (Quinlan, 1990), SMART+ (Botta and Giordana, 1993) and G-Net (Anglano et al., 1998). Some interpretations for these results are proposed in Section 5. Section 6 discusses the extent to which these findings lead to the reconsideration of the current biases and search strategies for relational learning, and the paper ends with some suggestions for further research.

2. Phase Transition in Matching Problems

Machine learning proceeds by repeatedly generating and testing hypotheses; these hypotheses are rated according to (among other criteria) their coverage of training examples (Mitchell, 1997). In this paper, we will consider the simplest setting used in relational learning, where concepts are described by sets of clauses in a restricted Datalog language (Date, 1995) having the form:

$$c :- h(X_1, X_2, \dots, X_n). \tag{1}$$

In (1) c is a *class* name[3] and h a conjunction of literals that may contain variables or constants, but no function symbols; negation is allowed, but only on single literals on the right hand side. The variables occurring in h, if any, are $X_1, X_2, \dots, X_n$ As a matter of fact, Datalog is a hypothesis language largely used in relational learning (Muggleton and De Raedt, 1994; Nienhuys-Cheng and de Wolf, 1997), and concept descriptions in the form (1) have been used in many applications, since the early approach by Michalski (1983).

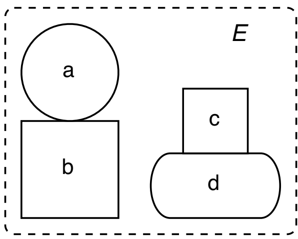
In several learning approaches, notably in data mining, an example E is described as a set of tables; each table corresponds to a basic predicate $r(X_1, X_2, \dots, X_s)$ of the language, and each row in the table is associated to a tuple of objects $\langle a_1, a_2, \dots, a_s \rangle$ in E such that $r(a_1, a_2, ..., a_s)$ is true. Figure 1 shows an example for the sake of illustration.

Alternatively, in the logical approach taken by ILP (Muggleton, 1992), examples are represented as conjunctions of a possibly large number of ground facts (ground literals). The transformation between tabular representation and ground literal representation is immediate: every row $\langle a_1, a_2, ..., a_s \rangle$ in the table associated to a predicate r is transformed into an equivalent ground literal $r(a_1, a_2, ..., a_s)$. For instance, the example E in Figure 1 can be represented as follows:

$$E: \ on(a,b),\ on(c,d),\ left(a,c),\ left(a,d),\ left(b,c),\ left(b,d). \tag{2}$$

All three learning algorithms used in this paper, namely FOIL (Quinlan, 1990), SMART+ (Botta and Giordana, 1993) and G-Net (Anglano et al., 1998), use the tabular representation of the examples, at least as the internal representation, and have been applied to the task of learning concepts in form (1).

3. We consider the simple case in which the head of the clause does not have variables.



(a)

on(X,Y)

X	Y
a	b
c	d

left(X,Y)

X	Y
a	c
a	d
b	c
b	d

(b)

left(X,Y),on(Y,Z)

X	Y	Z
a	c	d
b	c	d

(c)

Figure 1: Tabular representation of structured examples of the block world. (a) Block world instance E composed of four objects, a, b, c, and d. (b) Tables describing E, assuming that the description language contains only two predicates, namely $on(X,Y)$ and $left(X,Y)$. (c) Two substitutions for X,Y,Z that satisfy the hypothesis $h = left(X,Y), on(Y,Z)$. In particular, $h(a,c,d) = left(a,c), on(c,d)$ is true in E.

More precisely, given a description (an inductive hypothesis) $h(X_1, X_2, \ldots, X_n)$ and an example E described by the finite set $\mathbf{R} = \{\mathrm{r}_1, \mathrm{r}_2, ..., \mathrm{r}_\mathrm{m}\}$ of tables, let $\mathbf{A}$ be the set of all constants occurring in the tables of $\mathbf{R}$. Let us consider a substitution $\theta = (X_1/a_1, X_2/a_2, ...X_n/a_n)$. Let $r_i(X_{i_1}, .., X_{i_k})$ be a literal occurring in h and built on predicate symbol r_i. Let, moreover, $(X_{i_1}/a_{i_1}, ..., X_{i_k}/a_{i_k})$ be the substitution defined by θ for the variables $\langle X_{i_1}, .., X_{i_k}\rangle$. If $h\theta$ is true in E (i.e., h covers E) the tuple $\langle a_{i_1}, ..., a_{i_k}\rangle$ must occur in the table $\mathrm{r_i}$ associated to predicate r_i. An analogous condition must hold for each literal in h. Then, when concept descriptions have form (1), the *covering test* consists in checking if the formula $\exists_{X_1, X_2, \ldots, X_n}.h(X_1, X_2, \ldots, X_n)$ is true in a given example E. This means to find a substitution $\theta = (X_1/a_1, X_2/a_2, ...X_n/a_n)$ such that $h(a_1, a_2, ..., a_n)$ is true.

It is easy to observe (see Giordana and Saitta, 2000, for details) that this covering test (or matching problem) is equivalent to a constraint satisfaction problem (CSP) (Prosser, 1996): the set $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ of variables occurring in h corresponds to the CSP's variable set, the set $\mathbf{A}$ of constants in E corresponds to the domain(s) over which the variables may range[4], and the set $\mathbf{R}$ of tables in E corresponds to CSP's set of constraints. Clearly, the hypothesis h is the formula that must be satisfied in the corresponding CSP.

Let us notice that a learning algorithm shall solve a number of matching problems (h, E) equal to the number of training examples multiplied by the number of generated hypotheses. This product can easily grow to the hundreds of thousands.

2.1 Order Parameters in the Constraint Satisfaction Problem

As the covering test is equivalent to a CSP, the theory developed for CSPs can be applied to the matching problem. In this paper, we are interested only in predicates (constraints) with arity not greater than two; thus, binary CSPs are the only ones of interest. In this paper, as well as in previous experiments (Giordana and Saitta, 2000; Botta et al., 1999), we have adopted the standard treatment

4. Each variable X_i may take values in a specific set $\mathbf{A}_i$. Then, $\mathbf{A}$ is the union of $\mathbf{A}_i$'s.

of binary CSP provided by Smith and Dyer (1996) and Prosser (1996). Hence, we will just recall here a few basic notions in order to make this paper as self-contained as possible. A binary CSP can be represented as a graph, whose vertices correspond to the variables in **X**; an edge between two vertices denotes the presence of a constraint on the corresponding variable pair.

Two parameters have been defined to characterize a CSP instance: *constraint density* p_1, and *constraint tightness* p_2 (Smith and Dyer, 1996; Prosser, 1996). Let γ denote the number of edges in the constraint graph; the constraint density p_1 can be defined as:

$$p_1 = \frac{\gamma}{\frac{n(n-1)}{2}} = \frac{2\gamma}{n(n-1)} \tag{3}$$

(We remind the reader that n is the number of variables in formula h.) The tightness, p_2, of a constraint is defined as the fraction of value pairs ruled out by the constraint. If N is the size of table $r(X_i, X_j)$ (i.e., the number of literals built on the predicate symbol r, in ILP terminology), and L is the size of the variable domain (assuming that all variables range over the same domain), constraint tightness p_2 is defined as:

$$p_2 = 1 - \frac{N}{L^2} \tag{4}$$

Studies on CSPs are based on stochastic models. For instance, Smith and Dyer (1996) propose Model B, where the number n of variables, the table size N, and the constraint density p_1 are kept constant, and the constraint tightness p_2 varies from 0 to 1. A constraint $r(X_i, X_j)$ is constructed by choosing the predicate symbol r and by uniformly selecting without replacement pairs of variables $\langle X_i, X_j \rangle$. Tables are then constructed by uniformly extracting, without replacement, N pairs $\langle a_k, a_l \rangle$ of constants from $\mathbf{A} \times \mathbf{A}$.

In the standard CSP model, p_1 is kept constant and p_2 varies in $[0,1]$ (Williams and Hogg, 1994). Accordingly, the probability P_{sol} that the current CSP instance to be satisfiable abruptly drops from 0.99 to 0.01 in the narrow "mushy" region. It has been observed that the complexity of either finding a solution or proving that none exists shows a marked peak for $P_{sol} = 0.5$, which is also called the *crossover point* (Crawford and Auton, 1996; Smith and Dyer, 1996). The p_2 value corresponding to the crossover point, $p_{2,cr}$, is called the *critical* value. It is conjectured that the critical value corresponds to an expected number of solutions close to 1 (Williams and Hogg, 1994; Smith and Dyer, 1996; Prosser, 1996; Gent and Walsh, 1996); experimentally, unsatisfiable instances (admitting no solution) are computationally much more expensive, on average, than satisfiable ones.

Some limitations of Model B (Smith and Dyer, 1996), regarding the asymptotic complexity, have been pointed out by Mitchell et al. (1992) and Achlioptas et al. (1997). However, this model can be considered adequate for the limited problem ranges considered in this paper and in relational learning in general.

2.2 The Phase Transition in Matching Problems

Let us briefly describe the framework used by Botta et al. (1999) and Giordana and Saitta (2000) to investigate the presence of a phase transition in matching.

An instance of the covering test is a pair (h, E) of (*hypothesis* - *example*). As previously defined, **X** is the set of variables in h, **A** is the set of constants in E, and **R** is the set of tables in E. The problem is to check whether h can be verified in E, i.e., h covers E. A number of simplifying assumptions have been considered:

- all predicates in the language are binary,
- the hypothesis h is conjunctive and contains only one occurrence of each of m predicate symbols,
- the constraint graph is connected, so that the corresponding CSP cannot be decomposed into independent smaller ones,
- every table r in $\mathbf{R}$ contains the same number N of tuples, and
- all variables range over the same domain $\mathbf{A} = \{a_1, \ldots, a_L\}$.

The emergence of a phase transition in matching has been experimentally studied by constructing a large number of matching problem instances. Each instance (h, E) can be characterized by a 4-tuple (n, N, m, L), where:

- n is the number of distinct variables in h,
- m is the number of distinct predicate symbols occurring in h,
- L is the total number of constants occurring in E, and
- N is the number of rows in each table in E.

The standard binary CSP order parameters p_1 and p_2 can be expressed in terms of n, m, N and L (Botta et al., 1999). However, we prefer to directly use n, m, N and L, because, unlike p_1 and p_2, they have a natural interpretation in relational learning.

A systematic analysis of the form and location of the phase transition in the 4-dimensional space (n, N, m, L) would have been practically impossible. Thes we have selected m and L as principal order parameters for running systematic experiments, whereas n and N have been considered as secondary parameters. The reason for this choice is that m and L are directly linked to hypothesis and example complexity, respectively.

Nevertheless, it is worth noticing that most works on phase transitions consider just one order parameter, a choice that makes both the analysis and the visualization of the results much easier. In our case, however, the elimination of one of the two chosen parameters would lose fundamental information; in fact, both the hypothesis and the example contribute in an essential way to the phenomenon. Even a combined parameter, such as Walsh's κ parameter (Walsh, 1998), could not retain all the relevant information.

The artificial problem generator, used for the experiments, is inspired by Model B proposed by Smith and Dyer (1996). For any 4-tuple (n, N, m, L) with $m \geq n-1$, a thousand matching problems $(h,\ E)$ have been stochastically constructed as follows:

- We first construct h such that it is connected[5] (first part in the right-hand side), and then the remaining $(m-n+1)$ literals are added:

$$h(X_1 \ldots X_n) = \bigwedge_{i=1}^{n-1} r_i(X_i, X_{i+1}) \wedge \bigwedge_{j=n}^{m} r_j(X_{i_j}, X_{k_j}) \tag{5}$$

5. The goal is to prevent the equivalent CSP from being decomposable into independent, smaller subproblems involving disjoint sets of variables. Such decomposability, referred to as k-locality, greatly reduces the complexity of matching (Kietz and Morik, 1994).

h contains exactly n variables and m literals, all built on distinct predicate symbols. The same pair of variables may appear in more than one literal.

- For each predicate symbol r, a corresponding table is built up, by uniformly selecting N elements without replacement from the set of all value pairs $\langle a_i, a_j \rangle$ from the set $\mathbf{A} \times \mathbf{A}$. This ensures that the table will not contain duplicate tuples.

Figures 2, 3 and 4 summarize the results obtained by Botta et al. (1999) and Giordana and Saitta (2000). The experiments have been done using a simple variant of a classical back-tracking algorithm (see Giordana and Saitta, 2000, for details). For each tuple (n, N, m, L) one thousand problems have been generated and the percentage of success P_{sol} has been recorded. When plotted in the plan (m, L) (Figure 2), it shows the existence of a very steep phase transition even for a small number n of variables.

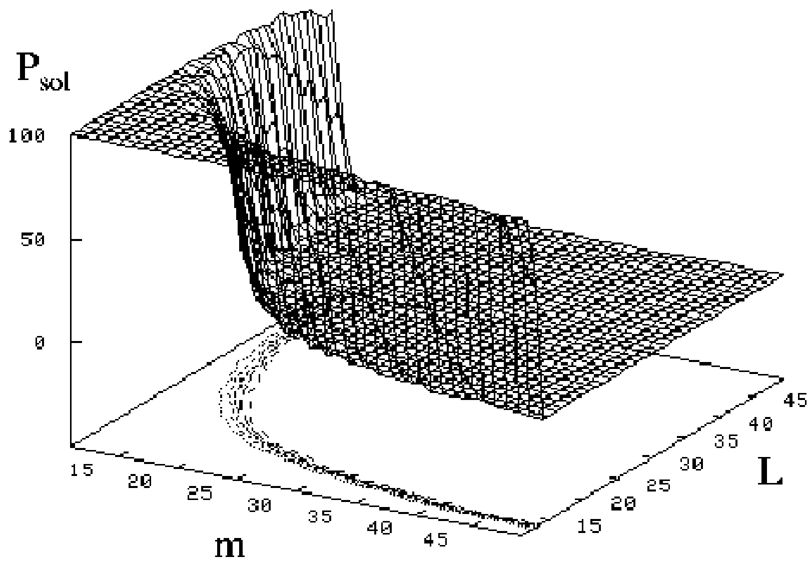
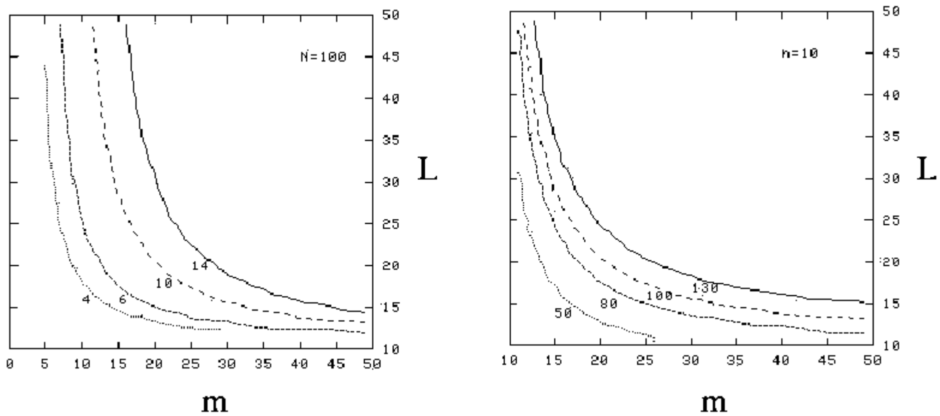


Figure 2: Probability P_{sol} that a random hypothesis covers a random example, averaged over one thousand pairs (h,E) for each (m, L) point. A hypothesis h contains $n = 10$ variables and is associated to m tables, each one containing $N = 100$ rows involving L constants. On the horizontal plane, the contour plots corresponding to P_{sol} values in the interval $[0.1, 0.9]$ have been projected.

As the number n of variables increases, the phase transition moves away from the axes and the average search complexity correspondingly increases; a similar effect occurs when the table size N increases (see Figure 3).

Figure 4 shows the average computational cost of the covering test, measured in seconds, on a Sparc Enterprise 450. The average complexity grows, and the phase transition becomes narrower as the number n of variables increases. As mentioned above, the matching algorithm was a basic one. Using more sophisticated CSP algorithms, the height of the peaks could be somewhat reduced (Maloberti and Sebag, 2001), but the peaks are still clearly noticeable (Hogg et al., 1996b).



(a) $n = 4, 6, 10, 14$, with $N = 100$

(b) $N = 50, 80, 100, 130$, with $n = 10$

Figure 3: Location of the phase transition (contour plot corresponding to $P_{sol} = 0.5$) in the (m, L) plane for various values of n (left) and N (right).

3. Goals and Experimental Setting

This section describes the problems used to investigate the impact of the presence of a phase transition on relational learning. Surprisingly, this impact is not limited to the obvious increase in computational complexity when the search for hypotheses enters the mushy region; it also deeply affects both the quality and the meaningfulness of the learned knowledge.

3.1 Generating Artificial Learning Problems

A relational learning problem Π is a triple $(h, \mathcal{E}_L, \mathcal{E}_T)$, where h is a target concept description that has to be discovered by a learning algorithm, and $\mathcal{E}_L$ and $\mathcal{E}_T$ are the learning and test sets, respectively. Every example E belonging to $\mathcal{E}_L$ or $\mathcal{E}_T$ is represented as a set of m tables, each one identified by a different predicate symbol r from a set $\mathbf{R}$.

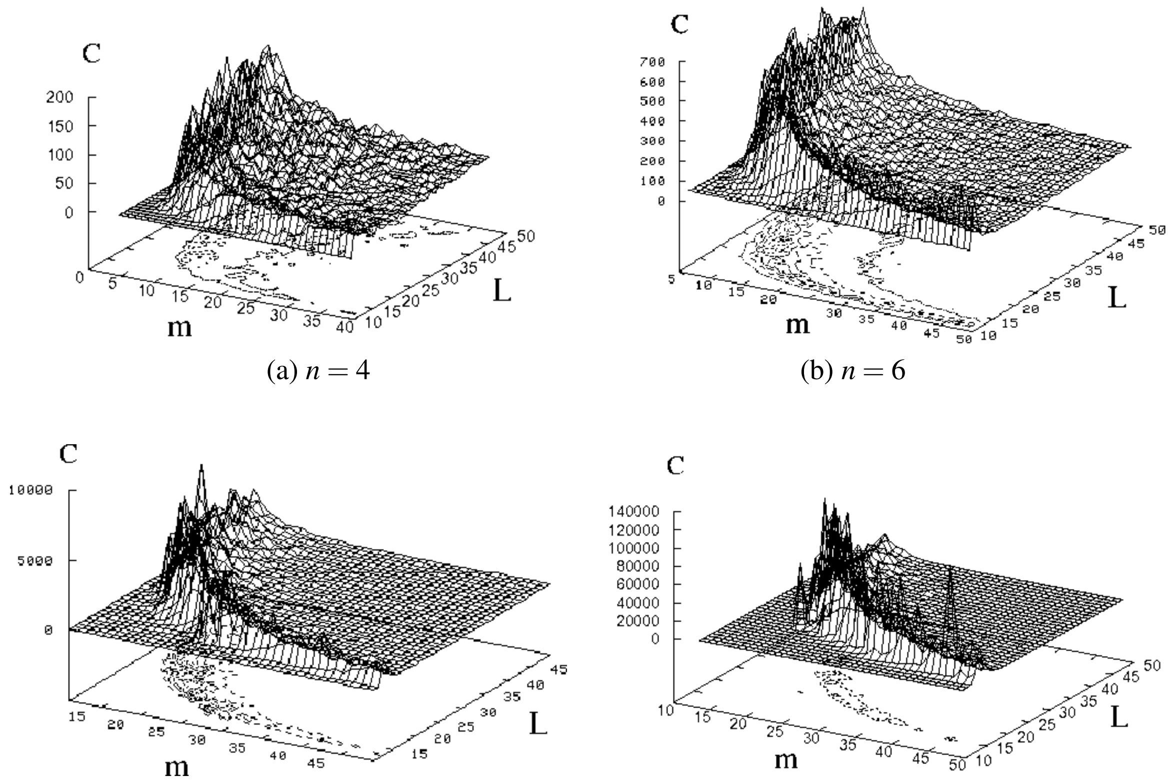
For the sake of simplicity, we consider binary predicates (tables) only, as previously done by Giordana and Saitta (2000) (see Section 2.2). Moreover, we assume that all positive examples in $\mathcal{E}_L$ and in $\mathcal{E}_T$ are perfectly discriminated by a single clause of type (1), where the right hand side

$$h(X_1, X_2, ..., X_n) = r_1(X_{1_1}, X_{1_2}), \ldots, r_m(X_{m_1}, X_{m_2}) \tag{6}$$

is a connected conjunction of literals, constructed as explained in Section 2.2. Finally, as we require that every predicate defined in the examples occurs exactly once in description (6), all the predicates in the description language are equally relevant. A complementary set of experiments (not reported in this paper) shows that this restriction does not affect the results to a significant extent.

As said above, in order to keep the computational cost within reasonable limits, the number n of variables is fixed at four ($n = 4$) in all target concepts.[6] Each example E has been generated

6. Note that up-to-date relational learners deal with similar restrictions. For instance, in the Mutagenesis domain (King et al., 1995), the maximum number of chemical atoms considered in a hypothesis, corresponding here to the number of distinct variables, varies from 3 to 5.



(a) $n = 4$ (b) $n = 6$

(c) $n = 10$ (d) $n = 14$

Figure 4: Computational cost for a single covering test (measured in seconds on a Sparc Enterprise 450) for various values of the number n of variables. Each point is the average on 1000 matching problems, as in Figure 2. Figure (a) corresponds to the case considered in the paper ($n = 4$).

following the procedure described in Section 2.2. Again, for computational reasons, the number N of literals is fixed at 100 ($N = 100$).[7]

In order to visit as uniformly as possible the YES, NO and mushy regions, while avoiding an exhaustive exploration, 451 pairs (m,L) have been uniformly selected without replacement, where m ranges in $[5,30]$ and L ranges in $[12,40]$.

For each selected pair (m,L) an artificial learning problem $\Pi_{m,L}$ is constructed; the target concept description is constructed as described in Section 2.2. The training and test examples are constructed as in Section 2.2, as well, and then stochastically modified to address the following problem. If (m,L) lies in the YES region (on the left of the phase transition), by construction the concept description will almost surely cover any stochastically constructed example. In other words, the training and test sets would contain a large majority of positive examples (the ratio of negative

7. Given the large number of possible arguments ($2 \times m \times N = 200m$), it seldom happens that an example involves less than L constants. The only effect of small variations of L from example to example is a (very) slight increase on the apparent width of the phase transition region. A similar argument can be put forward for the uniformity of the N value in all tables.

versus positive examples is 1 to 10^6 or higher). Symmetrically, if (m, L) lies in the NO-region (on the right of the phase transition), the training and test sets would contain a large majority of negative examples.

However, it is widely acknowledged that ill-balanced datasets make learning considerably more difficult. As this additional difficulty might blur the results and their analysis, we construct balanced training and test sets, each consisting of 100 positive and 100 negative examples. The example generator is then modified; a repair mechanism is added to ensure the fair distribution of the training and test sets for learning problems lying outside the phase transition region. As a consequence, the generation of the examples proceeds as follows.

Function Problem_Generation(m, L)

Construct a description h with m literals of concept c.
$\mathcal{E}_L$ = Data_Generation(m, L, h).
$\mathcal{E}_T$ = Data_Generation(m, L, h).
Return $\Pi = (h, \mathcal{E}_L, \mathcal{E}_T)$.

Function Data_Generation(m, L, h)

```
nb_positive = 0, nb_negative = 0
Let ℰ = ∅
while nb_positive < 100 or nb_negative < 100 do
    Generate a random example E

    if E is covered by h then
        if nb_positive = 100 then
            E = ChangeToNegative(h, E)
            Set label = NEG
        else Set label = POS
    else
        if negative = 100 then
            E = ChangeToPositive(h, E)
            Set label = POS
        else Set label = NEG
    ℰ = ℰ ∪ { (E,label) }
   if label = POS then nb_positive = nb_positive + 1
       else nb_negative = nb_negative + 1
end
Return ℰ.
```

Function Problem_Generation first constructs a description h of the target concept c; then, the training and test sets are built by the function Data_Generation. The latter accumulates examples constructed with the stochastic procedure described in Section 2. When the maximum number of positive (respectively, negative) examples is reached, further examples are repaired using the ChangeToNegative (respectively, ChangeToPositive) function, to ensure that the training and test sets are well balanced. Function ChangeToPositive turns a negative example E into a positive one, by inserting into its tables a proper set of tuples, satisfying the conditions set in h, randomly selected in the space $\mathbf{A}^4$.

Function ChangeToPositive(h, E)

Uniformly select four constants, a_1, a_2, a_3, a_4, from $\mathbf{A}$.
Let θ denote the substitution $\theta = \{X_1/a_1,\ X_2/a_2,\ X_3/a_3,\ X_4/a_4\}$
for each literal $r_k(X_i, X_j)$ in h, **do**
  if tuple $\langle a_i, a_j \rangle$ does not already occur in table r_k of E **then**
    select randomly and uniformly a tuple occurring in table r_k
      and replace it by $\langle a_i, a_j \rangle$
end
Return E.

Conversely, function ChangeToNegative modifies E in order to prevent it from being covered by h. Let $\theta = \{X_1/a_1,\ X_2/a_2,\ X_3/a_3,\ X_4/a_4\}$ be a substitution verifying h in an example E. Let (X_j/a_j) $(2 \leq j \leq 3)$[8] be the j-th element of θ, and let $r_k(X_i, X_j)$ be one of the literals in h. In order to falsify $h(a_1, a_2, a_3, a_4)$ in E it is sufficient to delete tuple $\langle a_i, a_j \rangle$, from the table associated to r_k. Unfortunately, this would decrease the number N of tuples in a table. To avoid this problem it is sufficient to add a new tuple (in substitution of $\langle a_i, a_j \rangle$) that is guaranteed not to satisfy h. This is done in either one of two alternative ways. First, we search for a constant a'_j that does not occur in the left column of the tables associated to the literals where variable X_j occurs on the left hand side. If such construct exists, tuple $\langle a_i, a_j \rangle$ is replaced with $\langle a_i, a'_j \rangle$. Otherwise, $\langle a_i, a_j \rangle$, is replaced with a tuple already existing in the table of r_k but selected among the ones that do not contribute to satisfy h.[9]

Function ChangeToNegative(h, E)

Build the set Θ of all substitutions $\{X_1/a_1,\ X_2/a_2,\ X_3/a_3,\ X_4/a_4\}$ such that $\langle a_1, a_2, a_3, a_4 \rangle$ satisfies h
while Θ is not empty **do**
 Randomly select an atom $r_k(X_j, X_i)$ in h
 Replace $\langle a_j, a_i \rangle$ in table r_k with a new tuple $\langle a'_j, a'_i \rangle$
   such that the number of alternative ways in which h is satisfied decreases
 Recompute Θ
end
Return E.

3.2 The Learners

Three learning strategies have been considered: a top-down depth-first search, a top-down beam search, and a genetic algorithm based search.

Most learning experiments presented in this paper have been done using the top-down learner FOIL (Quinlan, 1990), which outputs a disjunction of conjunctive partial hypotheses. FOIL starts

8. As the constants are associated to "chained" variables, in order to falsify $h(a_1, a_2, a_3, a_4)$, it is sufficient to consider the variables internal to the chain.
9. Notice that, in this case, the tuple distribution in the tables may be modified. However, this happens very rarely and only when L is small, so that the influence on the results is irrelevant.

with the most general hypothesis, and iteratively specializes the current hypothesis h_t by adding to its body the "best" conjunct $r_k(X_i, X_j)$, according to some evaluation criterion, such as Information gain (Quinlan, 1990, 1986) or minimum description length (MDL) (Rissanen, 1978). When any specialization able to improve h_t can be found, the latter is retained, all positive examples covered by h_t are removed from the training set, and the search is restarted, unless the training set is empty. The final hypothesis $\widehat{h}$ returned by FOIL is the disjunction of all the partial hypotheses h_t.

Another top-down learner, SMART+ (Botta and Giordana, 1993), has also been used. The main difference between FOIL and SMART+ resides in their search strategies; FOIL basically uses a hill-climbing search strategy, whereas SMART+ uses a beam search strategy with a user-supplied beam width.

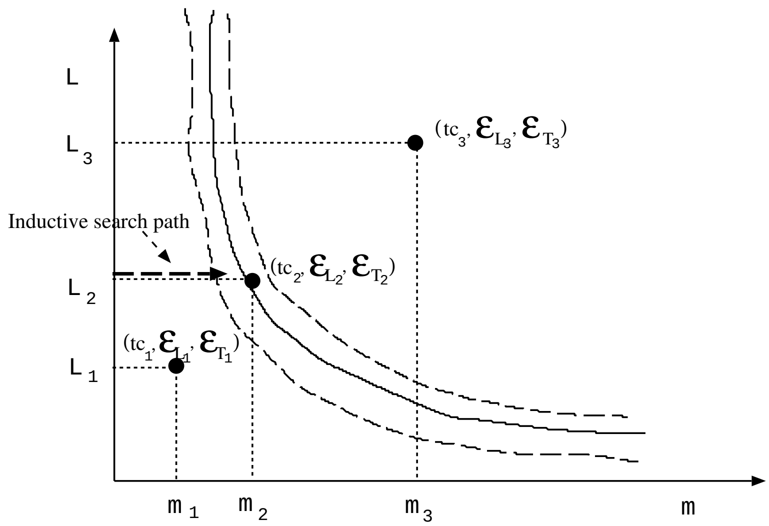


Figure 5: Location of the learning problems in the (m, L) plane. Top-down learners visit candidate hypotheses from left to right.

The search space visited by FOIL or SMART+ can be visualized as a path in the plane (m, L) (see Figure 5). Both learners navigate in the plane moving from left to right, as the number of literals in the current hypothesis is incremented at each step.

A third learner, named G-Net (Anglano et al., 1998) and based on genetic search, was also considered. G-Net starts with an initial population of candidate hypotheses; these correspond to problems randomly distributed on a segment of the horizontal line $L = |\mathbf{A}|$ in the (m, L) plane. The learner navigates on this straight line, moving to the right or to the left, since genetic operators allow candidate hypotheses to be either specialized or generalized. As usual with evolutionary computation-based search, the computational cost of G-Net is significantly higher than that of the other two learners. Only a reduced number of experiments have therefore been performed with G-Net, just to see whether a mixed top-down/bottom-up strategy would show any significant change in the results of learning.

Other experiments also considered the relational learners PROGOL (Muggleton, 1995) and STILL (Sebag and Rouveirol, 2000). In preliminary tests, PROGOL was never able to learn any hypothesis in acceptable time. In a similar way, STILL systematically failed. STILL uses a bottom-up approach, based on the stochastic (uniform or biased) sampling of the matchings between a hypothesis and the examples. Its failure is due to the uniform construction of the examples and the lack of any domain bias.

3.3 Discussion

Let us summarize here all simplifications and assumptions. Some of them should facilitate the search (legend $+$), while others rather hinder relational learning (legend $-$):

$+$ There are no constants in the target concept and no variables in the examples.

$+$ The training and test sets are equally distributed (100 positive and 100 negative examples), without any noise.

$+$ All target concepts are conjunctive: a single hypothesis h covers all positive examples and rejects all negative ones.

$+$ Both target concept and examples are single definite clauses.

$+$ All predicates in the examples are relevant: they all appear in the target concept. No other background knowledge is given to the learner.

$+$ variables in the description of the target concept are chained.

$-$ All examples have the same size (N times the number of predicate symbols in the target concept).

$-$ All tables (predicate definitions) have the same number of rows in every example.

$-$ All predicates are binary.

$-$ All predicate arguments have the same domain of values.

Let us notice that, even if the structure of the target concept (m literals involving $n = 4$ variables) were known by the learners (which is obviously not the case), the size of the search space (4^{2m}) prevents the solution being discovered by chance. Moreover, given the description h of c, the learning task amounts to finding the right bindings among the $2 \times m$ arguments involved in the m binary predicates, i.e., partitioning the $2m$ arguments into four subsets.

3.4 Goal of the Experiments

In order to investigate the impact of the phase transition on relational learning, our goal is to assess the behavior of all considered learners depending on the position of learning problems with respect to the phase transition. The behavior of each learner is examined with regards to three specific criteria:

- **Predictive accuracy**. The accuracy is commonly measured by the percentage of test examples correctly classified by the hypothesis $\widehat{h}$ produced by the learner.[10] The accuracy is considered satisfactory if and only if it is greater than 80% (the issue of choosing this threshold value will be discussed later).

10. The predictive accuracy was not evaluated according to the usual cross-validation procedure for two reasons. First of all, the training and test sets are drawn from the same distribution; it is thus equivalent to doubling the experiments

- **Concept identification**. It must be emphasized that a high predictive accuracy does *not* imply that the learner has discovered the true target concept. The two issues must therefore be distinguished. The identification is considered satisfactory if and only if the structure of $\widehat{h}$ is close to that of the true target concept, i.e., if $\widehat{h}$ is conjunctive with the same size as h.

- **Computational cost**. The computational cost reflects both the total number of candidate hypotheses considered by the learner, and the cost of assessing each of them on the training set. Typically, the more candidate hypotheses in the phase transition region, the higher the computational cost.

4. Results

This section reports the results obtained by FOIL, SMART+ and G-Net on the artificial relational learning problems constructed as previously described.

4.1 Predictive Accuracy

Figure 6 summarizes the results obtained by FOIL with respect to predictive accuracy. As mentioned earlier, 451 pairs (m, L) have been chosen in order to explore significant parts of the YES, mushy, and NO regions. For each selected pair (m, L) a learning problem $\Pi_{m,L} = (h,\ \mathcal{E}_L,\ \mathcal{E}_T)$ has been constructed, where m is the number of literals in h and L the number of constants in the training/test examples.

On each problem, FOIL either succeeds (legend "+", indicating that the predictive accuracy on the test set is greater than 80%), or fails (legend "·").

Let us first comment on the significance of these results, with respect to the success threshold and the learning strategy. First, the shape of the failure region (the "blind spot") is almost independent of the threshold used to define a failure case (predictive accuracy on the test set lower than 80%). In a vast majority of cases, the hypotheses $\widehat{h}$ learned by FOIL are either very accurate (predictive accuracy close to 100%), or comparable to a random guess (predictive accuracy close to 50%). The threshold could thus be any value between 95% and 60%, without making any significant difference in the shape of the blind spot.

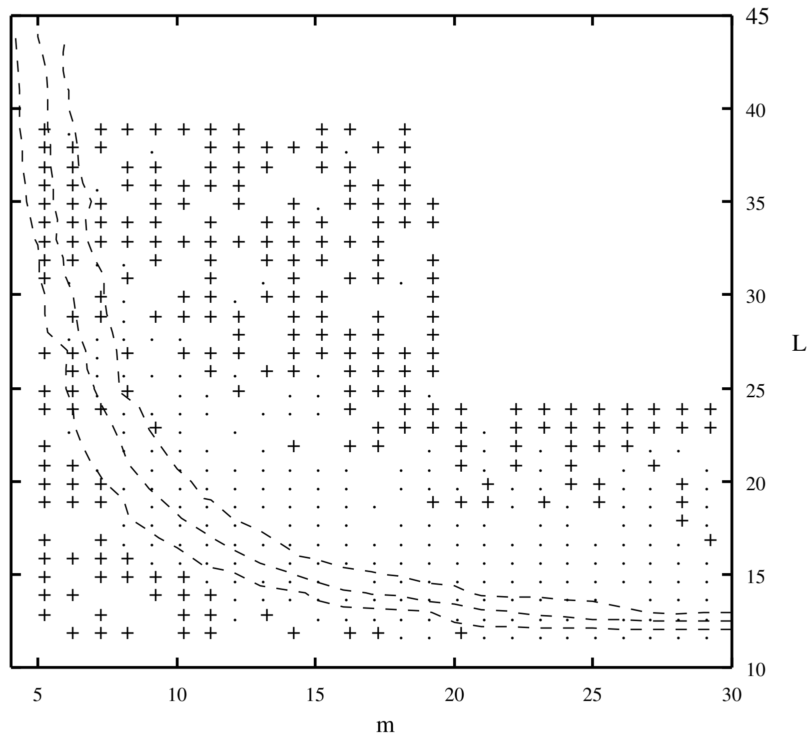
Regarding the mere learning performances, it appears that FOIL succeeds mainly in two cases: either when the target concept is simple (for low values of m), or when the learning problem is far (to the right) from the phase transition region.

The first case is hardly surprising; the simpler the target concept, the easier learning should be. Much more unexpected is the fact that learning problems far to the right of the phase transition appear to be easier to solve. In particular, the fact that increasing the number of constants[11] in the application domain *facilitates* relational learning (everything else being equal, i.e., for the same target concept size), is counter-intuitive. Along the same lines, it is counter-intuitive that increasing the size m of the target concept might facilitate relational learning (everything else being equal

---

and taking the average result, or performing a twofold cross-validation (Dietterich, 1998). We did not double the experiments because of the huge total computational cost. Moreover, though the learning result obtained for (m, L) is based on a single trial, it might be considered significant to the extent that other trials done in the same area give the same result.

11. Note that for $m > 6$ the learning problem moves away from the phase transition as L increases.



"+" : success (accuracy on $\mathcal{E}_T$ more than 80%) "·" : failure (accuracy on $\mathcal{E}_T$ less than 80%).

Figure 6: FOIL's competence map: *success* and *failure* regions, for $n = 4$ and $N = 100$. The phase transition region is indicated by the dashed curves, corresponding to the contour plots for $P_{sol} = 0.9$, $P_{sol} = 0.5$, and $P_{sol} = 0.1$, respectively, as determined by Giordana and Saitta (2000)).

again, i.e., for the same number of constants). These remarks will be commented upon further in Section 5.

4.2 Concept Identification

But what is it that really happens when FOIL succeeds or fails? Table 1 reports the characteristics of the final hypothesis $\widehat{h}$ produced by FOIL for a few representative learning problems.

The first column indicates the region the learning problem belongs to. The second one reports the identifier of the problem, which will be referred to in the discussion. Columns 3 and 4 show the parameters of the learning problem, i.e., the size m of the target concept description h, and the number L of constants in the examples. Columns 5 and 6 describe the hypothesis $\widehat{h}$ learned by FOIL; $\widehat{h}$ involves one or several conjunctive hypotheses h_t, iteratively produced by FOIL. The number of such h_t, noted $|\widehat{h}|$, is given in Column 5. It should be remembered that the true target concept h is conjunctive, i.e., a correct identification of h implies $|\widehat{h}| = 1$. The maximum, minimum and average size of the conjunctive hypotheses h_t learned by FOIL are displayed in column 6 (legend $m(h_t)$). These are to be compared to the true size m of the target concept description (column 3).

The last three columns report the predictive accuracy of $\widehat{h}$ on the training and test set, and the total CPU time required by FOIL to complete the learning task, measured in seconds on a Sparc Enterprise 450. The learning problems in Table 1 can be grouped into three categories:

- *Easy* problems, which are correctly solved; FOIL finds a conjunctive hypothesis $\widehat{h}$ that accurately classifies (almost) all training and test examples. Furthermore, $\widehat{h}$ is identical to the true concept description h, or differs by at most one literal. Problems of this type are Π_0 to Π_5, Π_7, Π_{10}, Π_{11}, Π_{27} and Π_{31}. Most easy problems lie in the YES region; some others lie in the mushy region for low values of m ($m \approx 6$).

- *Feasible* problems, which are efficiently solved, even though the correct target concept description is not found. More precisely, FOIL learns a concept description $\widehat{h}$ which (a) is predictively accurate (nearly all training and test examples are correctly classified), (b) consists of a single conjunctive hypothesis, as the original target concept description h, and (c) shares many literals with h. However, $\widehat{h}$ is significantly shorter than h (e.g., $\widehat{h}$ involves 9 literals versus 29 in h for problem Π_{26}); in many cases, $\widehat{h}$ largely over-generalizes h. Most feasible problems lie in the NO-region, rather far away from the phase transition. Problems of this kind are Π_{13}, Π_{15}, Π_{17}, Π_{18}, Π_{21}, Π_{22}, Π_{24} to Π_{26}, Π_{33} and Π_{35}.

- *Hard* problems, which are not solved by FOIL. The learned concept description $\widehat{h}$ is the disjunction of many conjunctive hypotheses h_t (between 6 and 15) of various sizes, and each h_t covers only a few training examples. From a learning perspective, over-fitting has occurred (each h_t behaves well on the training set, but its accuracy on the test set is comparable to that of random guessing), related to an apparent *small disjuncts* problem (Holte et al., 1989). Hard problems lie in the PT region or in the NO region, but, in contrast to feasible problems, close to the phase transition.

These results confirm that predictive accuracy may be related only loosely to the discovery of the true concept.

It is clear that in real-world problems there is no way distinguishing between feasible and easy problems, since the true concept is unknown. Again, we shall return to this point later on. A summary of the average results obtained in the YES, NO and PT regions (Table 2) shows that most hard problems are located in the mushy region; conversely, most problems in the PT region are hard.

A second remark concerns the location of the hypotheses h_t learned by FOIL. It is observed that for all learning problems, except the easy ones located in the YES region, *all hypotheses h_t lie inside the phase transition region* (see Figure 7). This is the case no matter whether FOIL discovers one or several conjunctive hypotheses h_t, and whatever the location of the learning problem, lying in the mushy or in the NO region. More precisely:

– when the target concept lies in the mushy region and the problem is easy, FOIL correctly discovers the true concept;

– for feasible learning problems, FOIL discovers a generalization of the true concept, which lies in the mushy region; and

– for hard problems, FOIL retains several seemingly random h_t's, most of which belong to the mushy region.

As previously noted by Giordana and Saitta (2000), the phase transition does behave as an *attractor* for the learning search. Interpretations of this finding will be discussed in Section 5.

						Accuracy [%]		CPU Time
Region	Problem	m	L	$\lvert\widehat{h}\rvert$	m(h_t)	$\mathcal{E}_L$	$\mathcal{E}_T$	[sec]
Y	Π_0	5	15	1	3.00	100	100	10.3
E	Π_1	6	20	1	5.00	100	99.5	21.4
S	Π_2	7	19	1	7.00	100	100	52.3
-	Π_3	8	16	1	8.00	100	100	106.2
r	Π_4	9	15	1	9.00	100	100	69.1
e	Π_5	10	13	1	14.00	100	99	144.2
g	Π_6	10	16	8	$< 10 - 13 >$ 11.75	88	48.5	783.5
i	Π_7	11	13	1	11.00	100	100	92.2
o	Π_8	11	15	6	$< 11 - 16 >$ 13.50	85	53.5	986.2
n	Π_9	12	13	3	$< 13 - 15 >$ 14.00	98.5	83	516.4
	Π_{10}	13	13	1	13.00	100	100	455.9
	Π_{11}	14	12	1	13.00	100	98.5	297.0
	Π_{12}	13	31	13	$< 1 - 8 >$ 4.77	90.5	49.5	1317.3
N	Π_{13}	15	29	1	6.00	100	100	185.3
O	Π_{14}	15	35	2	$< 5 - 7 >$ 6.00	97.5	84.5	894.6
-	Π_{15}	15	38	1	6.00	100	99.5	101.5
r	Π_{16}	16	38	3	$< 5 - 8 >$ 6.33	97.5	90	1170.6
e	Π_{17}	18	24	1	10.00	100	100	196.4
g	Π_{18}	18	35	1	6.00	100	100	201.0
i	Π_{19}	19	26	2	$< 1 - 8 >$ 4.50	100	98.5	298.4
o	Π_{20}	21	18	8	$< 1 - 10 >$ 4.13	81.5	58	1394.9
n	Π_{21}	24	20	1	10.00	100	99.5	252.3
	Π_{22}	25	24	1	6.00	100	99	135.9
	Π_{23}	27	18	10	$< 1 - 13 >$ 5.60	94	72.5	1639.6
	Π_{24}	29	17	1	12.00	100	99.5	144.9
	Π_{25}	29	23	1	10.00	100	99.5	720.5
	Π_{26}	29	24	1	9.00	100	99	618.8
	Π_{27}	6	26	1	6.00	100	100	82.5
	Π_{28}	6	28	12	$< 5 - 11 >$ 8.08	91.5	50.5	815.4
P	Π_{29}	7	27	11	$< 5 - 11 >$ 8.27	92	53	1237.0
T	Π_{30}	7	28	11	$< 1 - 10 >$ 7.64	91.5	60.5	1034.2
-	Π_{31}	8	27	1	7.00	100	100	58.8
r	Π_{32}	11	22	5	$< 1 - 12 >$ 3.20	71.5	70.5	851.0
e	Π_{33}	11	27	1	8.00	99	98.5	250.4
g	Π_{34}	13	21	10	$< 1 - 11 >$ 4.10	85.5	63	1648.2
i	Π_{35}	13	26	1	9.00	100	99	476.8
o	Π_{36}	14	20	5	$< 1 - 11 >$ 4.80	94	88	722.7
n	Π_{37}	14	24	3	$< 7 - 9 >$ 7.67	99	92.5	774.0
	Π_{38}	17	14	8	$< 13 - 17 >$ 15.00	93	46	294.6
	Π_{39}	17	15	9	$< 1 - 13 >$ 5.00	78.5	66	916.8
	Π_{40}	18	16	8	$< 1 - 15 >$ 8.87	91	58.5	404.0
	Π_{41}	19	16	7	$< 1 - 12 >$ 8.14	83.5	60.5	1268.5
	Π_{42}	26	12	3	$< 24 - 25 >$ 24.33	80	58	361.4

Table 1: Hypotheses produced by FOIL for some representative learning problems.

Region	Nb of Pbs	Percentage of pbs solved	Average nb of hyp.		Avg on solved pbs	
			pbs solved	pbs unsolved	Test acc.	CPU time
YES	46	88.1% (37)	1	6.33	99.61	74.05
NO	195	72.8% (142)	1.27	8.28	99.61	385.43
PT	210	28.1% (59)	1.10	8.18	99.12	238.25
Total	451	52.8% (238)	1.12	7.60	99.45	232.58

Table 2: Summary of the experiments. Easy and feasible learning problems (Solved Pbs) are distinguished from hard problems (Unsolved Pbs).

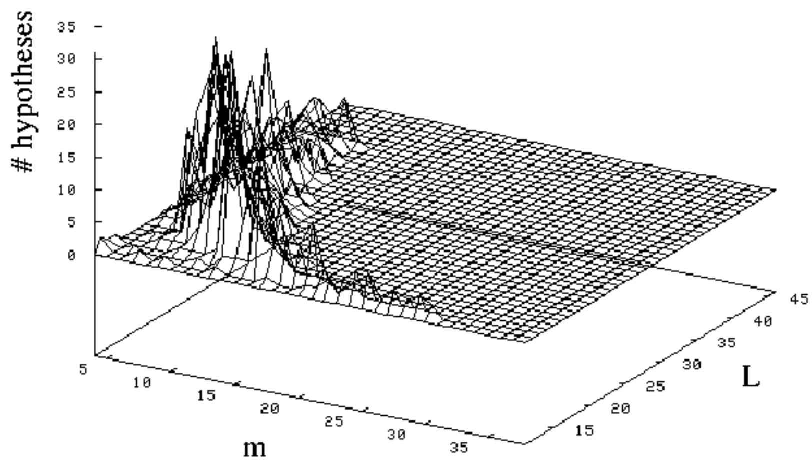


Figure 7: Location distribution of conjunctive hypotheses h_t learned by FOIL, centered on the phase transition region.

4.3 Computational Complexity

The computational complexity of FOIL's search depends mostly on two factors: the number of generated hypotheses, and the average number of their models[12] in the examples.

For easy problems, one single hypothesis h_t ($\widehat{h} \approx h$) is constructed; the computational cost remains low, even though it increases, as expected, when the average number of models of h_t tends to one.

For feasible problems, one single hypothesis h_t ($\widehat{h}$ most often overgeneralizes h) is constructed, as well. In most cases, the computational cost is very low, and the average number of models is very high.[13]

Finally, in the case of hard problems, many hypotheses h_t are constructed and the computational cost is always very high. This might be explained as most h_t lie in the phase transition region, and some of them admit just one single model in the examples.

12. "Models" correspond to "alternative ways" of satisfying a hypothesis in an example.
13. A single exception can be seen in Table 1: for the learning problem Π_{25}, the average number of models is one, and the computational cost is high.

Everything else being equal, the learning cost is higher for problems in the NO region. One for this higher complexity is the size of the hypothesis space, which exponentially increases with the number m of literals in h; this causes many more hypotheses to be considered and tested in each learning step. Another cause is that the NO region includes many hard problems (see Figure 6); on such problems, the PT region is visited again and again because many hypotheses are tried.

4.4 Comparison to other Learning Algorithms

The results presented so far have been obtained by FOIL. A natural question arises about the bias of the specific learner employed.

In order to explore this issue, SMART+ and G-Net have been tested on the learning problems reported in Figure 6. SMART+ uses a beam search strategy so that the system runs slower than FOIL in proportion to the size of the beam. G-Net uses an elitist genetic algorithm, which repeats a cyclic procedure that, at every iteration, explores a new inductive hypothesis, until it converges to a stable solution. On the considered set of learning problems, a stable solution is reached only after many thousands of iterations. As each iteration requires testing the newly created hypothesis on the whole learning set, G-Net's computational time is heavily affected by the presence of the complexity peak in the mushy region. In the present experiments, the number of iterations has been fixed to 50,000 for all runs. With this setting, the CPU time may range from several hours to several days for a single problem.

Thus, due to the high computational cost, the comparison has been restricted to the subset of problems reported in Table 1. The results are reported in Table 3. It appears that FOIL and SMART+ are discordant on 7 problems out of 43, in the sense that the same problem has been solved by one of the algorithms but not by the other. Nevertheless, such problems appear to be on the border of the blind spot (see Figure 6), where FOIL itself alternates successes and failures.

On all the other cases, there is a substantial agreement. In all runs the beam width of SMART+ has been fixed to 5. Increasing the width of the beam and granting very large computational resources, SMART+ can solve a few more problems. However, it is unrealistic to use this approach systematically. On the other hand, G-Net solved fewer problems than the other two systems. In fact, G-net would need a higher number of iterations in order to converge, but this was prohibitive because of the computational complexity. Tuning G-Net control parameters did not bring any significant improvement.

SMART+ has also been applied to other problems (about 100) sampled randomly inside and outside the blind spot, but far from the borders; in all cases successes and failures have been in agreement with FOIL.

5. Interpretation

This section proposes an interpretation of the results reported so far. The discussion focuses on three main questions: why is the learning search captured by the PT region? When and why does relational learning miss the true target concept? When and why should relational learning fail to find any accurate approximation of the target concept?

				SMART+					G-NET			
						Accuracy [%]		CPU Time			Accuracy [%]	
Region	Problem	m	L	$\|\widehat{h}\|$	m(h_t)	$\mathcal{E}_L$	$\mathcal{E}_T$	[sec]	$\|\widehat{h}\|$	m(h_t)	$\mathcal{E}_L$	$\mathcal{E}_T$
Y	Π_0	5	15	1	3	100	99	130	1	3	100	99
E	Π_1	6	20	1	6	100	99.5	429	1	5	100	99.5
S	Π_2	7	19	1	7	99.5	100	667	21	9	100	73.5
-	Π_3	8	16	1	8	100	100	218	14	12	100	74
r	Π_4	9	15	1	9	100	100	99	13	16	100	67
e	Π_5	10	13	1	10	100	100	609	1	16	100	98
g	Π_6	10	16	-	-	-	-	1305	21	12	100	58.5
i	Π_7	11	13	1	11	100	100	509	2	12	100	94.5
o	Π_8	11	15	-	-	-	-	592	22	18	100	50.5
n	Π_9	12	13	-	-	-	-	418	1	10	100	95
	Π_{10}	13	13	1	13	100	100	3368	24	12	100	48.5
	Π_{11}	14	12	1	14	100	100	1935	1	13	100	98.5
	Π_{12}	13	31	1	8	96.5	98	626	1	7	100	100
N	Π_{13}	15	29	1	7	99.5	100	1081	34	6	100	55.5
O	Π_{14}	15	35	4	7	100	97.5	743	29	6	100	68
-	Π_{15}	15	38	11	8	100	98	882	36	6	100	60
r	Π_{16}	16	38	2	7	98.5	100	514	1	6	100	99
e	Π_{17}	18	24	1	9	100	99.5	1555	28	8	100	54.5
g	Π_{18}	18	35	1	7	96	99	590	1	6	99.5	98.5
i	Π_{19}	19	26	8	8	100	99.5	1410	33	7	100	53
o	Π_{20}	21	18	1	12	100	99.5	2396	22	10	100	47.5
n	Π_{21}	24	20	10	10	99	93	2034	27	8	100	52
	Π_{22}	25	24	1	8	99	99.5	2331	30	6	99.5	57.5
	Π_{23}	27	18	1	10	100	97	3289	26	13	96.5	50.5
	Π_{24}	29	17	24	12	97.5	75	7004	27	11	100	53
	Π_{25}	29	23	1	9	100	99.5	2241	35	8	100	46.5
	Π_{26}	29	24	2	9	100	100	2887	30	8	97	51
	Π_{27}	6	26	-	-	-	-	-	17	9	99	86
	Π_{28}	6	28	1	6	97.5	100	932	34	8	100	51.5
P	Π_{29}	7	27	-	-	-	-	-	42	7	100	50.5
T	Π_{30}	7	28	1	7	98.5	100	396	32	8	100	57
-	Π_{31}	8	27	5	8	99	93.5	640	19	8	100	81.5
r	Π_{32}	11	22	-	-	-	-	-	30	8	100	50
e	Π_{33}	11	27	-	-	-	-	-	37	7	100	54.5
g	Π_{34}	13	21	-	-	-	-	-	33	7	99	49.5
i	Π_{35}	13	26	1	8	98.5	98	864	1	7	100	100
o	Π_{36}	14	20	-	-	-	-	-	29	9	100	44.5
n	Π_{37}	14	24	-	-	-	-	-	32	8	100	59
	Π_{38}	17	14	-	-	-	-	4371	24	15	100	47.5
	Π_{39}	17	15	-	-	-	-	-	21	15	97	56
	Π_{40}	18	16	25	14	96	53	24172	24	13	100	46
	Π_{41}	19	16	-	-	-	-	-	21	13	100	52.5
	Π_{42}	26	12	-	-	-	-	14273	29	10	99.5	47

Table 3: Hypotheses produced by SMART+ and G-Net for the same learning problems reported in Table 1. Smart+ uses beam search, and the beam width was set to 5. Symbol '-', means that the learning process reached the maximum allowed time of 6 hours, without finding a solution. G-Net always run for a number of iterations fixed to 50,000.

5.1 Why the Phase Transition Attracts the Learning Search

Being a top-down learner, FOIL constructs a series of increasingly specific candidate hypotheses, $h_1, \ldots, h_t$. The earlier hypotheses in the series belong to the YES region by construction.

If the most specific hypothesis h_i built up in the YES region is not satisfactory according to the stop criterion (see below), FOIL moves into the PT region, and h_{i+1} belongs to it. It might also happen that the most specific hypothesis h_j $(j > i)$ in the PT region is not satisfactory either; FOIL then comes to visit the NO region.

Let us consider the stop criterion used in FOIL. The search is stopped when the current hypothesis is *sufficiently correct*, covering no or few negative examples; on the other hand, at each step, the current hypothesis is required to be *sufficiently complete*, covering a *sufficient number* of positive examples. The implications of these criteria are discussed, depending on the location of the target concept description h with respect to the phase transition.

Case 1: h belongs to the PT region.
By construction, the target concept description h covers with probability close to 0.5 any random example (Section 2.2); therefore, the repair mechanism that ensures the dataset is balanced is not employed (Section 3.1). It follows that:

- Since any hypothesis in the YES region almost surely covers any random example, it almost surely covers all training examples, both positive and negative. Therefore the search *cannot stop* in the YES region, but must proceed to visit the PT region.

- Symmetrically, any hypothesis in the NO region almost surely rejects (does not cover) any random example; then, almost surely, it will cover no training examples at all. Although these hypotheses are correct, they are too incomplete to be acceptable. Therefore, the search must stop before reaching the NO region. As a consequence, FOIL is bound to produce hypotheses h_t lying in the PT region.

Case 2: h belongs to the NO region.
In this case, randomly constructed examples are almost surely negative (see Section 3.1). It follows that any hypothesis in the YES region will almost surely cover the negative examples; this implies that the search cannot stop in the YES region. On the other hand, any hypothesis in the NO region will almost surely be correct (covering no negative examples); therefore, there is no need for FOIL to go deeply into the NO region. Hence, FOIL is bound to produce hypotheses h_t lying in the PT region, or on the verge of the NO region.

Case 3: h belongs to the YES region.
The situation is different here, since there exist correct hypotheses in the YES region, namely the target concept itself and possibly many other hypotheses more specific than it. Should these hypotheses be discovered (the chances for such a discovery are discussed in the next subsection), the search could stop immediately. But in all cases, the search must stop before reaching the NO region, for the following reason. As h belongs to the YES region, randomly constructed examples are almost surely positive examples (see Section 3.1). This implies that any hypothesis in the NO region will almost surely reject the positive examples, and will therefore be considered insufficiently complete. In this case again, FOIL is bound to produce hypotheses h_t in the YES region or in the PT region.

In conclusion, FOIL is unlikely to produce hypotheses in the NO region, whatever the location of the target concept description h is, at least when the negative examples are uniformly distributed. Most often, FOIL will produce hypotheses belonging to the PT region, but it might produce a hypothesis in the YES region if h itself belongs to it. It is worth noting that such a behavior has also been detected in several real-world learning problems (see Giordana and Saitta, 2000).

Analogous considerations hold for SMART+, and, more generally, for any top-down learner: as maximally general hypotheses are preferred, provided that they are sufficiently discriminating, searching in the NO region does not bring any benefit. It follows that the phase transition *behaves as an attractor for any top-down relational learner*.

The experiments done with G-Net indicate that the same conclusion also holds for GA-based learning, notwithstanding the strong difference between its search strategy and the top-down one. The explanation offered for this finding is the following. The starting point in genetic search (the

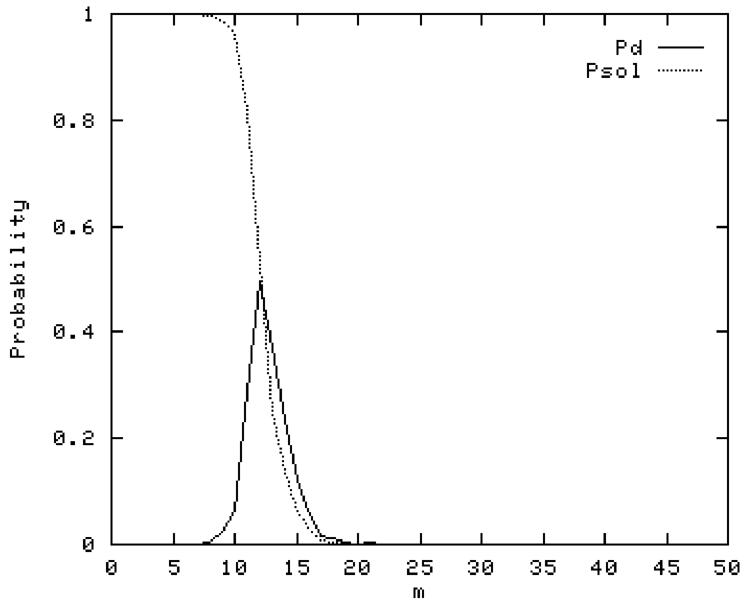


Figure 8: Probability P_d of a hypothesis discriminating any two examples, compared to the probability P_{sol} of a hypothesis covering a random example. The crossover point is at $P_{sol} = .5$. P_d and P_{sol} are estimated as an average from 90,000 randomly generated pairs (hypothesis, example), with L = 16.

initial population of solutions) consists of random hypotheses, which are distributed on the horizontal line $L = |\mathbf{A}|$. Afterward, the evolutionary search focuses on the most fit hypotheses, i.e., where the fitness function favors the most *discriminating* and *simple* hypotheses.

On one hand, discriminating hypotheses are mostly found close to the phase transition (see Figure 8). On the other hand, since simple hypotheses score higher than complex ones, everything else being equal, the genetic search will favor hypotheses close to the phase transition, on the verge of the NO region. Like FOIL and SMART+, G-Net will most often produce hypotheses in the PT region (see Giordana and Saitta, 2000, for details).

In retrospect, this general attraction toward the phase transition can be explained if one looks for hypotheses able to separate positive from negative examples. In fact, the probability that a random hypothesis separates any two examples reaches its maximum in the PT region, and more precisely at the crossover point (see Figure 8).

5.2 Correct Identification of the Target Concept

Given that the hypotheses selected by the learner are most often close to the phase transition, let us examine why and when these hypotheses might differ from the true target concept description h.

When h belongs to the mushy region, two possibilities have been observed (Table 1). If h involves few literals ($m \leq 6$), then it is correctly identified. Otherwise, several hypotheses h_t are retained, each h_t covering a few positive training examples, and their disjunction $\widehat{h}$ performs very poorly on the test set.

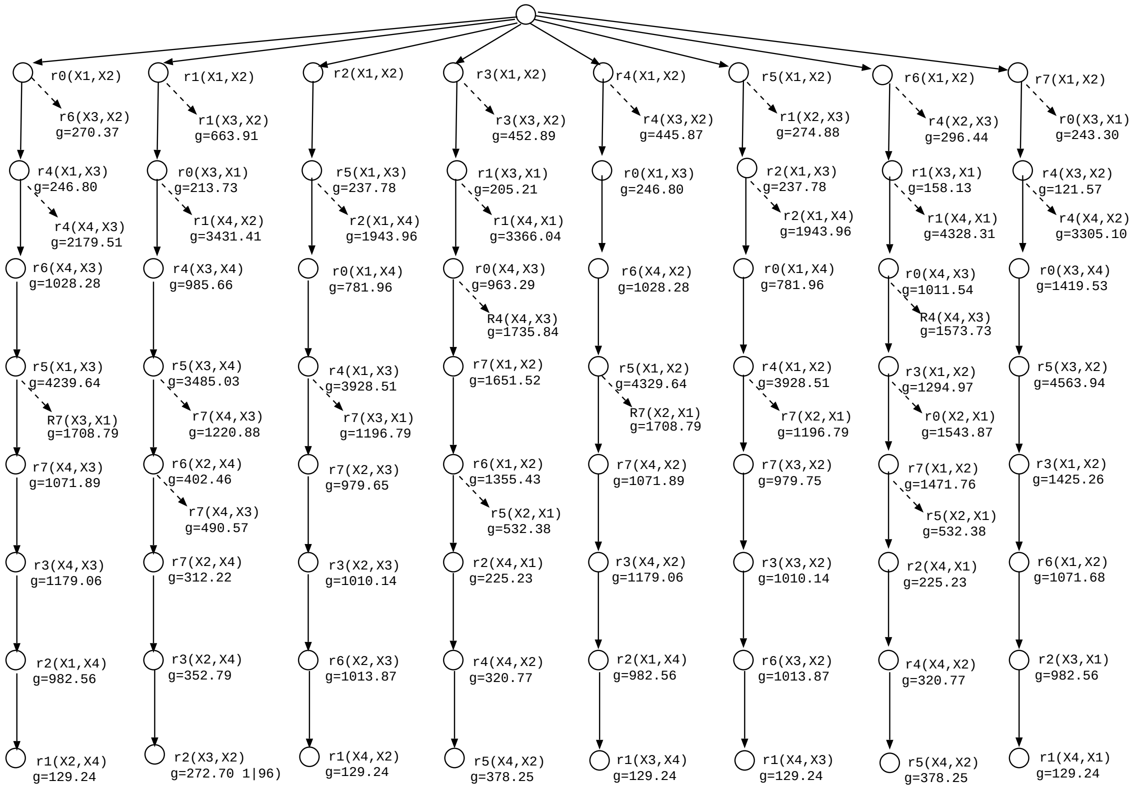


Figure 9: Visiting the specialization tree for problem $\Pi_{8,20}$. All specialization steps misled by the information gain maximization are indicated with oblique dashed arrows; the incorrect best literal is given with the associated information gain. In such cases, the choice is forced to the best correct literal (indicated with a vertical plain arrow, together with the associated information gain).

The reasons why a top-down learner should fail to identify a *long* target concept ($m > 6$) are illustrated with an example. Let us consider the target concept description h for problem $\Pi_{8,20}$ ($m = 8$, and $L = 20$), which belongs to the mushy region:

$$\begin{aligned} h(X_1,X_2,X_3,X_4) \;=\; & r_0(X_1,X_2), r_1(X_2,X_3), r_2(X_2,X_3), r_3(X_3,X_4), \\ & r_4(X_1,X_4), r_5(X_1,X_4), r_6(X_3,X_4), r_7(X_3,X_4) \end{aligned}$$

The top-down search proceeds by greedily optimizing the information gain. The choice for the first literal is indeterminate[14] since any predicate has, by construction, $N = 100$ models in every example. But this does not penalize the search, as any choice is relevant since all predicates appear in h by construction. All eight specialization paths, corresponding to all possible choices for the first literal, are thus considered in parallel (Figure 9).

After a first literal (say $h_1 = r_0(X_1,X_2)$) has been selected, the information gain of all literals connected to h_1 is computed, and the one with maximum information gain is retained. Unfortu-

14. In a real-world application, the first literal is selected on the basis of pure attribute-value-like information: the information gain only depends on the number of occurrences of a predicate symbol in positive/negative examples.

nately, it turns out that the best literal according to this criterion (e.g., $r_6(X_3,X_2)$ with gain 270.37) is *incorrect*, i.e., it is such that hypothesis $h_2 = r_0(X_1,X_2), r_6(X_3,X_2)$ does *not* generalize h; hence, the search cannot recover and will fail, unless backtracking is used (see below).

On this particular problem, the maximization of the information gain appears to be seriously misleading. In all but one of the eight specialization paths, the first specialization step (regarding the second literal) fails as FOIL selects incorrect literals (displayed in Figure 9 with a dashed oblique arrow, together with the corresponding information gain value).

When a specialization choice is incorrect, FOIL must either backtrack, or end up with an incorrect hypothesis h_t. In order to see the total amount of backtracking needed to find the true target concept description h, let us manually correct hypothesis h_2, and replace the wrong literal selected with the best correct literal (literal with maximum information gain such that h_2 does generalize the true target concept). The best correct literal is indicated with a solid vertical arrow in Figure 9, together with the corresponding information gain; clearly, the best correct literal appears to be often poor in terms of information gain.

Unfortunately, it appears that forcing the choice of a correct second literal is not enough; even though h_2 is correct, the selection of the third literal is again misled by the information gain criterion, in all branches but one. To pursue the investigation, let us force again the choice of the best correct i-th literal, in all cases where the optimal literal with respect to information gain is not correct. All repairs needed are reported in Figure 9.

These considerations show that greedy top-down search is most likely bound to miss the true target concept, as there is no error-free specialization path for this learning problem.

5.3 Impact on the Search Strategy

According to Figure 9, a large amount of backtracking would be needed in order to discover the true target concept from scratch. More precisely, the information gain appears to be reliable in the late stages of induction, provided that the current hypothesis is correct (in the case of Figure 9, the search needs to be "seeded" with four correct literals). In other words, the information gain criterion can be used to transform an *informed* guess (the first four literals) into an accurate hypothesis, if and only if the guess has reached some critical size (in the particular case of problem $\Pi_{8,20}$, the critical size corresponds to half the size of the true concept).

Let us define the size of the informed guess m_k as the minimum number of literals such that, with probability 0.5, FOIL finds the target concept description h, or a correct generalization of it, by refining a m_k-literal guess.

Figure 10 reports the critical size $m_k(m,L)$ of an informed guess for all problems $\Pi_{m,L}$ within or close to the mushy region, obtained, as for problem $\Pi_{8,20}$, by a systematic backtracking. Figure 10 could thus be interpreted as a *reliability map* of the information gain: high values of $m_k(m,L)$ indicate poor reliability.

These empirical limitations of the information gain criterion can be explained by the phase transition paradigm. Let us consider the sequence of hypotheses explored by FOIL (or SMART+). When the current hypothesis h_i belongs to the YES region, it covers any example with a number of models that exponentially increases with the number of variables in h_i, regardless of the example label. This fact masks the distinction between correct and incorrect literals: the signal-to-noise ratio is very low.

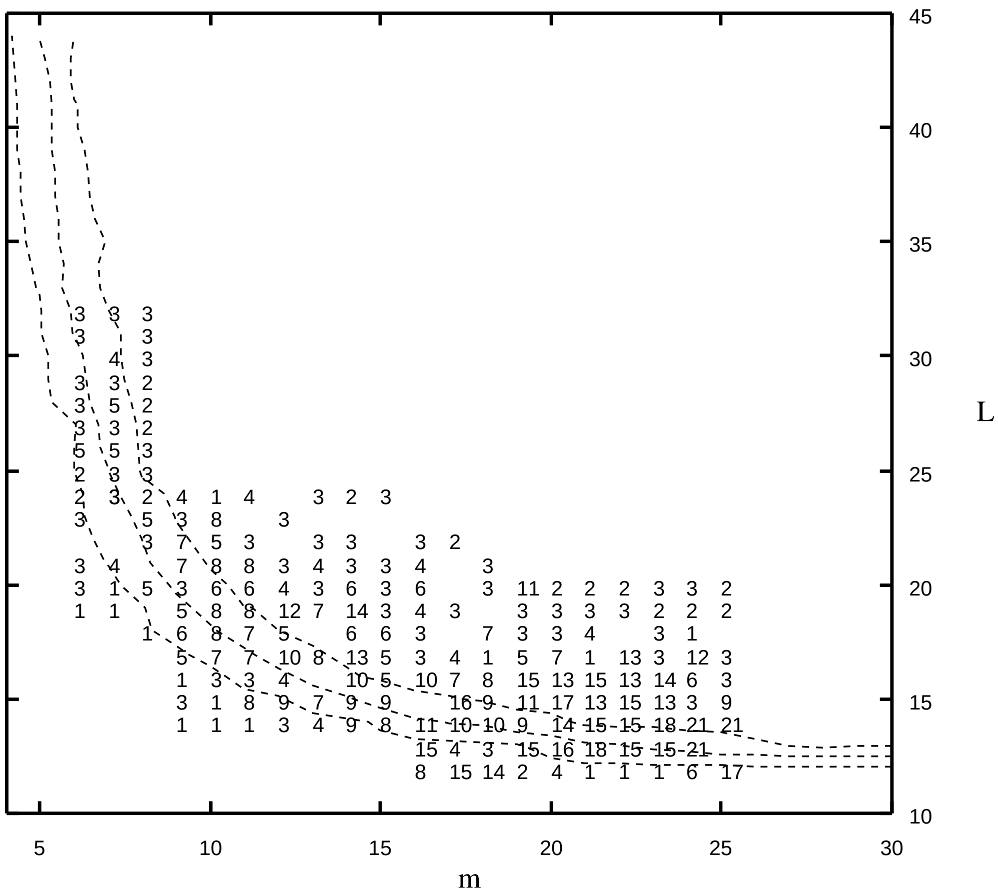


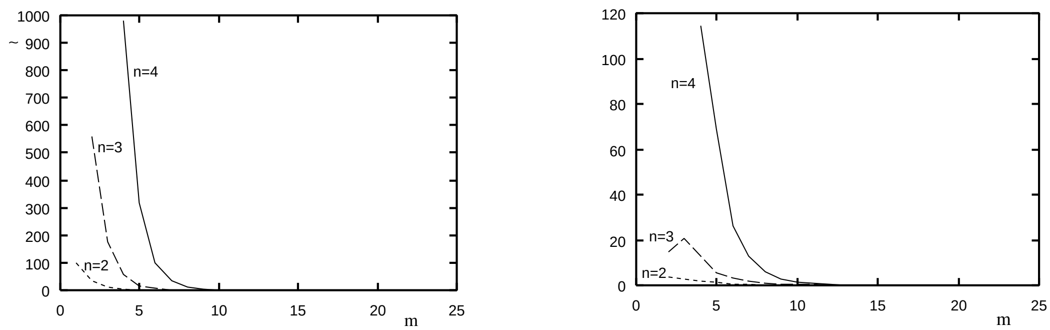
Figure 10: Number of correct literals to be "generated" before information gain becomes reliable.

When the search enters the PT region, the information gain criterion becomes effective, and guides the learner toward one among many discriminating hypotheses. However, the selected discriminating hypothesis may significantly differ from the true target concept, due to earlier incorrect choices.

To back up these considerations, the average and the standard deviation of the number of models of a hypothesis h_t in an example E have been experimentally measured. Figure 11(a) reports the average number of models for random h_t and E, versus the number m of literals in h_t. Hypotheses involving 2, 3 and 4 variables have been considered; it appears that the number of models decreases very quickly as one approaches the phase transition. Figure 11(b) shows the standard deviation of the number of models, which is very high for all h_t in the YES region.

5.4 Correct Approximation of the Target Concept

According to the above discussion, the target concept description h should have hardly any chance to be correctly identified through top-down learning, when either its size m or the number L of constants in the application domain are large, which is the case for all problems in the NO region.



(a) Average number of models

(b) Standard deviation of the number of models.

Figure 11: Number of models in a random h_t in a random E, versus the number m of literals in h_t.

On the contrary, it is observed that FOIL does succeed in finding highly accurate hypotheses (Figure 6) for many problems in the NO region, when both the target concept is large, and the examples involve many constants (upper right region, large values of m and L). A closer inspection shows that this is the case when m is more than twice the critical value m_{cr}, where the horizontal line $L = |\mathbf{A}|$ crosses the phase transition line.

In order to see why this happens, let us consider a learning problem in the NO region. As the size m of the target concept increases, so does the amount of modifications needed to transform a random example into a positive one (Section 3.1). The underlying distributions of the positive and negative examples are increasingly different, which intuitively explains why it becomes easier to separate them.

More formally, let us consider a generalization gen_c of the target concept; by construction gen_c is complete, i.e., it covers all positive examples. On the other hand, if gen_c belongs to the NO region, it almost surely rejects all random examples, and negative examples in particular (the argument closely follows the one in Section 5.1). All generalizations of h in the NO region are thus *complete* and almost surely *correct*. Hence, if the learner ever discovers a generalization gen_c of the target concept close to the NO region, the learning search stops because gen_c behaves perfectly on the training set; as gen_c behaves perfectly on the test set as well, learning has succeeded. From the standpoint of predictive accuracy, the success of relational learning thus depends on the probability of finding a generalization gen_c of h on the edge of the phase transition.

Let m and g denote the number of literals of h and gen_c, respectively. The number of g-literal generalizations of h, denoted $H(g,m)$, has been analytically computed (see Appendix A). As expected, $H(g,m)$ reaches its maximum for $g = \frac{m}{2}$. Figure 12 reports $H(g,m)$ versus g, for $m = 22$.

From Figure 13 one can see that the number of generalizations starts growing very fast as m increases, and that half of them belong to the PT region when m is greater than twice the critical value m_{cr}.

Both considerations explain why relational learning is more likely to succeed when the size m of the target concept increases *and* is at least twice as great as the critical value m_{cr}.

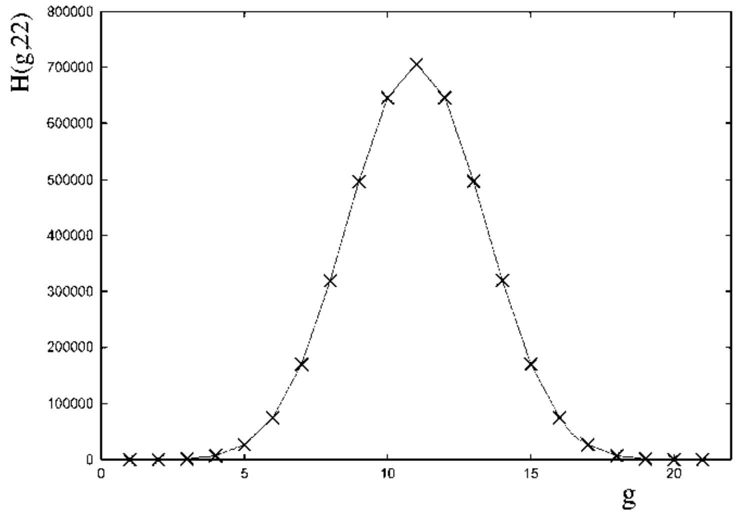


Figure 12: Number $H(g,22)$ of g-literal generalizations of a 22-literal target concept description h *versus* g.

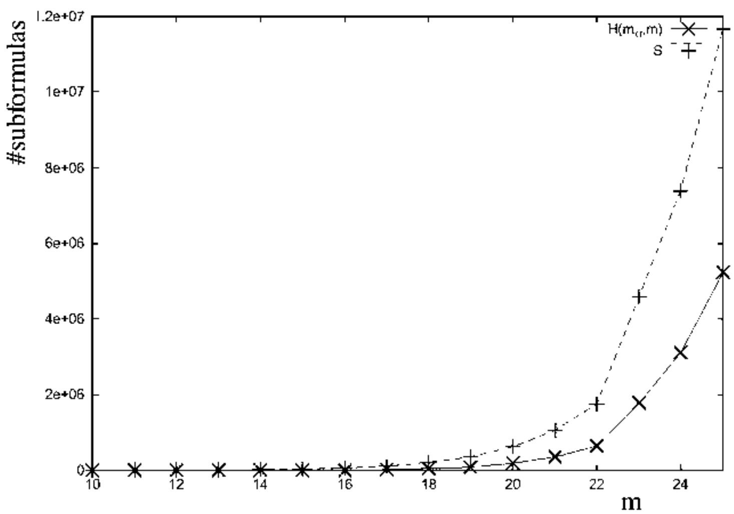


Figure 13: Number $H(m_{cr},m)$ of m_{cr}-literal generalizations of a m-literal target concept description h, and number S of all g-literal generalizations of h for $g \leq m_{cr}$.

6. Discussion and Conclusions

This paper presents a novel perspective on relational learning, based on the distributional analysis of computational complexity, developed for combinatorial search and constraint satisfaction problems (Hogg et al., 1996b). Following this new paradigm, experiments using a set of artificial problems delivers two major lessons.

The first lesson concerns the existence of an attractor for relational learning, namely the PT region. This behavior is not specific to artificial learning problems; it also occurs in real-world problems (King et al., 1995; Giordana et al., 1993), as reported by Giordana and Saitta (2000). This behavior is observed for top-down and GA-based search strategies; it is explained by the common learning bias toward simplicity (Occam's Razor), and the fact that discriminating hypotheses mostly lie close to the phase transition.

As the PT region concentrates the (empirically) most complex problem instances, it follows that relational learning cannot sidestep the complexity barrier. How to scale up current algorithms (e.g., in order to learn concepts with more than four variables) thus becomes an open question. Indeed, the most complex applications of relational learning that are described in the literature refer to concepts with few literals and few variables (King et al., 1995; Giordana et al., 1993; Dolsak et al., 1998).

The second lesson concerns the existence of a "blind spot" for all relational learners considered in the present study. Whatever the learning problem in this area, the hypotheses extracted from the training set behave as random guesses on the test set. This blind spot adversely affects the scalability of relational learning, too.

Actually, both results are explained by the same causes, related to the average number of models, associated to a pair (hypothesis, example), and its variance. More precisely, this number of models is high for small-size hypotheses; the existing search criteria (e.g., information gain or minimum description length) turn out to mislead the search due to the high variance of this number. Meanwhile, the structure of the substitution tree explains both the high computational cost in the PT region, and the fact that discriminating hypotheses belong to it. Of course, the presence of noisy examples in the training set, and/or irrelevant predicates in the description of the examples, would only make things worse.

These results are worrying, as they suggest that current techniques cannot easily scale up and acquire concepts more complex than those described in the current literature. Meanwhile, the need for relational learning gets stronger in various large size application domains (e.g., learning from visual data; learning from numerical engineering-related data). New approaches should therefore be devised to address the above limitations.

One perspective for further research concerns the use of bottom-up search strategies for learning concepts in the NO region (that is, for characterizing rare events). The difficulty is twofold. First of all, bottom-up learners are commonly considered less robust than top-down learners (being sensitive to the order of consideration of the examples on one hand, and to noisy examples on the other hand). Secondly, bottom-up operators such as least general generalization (lgg) or reduced lgg (Nienhuys-Cheng and de Wolf, 1997) are not directly applicable to large size examples. The artificial examples considered in this paper globally contain from 500 to 3,000 tuples (ground literals, in the ILP terminology) in their tables; such sizes are realistic in a numerical engineering context, where meshes involve up to some hundred thousands finite elements.

An alternative might be to exploit any prior knowledge available, and to start the learning search with a correct partial hypothesis (informed guess). In this way, the chance of the search being misled would be significantly decreased. The limitation is that, even though prior knowledge would significantly help in finding an accurate solution, the computational complexity of assessing any candidate hypothesis would remain the same.

A third alternative could be to reconsider the learning strategy *and* the search criterion. More specifically, it would make sense to restrict the exploration to the PT region, given that most relevant hypotheses lie there.

Appendix A.

Let c be a target concept whose description h contains m literals, built on m distinct binary predicate symbols. The number $H(g,m)$ of g-literal generalizations of h with the same number of variables as $h$is computed according to the following reasoning.

Given the number of variables n occurring in h, the number of possible variable pairs (X_i, X_j) in a literal of h is $K = n^2$. Any g-literal, n-variable concept description with the same predicate symbols as h is a subset of by h if and only if all its variables are chained as in h.

Let r_j $(1 \leq j \leq K)$ denote the number of literals in h involving a given pair of variables. By construction,

$$\sum_{j=1}^{K} r_j = m \tag{7}$$

A g-literal generalization of h can be obtained by selecting s_j literals from the r_j's $(1 \leq j \leq K)$, such that $\sum_{j=1}^{K} s_j = g$.

Then, the number of g-literal, n-variable generalizations of h (up to variable renaming) is given by:

$$H(g,m) = \sum_{j=1}^{K} \sum_{s_j=0}^{min(r_j, g-\sum_{i=1}^{j-1} s_i)} \binom{r_K}{v - \sum_{i=1}^{K-1} s_i} \prod_{i=1}^{K-1} \binom{r_i}{s_i}. \tag{8}$$

Let us notice that $H(g,m)$ is maximum when the numbers s_i are roughly equal to $\frac{r_i}{2}$, that is, g is about $\frac{m}{2}$.

References

D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, and M.S.O. Molloy. Random constraint satisfaction: A more accurate picture. In *Third International Conference on Principles and Practice of Constraint Programming*, pages 107–120. Springer, 1997.

C. Anglano, A. Giordana, G. Lobello, and L. Saitta. An experimental evaluation of coevolutive concept learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 19–23, Madison, WI, July 1998.

F. Bergadano, A. Giordana, and L. Saitta. *Machine Learning: An Integrated Framework and its Applications*. Ellis Horwood, Chichester, UK, 1991.

M. Botta and A. Giordana. SMART+: A multi-strategy learning tool. In *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 937–943, Chambéry, France, 1993.

M. Botta, A. Giordana, and L. Saitta. An experimental study of phase transitions in matching. In *Proceedings of th 16th International Joint Conference on Artificial Intelligence*, pages 1198–1203, Stockholm, Sweden, 1999.

M. Botta, A. Giordana, L. Saitta, and M. Sebag. Relational learning: Hard problems and phase transitions. In *Proceedings of the 6th National Italian Conference on Artificial Intelligence*, volume LNAI 1792, pages 178–189. Springer-Verlag, 2000.

P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings 12th International Joint Conference on Artificial Intelligence*, pages 331–337, Sidney, Australia, 1991.

W. Cohen. A PAC-learning algorithm for a restricted class of recursive logic programs. In *Proceedings 10th National Conference On Artificial Intelligence*, Washington, DC, 1993.

J.M. Crawford and L.D Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–58, 1996.

C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, 1995.

T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10 (7):1895–1924, 1998.

B. Dolsak, I. Bratko, and A. Jezernik. Application of machine learning in finite element computation. In R.S. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining*, pages 147–171. Wiley, 1998.

S. Džeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proceedings of COLT-92*, pages 128–134, Pittsburgh, PA, 1992.

I.P. Gent and T. Walsh. The TSP phase transition. *Artificial Intelligence*, 88:349–358, 1996.

A. Giordana and L. Saitta. Phase transitions in relational learning. *Machine Learning*, 41(2):17–251, 2000.

A. Giordana, L. Saitta, and F. Bergadano. Enigma: A system that learns diagnostic knowledge. *IEEE Transactions on Knowledge and Data Engineering*, KDE-5:15–28, February 1993.

A. Giordana, L. Saitta, M. Sebag, and M. Botta. Analyzing relational learning in the phase transition framework. In *17th International Conference on Machine Learning*, pages 311–318, Stanford, CA, July 2000. Morgan Kaufmann.

T. Hogg, B.A. Huberman, and C.P. Williams, editors. *Artificial Intelligence: Special Issue on Frontiers in Problem Solving: Phase Transitions and Complexity*, volume 81(1-2). Elsevier, 1996a.

T. Hogg, B.A. Huberman, and C.P. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996b.

R. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, volume 1, pages 813–818, Detroit, MI, 1989. Morgan Kaufmann.

M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

R. Khardon. Learning first order universal Horn expression. In *Proceedings of COLT-98*, pages 154–165, Madison, WI, 1998.

J.U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14:193–218, 1994.

R.D. King, A. Srinivasan, and M.J.E. Stenberg. Relating chemical activity to structure: an examination of ILP successes. *New Generation Computing*, 13:411–433, 1995.

J. Maloberti and M. Sebag. Theta-subsumption in a constraint satisfaction perspective. In *Proceedings of Inductive Logic Programming*, pages 164–178. Springer Verlag, LNAI 2157, 2001.

R. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134, Los Altos, CA, 1983. Morgan Kaufmann.

D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *10th National Conference on Artificial Intelligence*, pages 459–465. MIT Press, 1992.

T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

T.M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, London, UK, 1992.

S. Muggleton. Inverse entailment and PROGOL. *New Gen. Comput.*, 13:245–286, 1995.

S. Muggleton and L. De Raedt. Inductive logic programming: theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*. Academic Press, London, UK, 1992.

S. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. Springer Verlag, 1997.

M.J. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9: 57–94, 1992.

G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

P. Prosser. An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81:81–110, 1996.

R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

R.J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

M. Sebag and C. Rouveirol. Resource-bounded relational reasoning: Induction and deduction through stochastic matching. *Machine Learning*, 38:41–62, 2000.

B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

T. Walsh. The constrainedness knife-edge. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 406–411, Madison, WI, 1998.

C.P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.