Mach Learn (2008) 70: 135–150
DOI 10.1007/s10994-007-5031-9

On the connection between the phase transition of the covering test and the learning success rate in ILP

Erick Alphonse · Aomar Osmani

Received: 25 September 2007 / Accepted: 9 October 2007 / Published online: 8 November 2007

Abstract It is well-known that heuristic search in ILP is prone to plateau phenomena. An explanation can be given after the work of Giordana and Saitta: the ILP covering test is NP-complete and therefore exhibits a sharp phase transition in its coverage probability. As the heuristic value of a hypothesis depends on the number of covered examples, the regions "yes" and "no" represent plateaus that need to be crossed during search without an informative heuristic value. Several subsequent works have extensively studied this finding by running several learning algorithms on a large set of artificially generated problems and argued that the occurrence of this phase transition dooms every learning algorithm to fail to identify the target concept. We note however that only generate-and-test learning algorithms have been applied and that this conclusion has to be qualified in the case of data-driven learning algorithms. Mostly building on the pioneering work of Winston on near-miss examples, we show that, on the same set of problems, a top-down data-driven strategy can cross any plateau if near-misses are supplied in the training set, whereas they do not change the plateau profile and do not guide a generate-and-test strategy. We conclude that the location of the target concept with respect to the phase transition alone is not a reliable indication of the learning problem difficulty as previously thought.

Keywords Inductive logic programming · Heuristic search · Plateau phenomena · Phase transition · Near-miss examples

1 Introduction

It was discovered early on that learning in relational languages had to face important *plateau phenomena* (see e.g. Hayes-Roth and McDermott 1977; Quinlan 1991; Silverstein and Pazzani 1991; Richards and Mooney 1992): the evaluation function, used to prioritize nodes

Editors: Stephen Muggleton, Ramon Otero, Simon Colton.

E. Alphonse (✉) · A. Osmani
LIPN-CNRS UMR 7030, Université Paris 13, Paris, France
e-mail: alphonse@lipn.univ-paris13.fr

A. Osmani
e-mail: ao@lipn.univ-paris13.fr

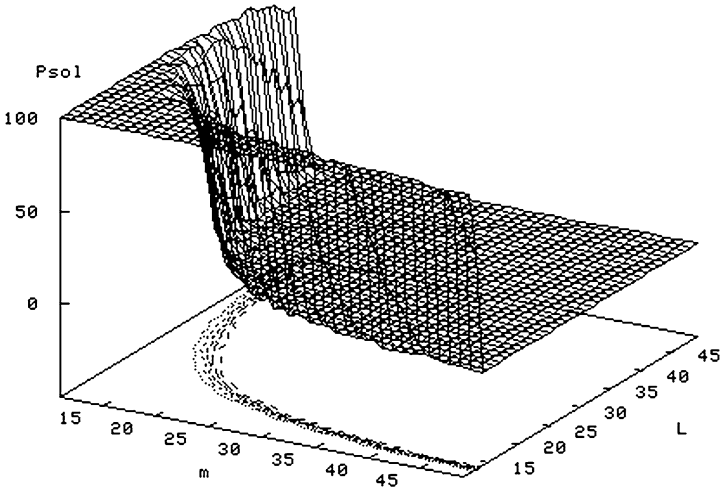

Fig. 1 From Giordana and Saitta (2000): Coverage probability Psol of an example with L constants by a hypothesis with m literals. The contour level plots, projected onto the plane (m, L), correspond to the region where *Psol* is between 0.99 and 0.01

in the refinement graph is constant in parts of the search space, and the search goes blind. These plateau phenomena are the pathological case of heuristic search, complete or not (Pearl 1985). An explanation can be given after the work of (Giordana and Saitta 2000), who studied the ILP covering test within the phase transition framework. As illustrated in Fig. 1, the covering test is NP-complete and therefore exhibits a sharp phase transition in its coverage probability (Cheeseman et al. 1991). When one studies the probability of covering a random example of a fixed size by a hypothesis given the hypothesis' size, one distinguishes three well-identified regions: a region named "yes", where the probability of covering an example is close to 1, a region named "no", where the probability is close to 0, and finally inbetween the phase transition, named the "mushy" or the "pt" region, where an example may or may not be covered. As the heuristic value of a hypothesis depends on the number of covered examples (positive or negative), we see that the two regions "yes" and "no" represent plateaus that need to be crossed during search without an informative heuristic value.

To investigate the impact of the occurrence of a phase transition in the covering test on the learning success rate, systematic experiments with several learning algorithms (FOIL, G-Net, Smart+ and PROGOL) have been conducted on a large set of artificially generated problems by Botta et al. (2003). The authors generated a set of 451 problems by choosing each target concept according to its location in the (m, L) plane with respect to the phase transition, as shown in Fig. 1. The "yes", "no" and "pt" regions are uniformly visited by varying (m, L) pairs without replacement (m ranges in $[5, 30]$ and L ranges in $[12, 40]$). Figure 2 summarizes the application of FOIL on this set of problems: learning failure or success is noted for each problem identified by the (m, L) pair. One important conclusion of their work is that the occurrence of a phase transition in the covering test is a general problem for all learning algorithms: the phase transition is viewed as an "attractor" for the heuristic search of any learning algorithm, which is bound to find a concept definition in the phase transition. Precisely, for all tested learners, there exists a failure region where the learnt theories are seemingly randomly constructed, with no better predictive accuracy than random guessing.

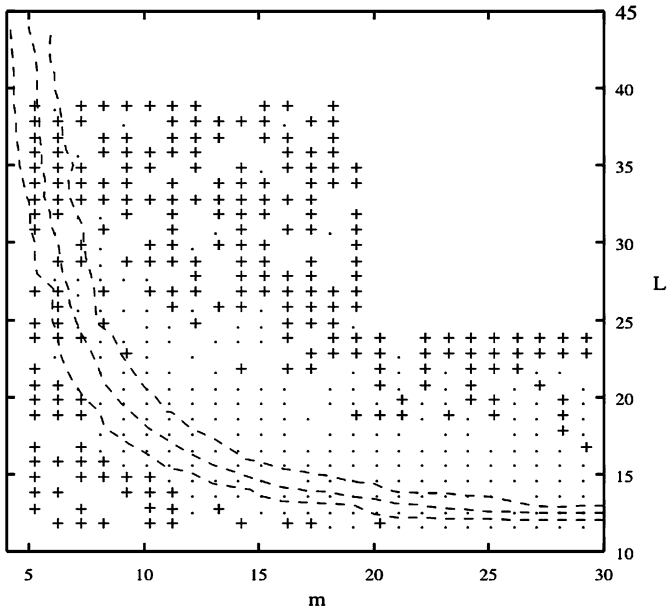

Fig. 2 From Botta et al. (2003): FOIL's competence map: success and failure regions. The phase transition region is indicated by the dashed curves, corresponding to the contour plots for $Psol = 0.9$, $Psol = 0.5$ and $Psol = 0.1$ respectively

We note however that only generate-and-test learning algorithms have been investigated in this work and that this conclusion has to be qualified in the case of data-driven learning algorithms. In the generate-and-test paradigm, refinements are only based on the structure of the hypothesis space, independently of the learning data. Therefore, for a given hypothesis, generate-and-test algorithms have to deal with many refinements that are not relevant with respect to the discrimination task. On the contrary, data-driven strategies allow to use the training data to prune irrelevant branches of the refinement graph before relying on the evaluation function and may overcome the problem of plateaus. Notably, building on the pioneering work of Winston on near-miss examples (Winston 1975), we show that, on the same set of problems as (Botta et al. 2003), a top-down data-driven strategy can cross any plateau and reach the target concept whenever near-misses are supplied in the training set, whereas these near-misses do not change the plateau profile and do not guide a generate-and-test strategy. We conclude that the location of the target concept with respect to the phase transition *alone* is not a reliable indication of the learning problem difficulty, as previously thought.

In the next section, we recall the two top-down learning strategies that we investigate in this paper, namely the top-down generate-and-test and the top-down data-driven strategies. For the sake of clarity, in Sect. 3, we illustrate their behavior on an attribute-value problem, namely an XOR problem, where a plateau is shown. We will see how the near-misses can be exploited by the top-down data-driven strategy to solve the problem without search. In Sect. 4, we draw a parallel between the phase transition of the covering test and the evaluation function of a learning algorithm. For that purpose, we re-use the set of learning problems proposed in (Giordana and Saitta 2000; Giordana et al. 2000;

Botta et al. 2003). Notably, we run additional experiments of FOIL on these problems with new settings where the hypothesis space is set so that the value of a given evaluation function can be directly read from the coverage probability of a hypothesis, as in Fig. 1. Although our conclusion on the ability of a generate-and-test approach in this setting differs from (Botta et al. 2003), these new experiments allow us to better compare the behavior of the generate-and-test and data-driven approaches when facing plateaus in the evaluation function. Next, in Sect. 5, we show how we can add near-misses to those problems in such a way that the plateaus are unchanged (and might even be wider) in order to guide a top-down data-driven algorithm to the target concept without search in the hypothesis space. This shows in particular that plateaus are pathological for generate-and-test algorithms but are not a complexity criterion for all algorithms. Finally, we will draw some conclusions about this work and present future research on the evaluation of learning complexity in a phase transition framework.

2 The top-down generate-and-test and data-driven strategies

In concept learning, we are given a learning set $E = E^+ \cup E^-$, with positive and negative examples of the unknown target concept, drawn from an example or instance language $\mathcal{L}_e$, a hypothesis language $\mathcal{L}_h$, a generality relation named covering test $\geq$ which relates $\mathcal{L}_e$ and $\mathcal{L}_h$ and partially order the hypotheses. After (Mitchell 1982), concept learning is defined as search in $\mathcal{L}_h$. The problem, known as the consistency problem, is to find a hypothesis $h \in \mathcal{L}_h$ such that h is consistent with the data:

$$\forall e^+ \in E^+, \quad h \geq e^+ \quad \text{(completeness)}, \qquad \forall e^- \in E^-, \quad h \not\geq e^- \quad \text{(correctness)}.$$

This seminal paper, relating (symbolic) concept learning to search in a state space, has enabled machine learning to integrate techniques from problem solving, operational research and combinatorics: greedy search in C4.5 (Quinlan 1986) and FOIL (Quinlan 1990), beam search in AQ (Michalski 1983) and CN2 (Clark and Niblett 1989), breadth-first search in (Shapiro 1983) and Aleph (Srinivasan 1999), 'A' search in PROGOL (Muggleton 1995), IDA (Iterative-Deepening A) search in MIO (Peña-Castillo and Wrobel 2002) to name a few systems. However, search in languages of interest, as those typically considered in ILP, is NP-hard (e.g. Haussler 1989; Cohen 1993; Kearns and Vazirani 1994) and heuristic search is crucial for efficiency (Pearl 1985). Heuristic search makes use of an evaluation function to prioritize nodes to consider at each stage of the search. A survey of evaluation functions used in machine learning can be found in (Fürnkranz and Flach 2005). It shows that all of them are based, without loss of generality, on three parameters that are the coverage rate of positive examples, the coverage rate of negative examples and a complexity measure of the hypothesis under consideration. The first two parameters are inherited from the learning task definition given above and stress the impact of the relevance of the covering test on the quality of search.

A central idea in concept learning is the use of a generality partial order between hypotheses to guide the resolution of the consistency problem (Mitchell 1982): if a hypothesis is incorrect (covering negative examples), it is said to be too general and has to be specialized. The natural stopping criterion[1] of a top-down strategy is the acquisition of a correct hypothesis. In the rest of the article, we will consider that learning of a

[1] In the article, only the noise-free learning setting is considered. In the case of noise, this stopping criterion is relaxed to accept hypotheses that are not consistent.

clausal theory is done with the covering strategy. Building on the works of (Plotkin 1970; Newell and Simon 1972; Winston 1975; Michalski 1983), Mitchell further refines the search strategy into the generate-and-test (TGT) and data-driven (TDD) strategies.

2.1 The top-down generate-and-test strategy

The large majority of top-down algorithms are based on the generate-and-test paradigm. In this paradigm, the refinement operator is only based on the structure of the hypothesis space, independently of the learning data (see e.g. Laird 1986).

Definition 1 (top-down generate-and-test refinement operator) Let $h \in \mathcal{L}_h$:

$$\rho(h) = \{h' \in \mathcal{L}_h \mid h \geq h'\}.$$

The advantage of this method, also called *weak method*, is that it is easy to apply it on a large variety of problems. However, for a given hypothesis, generate-and-test algorithms have to deal with many refinements that are not relevant with respect to the discrimination task. They only rely on the evaluation function to prune the irrelevant branches.

2.2 The top-down data-driven strategy

The top-down data-driven strategy has been advocated by Winston (1975) and developed in attribute-value learning in the AQ system (Michalski 1983) and among others AQR (Clark and Niblett 1989), INBF (Smith and Rosenbloom 1990), and recently in ILP (Alphonse and Rouveirol 2006). This strategy searches the space of hypotheses that are more general than or equal to a given positive example, named the seed example, and uses negative examples to prune irrelevant branches in the refinement graph. As opposed to a TGT approach, a TDD approach can therefore compensate for a poor evaluation function by using the learning data.

In attribute-value learning, the top-down data-driven refinement operator is described in (Michalski 1983) as a set of rules (*extension-against rules*) for computing refinements of boolean attributes, numerical attributes, nominal as well as hierarchical ones. The name of *extension-against* rules comes from the fact that the refinements are computed with respect to a chosen negative example: only refinements rejecting that particular example are generated. Formally, it is defined as a binary operator:

Definition 2 (top-down data-driven refinement operator) Let $h \in \mathcal{L}_h$, $e^- \in E^-$:

$$\rho(h, e^-) = \{h' \in \mathcal{L}_h \mid h \geq h' \quad \text{and } h' \not\geq e^-\}.$$

Note that the idea of reducing the hypothesis space by covering a seed example is used within systems like PROGOL, Aleph or MIO. However they do not benefit from the associated TDD operator and address the learning task within the generate-and-test paradigm. Relying on the negative examples allows a TDD strategy to have a branching factor that is smaller than the branching factor of a generate-and-test strategy searching in the same hypothesis space. Moreover, some TDD strategies make the most of the negative instances in order to select informative negative examples only (Winston 1975; Smith and Rosenbloom 1990; Alphonse and Rouveirol 2006), e.g. *near-misses* after Winston. A near-miss in Winston's terminology is a negative example that differs from the seed example by only one description (a literal or an attribute of the positive example that is

not in the negative example). This description necessarily belongs to the hypothesis that discriminates them. Ultimately, a TDD algorithm learning from a dataset provided with all near-misses with respect to the target concept would converge to the concept without search, generating only one refinement at each step. This is the result we are going to use in ILP to overcome the problem of plateaus exemplified in the phase transition framework (Giordana and Saitta 2000), showing that plateaus do not necessarily imply a high complexity of search in learning.

In the following, we describe the behavior of two greedy learners implementing the two above-mentioned strategies on an attribute-value learning example, the XOR problem. The greedy TGT strategy is implemented, for example, in attribute-value learning in C4.5 (Quinlan 1993) and Ripper (Cohen 1995) or in ILP in FOIL (Quinlan 1990) and Tilde (Blockeel and Raedt 1998). As for the TDD strategy, it is implemented in attribute-value learning in AQ (Michalski 1983) and in ILP in PROPAL(Alphonse and Rouveirol 2006). They use a beam search but are essentially equivalent to greedy search for non-trivial plateaus. The XOR problem is famous for exhibiting a plateau in top-down search. We are going to describe the resolution of the problem with the same methodology as the one used in the case of the phase transition of the covering test framework proposed by (Botta et al. 2003).

3 Plateau phenomena in the attribute-value case

3.1 Problem description

A k-n-XOR problem is a boolean learning problem where instances are drawn from $\mathcal{L}_e = \{0, 1\}^n$ and are tagged positive (1) or negative (0) according to the exclusive-or of the last k boolean attributes. The width of the plateau is $k - 1$, that is to say, for a given depth $d < k$, all hypotheses generated with d literals have the same evaluation. We give an example for $k = 2$ and $n = 3$ where the complete set of examples is given in Table 1.

3.2 Facing the plateau with a TGT strategy

A TGT learner starts from the most general hypothesis and generates all of its refinements: $\{a_1 = 0, a_1 = 1, a_2 = 0, a_2 = 1, a_3 = 0, a_3 = 1\}$. Each new hypothesis covers two positives examples and two negatives examples. Given that all evaluation functions use the coverage rate of the positive and negative examples (and that all hypotheses are of size 1), they cannot make a difference between all the possible refinements and have to rely on an arbitrary tie-breaking mechanism. For instance, we can consider that the first refinement is chosen and then it does not lead to a correct definition as only literals involving the two last attributes are relevant.

Table 1 Complete XOR examples with $k = 2$ and $n = 3$

Example	a_1	a_2	a_3	Concept	Example	a_1	a_2	a_3	Concept
e_1	0	0	0	0	e_5	1	0	0	0
e_2	0	0	1	1	e_6	1	0	1	1
e_3	0	1	0	1	e_7	1	1	0	1
e_4	0	1	1	0	e_8	1	1	1	0

3.3 Crossing the plateau using near-misses with a TDD strategy

We now run the TDD strategy, which makes use of the negative examples to prune branches of the refinement graph before relying on the evaluation function to guide the search. The TDD strategy sets the search to the generalization set of a seed example, which is the first uncovered positive example so far, namely e_2. The closest negative example to the seed is e_1. The two examples differ only by the value of one attribute and therefore e_1 is a Winston's near-miss: the application of the TDD refinement operator (Definition 2) produces only the refinement $\{a_3 = 1\}$, which is a literal of the target concept. There is no choice to make and the search does not have to rely on an evaluation function. As we have the whole set of instances, at the next refinement step, the search is conducted further without choice thanks to e_4, which is also a near-miss more specific than the hypothesis. The resulting hypothesis $a_3 = 1 \wedge a_2 = 0$ is correct and has been found without search.

Note that the use of a seed example in a generate-and-test approach like in PROGOL, ALEPH or MIO does not change the problem. While the number of generated hypotheses is reduced to three ($\{a_1 = 0, a_2 = 0, a_3 = 1\}$), the search still sees a plateau and goes blind.

We have illustrated the difference in behavior between the TGT and TDD strategies when search faces a plateau of the evaluation function: the efficiency of search can be hindered by plateaus but is not alone a reliable criterion of the search complexity. We next draw a parallel between this behavior and the one of the same strategies in ILP, when the phase transition of the covering test acts as a plateau for the evaluation function.

4 Impact of the phase transition of the covering test on the learning success

In this section, we investigate the impact of the phase transition of the covering test on the quality of the evaluation function and subsequently on the efficiency of the heuristic search. For that purpose, we re-use the set of learning problems[2] presented in (Botta et al. 2003) but in a different setting where there is a direct parallel between the plateaus of the "yes" and "no" regions in Fig. 1 and the plateau of the heuristic search. As noted in (Botta et al. 2003), the hypothesis space where FOIL is run is different from the hypothesis space where the phase transition is drawn. Precisely, the learning algorithms are allowed to add several literals based on the same predicate symbol and they evaluate hypotheses whose coverage rates are different from those pictured in the figure of the phase transition (Fig. 1). We conduct the same experiments but FOIL is bound to search in the same hypothesis space as the one where the phase transition is exhibited (Giordana and Saitta 2000). To run the experiments, we use the learning platform Aleph (Srinivasan 1999) as FOIL, as far as we know, cannot be set to run in such a hypothesis space when only one occurrence of a given predicate symbol is allowed. Aleph is a highly configurable platform that can emulate a large variety of learning algorithms including FOIL, as a pure greedy learner lead by the information gain.

For the sake of completeness, we first recall the ILP setting of Botta et al. for the hypothesis space and the generation of the learning problems that have been used to study the behavior of the learning algorithms. We will describe our experiments performed with FOIL on the same set of problems with the different hypothesis space. We then discuss the difference in behavior between the TGT learner FOIL and the TDD learner PROPAL(Alphonse and Rouveirol 2006) on the sets where near-misses are added.

[2]These problems are available at http://www.di.unito.it/~mluser/challenge/index.html.

4.1 Problem description

Botta et al. tackle the learning of a single Datalog clause from the hypothesis space $\mathcal{L}_h^m$ built as follow:

$$\mathcal{L}_h^m = \left\{ c \leftarrow \bigwedge_{k=1}^{n-1} p_{l_k}(X_k, X_{k+1}) \wedge \bigwedge_{k=n}^{m} p_{l_k}(X_{i_k}, X_{j_k}) \right\}$$

where c is the clause head without variables, $i_k < j_k \in \{1, \ldots, n\}, l_k \in \{1, \ldots, m\}$ and such that all literals in the clause body are built on distinct binary predicate symbols. We can see that the first $n-1$ literals ensure that all variables are linked and therefore that the set of variables cannot be decomposed into sets of independent variables that can break the covering test into easier sub-problems (Giordana and Saitta 2000). Examples are represented as ground Datalog clauses. Formally, let $A = \{a_1, \ldots, a_L\}$ be the set of all constants of cardinality L, and N the number of literals per predicate symbol. The example language $\mathcal{L}_e$ is the set of ground Datalog clauses defined as follow:

$$\mathcal{L}_e = \left\{ c \leftarrow \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{N} p_i(a_{ij_1}, a_{ij_2}) \right\}.$$

Each pair of constants (a_{ij_1}, a_{ij_2}) is drawn uniformly without replacement from the set $A \times A$.

A learning problem is parameterized with the tuple (m, L). The number of variables n is fixed to 4 and the number of literals per predicate symbol N is fixed to 100. In a (m, L) problem, m is the size of the target concept drawn from $\mathcal{L}_h^m$ and L the number of constants in $\mathcal{L}_e$. One can note that the target concept is one of the most specific clauses of the hypothesis space. For each problem, a learning set and a test set are built. Both are balanced sets of examples with 100 positive examples and 100 negative examples. In order to visit as uniformly as possible the "yes", "no" and "pt" regions, (m, L) problems have been uniformly selected without replacement for $m \in [5, 29]$ and $L \in [12, 39]$. It has to be noted that if (m, L) lies in the "yes" (resp. "no") region, by construction the concept description will almost surely cover (resp. reject) any randomly constructed example. For those problems, the example generator is modified and relies on a repair mechanism to ensure a balanced distribution of positive and negative examples (Botta et al. 2003).

4.2 Facing the plateau with a TGT learner

We ran FOIL on problems on $m = 5$ and $m = 10$ lines and on the upper-right corner problems ranging from $L \in [24, 39]$ on the $m = 18$ line and from $m \in [18, 29]$ on the $L = 24$ line. We also sampled some other problems without any difference in the results. On all of these problems, FOIL is unable to find any good approximation of the target concepts, being in the "yes", "no" or "pt" regions. Note that this result differs from (Botta et al. 2003) where FOIL, in a different hypothesis space, was shown to fail only in the "pt" region and in the "no" region for small numbers of constants L (see Fig. 2). To show why, we plot for a problem the coverage rate of the positive and the negative examples, as well as their standard deviation depending on the size of the hypothesis, averaged over 1000 randomly and uniformly drawn hypotheses as in (Giordana and Saitta 2000). A plateau is materialized by a standard deviation of the coverage rates of the examples close to 0.

Figure 3 shows the results for the $(5, 15)$ and $(10, 15)$ problems in the "yes" region, $(5, 35)$ and $(10, 20)$ in the "pt" region and $(8, 34)$ and $(10, 35)$ in the "no" region. For each

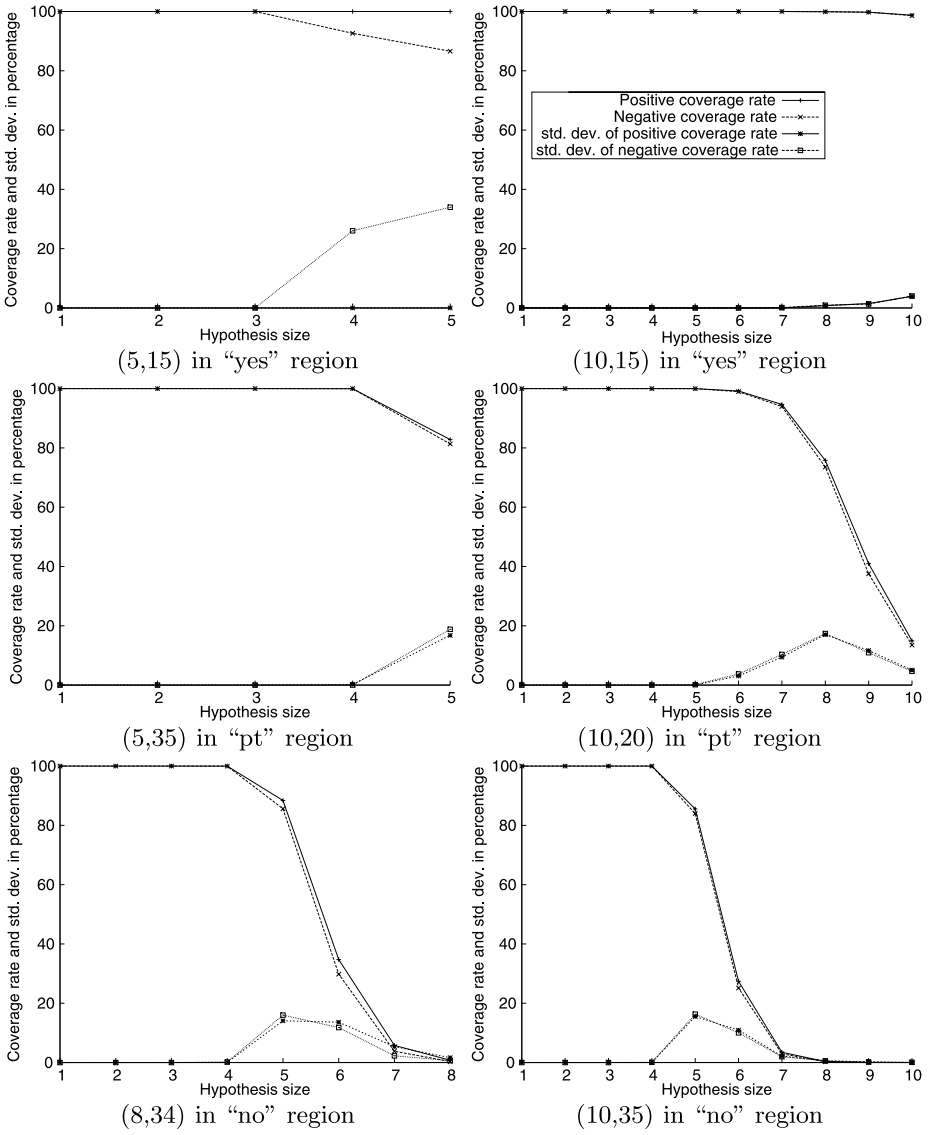

Fig. 3 Covering rates and plateau profiles for representative problems in the "yes", "pt" and "no" regions

problem, the value of a hypothesis, given an evaluation function, can be read on average from the corresponding sub-figure. As the top-down learner searches the hypothesis space, it evaluates hypotheses in turn of increasing size. For instance, on the (5, 15) problem, we can see that all positive examples are covered (100% of positive coverage rate) for any hypothesis size, and all negative examples up to a size of 3. Whatever the evaluation function is, the top-down learner will see hypotheses up to 3 literals long of equal value, or equivalently, it has to cross a plateau of width 3 before being able to use the evaluation function to discriminate between hypotheses.

Table 2 Accuracy measured on the test set, the number of clauses, as well as their minimum and maximum size, output by FOIL in the Aleph framework for some representative problems of the "no" region

(m, L)	$(8, 34)$	$(8, 39)$	$(14, 17)$	$(14, 28)$	$(14, 38)$	$(19, 17)$	$(29, 15)$	$(29, 24)$
Accuracy(%)	54.5	58	47.5	53	54.5	55.5	55.5	46
# of clauses	91	83	55	55	66	44	36	38
⟨min, max⟩	⟨7, 7⟩	⟨6, 7⟩	⟨7, 8⟩	⟨7, 9⟩	⟨6, 8⟩	⟨11, 14⟩	⟨15, 18⟩	⟨8, 9⟩

As we can see for the "yes" and the "pt" regions, hypotheses at the bottom of the search space are incorrect (i.e. for the maximum hypothesis size): on average, there does not exist a correct theory of the learning problems in these regions. The target concept is hidden among the bottom hypotheses and a TGT algorithm can only find it by chance as we will detail later. The natural stopping criterion of the TGT strategy is only met for problems generated in the "no" region and a TGT learner is bound to output hypotheses belonging to the intermediate region inbetween the "pt" and the "no" regions. This is similar to the findings of (Botta et al. 2003). Therefore, we ran FOIL on several problems in the "no" region, whose results of typical cases are summarized in Table 2.

This table gives for each problem the accuracy evaluated on the corresponding test set, the number of clauses in the theory and the minimum and maximum size of the clauses in the theory. Again FOIL is unable to produce any reasonable approximations of the target concept and performs seemingly as a random classifier. These results are not surprising as we can see that the "yes" region of the phase transition represents plateaus for the heuristic search that must be crossed without an informative evaluation function to guide the search toward the good path to the target concept.

The impact of plateaus is well-known on classical heuristic search and they are a pathological case of heuristic search, whether complete or not, as the search goes blind (see e.g. Pearl 1985; Korf 1985b; Russell and Norvig 1995). For instance, in the case of a greedy search such as used in FOIL, the algorithm has to make a random choice between possible refinements and therefore acts as a random walker. In one of the smaller problem $(5, 15)$, the plateau is already as wide as 3 literals. FOIL's search can be seen as starting the top-down search from a randomly and uniformly drawn partial hypothesis of size 3. It appears on all learning problems that a plateau of such width, which lengthens as we take problems with smaller number of constants L, is enough to fool the search. A parallel can be made with the XOR problem described in Sect. 1 where a plateau of width 1 was enough to miss the target concept. Several approaches have been proposed to overcome the limitation of a greedy search in ILP like look-ahead (Blockeel and Raedt 1997), randomized look-ahead (Serra et al. 2001) or macro-operators (Alphonse 2004) but we do not discuss these techniques here as their effectiveness strongly depends on the width of the plateau and we rather focus on complete heuristic searches. This is also a pathology for complete heuristic searches, like the best-first or the best-first iterative-deepening searches (Korf 1985a) implemented in PROGOL and MIO respectively, although they can theoretically find the target concept in any of the considered learning problems (see below). Best-first search (and its relatives A or A*) cannot be guided by the evaluation function and it therefore degenerates into a breadth-first search where the space requirement grows exponentially. It is often said that A-like search runs out of memory before reaching the time limit (Korf 1985b). Best-first iterative-deepening (and its relatives IDA or IDA*), used in MIO, avoid the exponential space complexity but will reach the time limit.

If general-purpose complete learning algorithms cannot solve such problems with large plateaus within the time limit, it has to be noted that a complete enumeration algorithm has

been proposed to solve the problems in the "no" region (Ales-Bianchetti et al. 2002). The authors propose to use dedicated pruning techniques to prune large parts of the hypothesis space. The algorithm generalizes a set of positive examples in a restricted hypothesis space that contains the target concept. If the restricted hypothesis space is in the "no" region, it is shown that the number of common generalizations of a set of positive examples decreases exponentially in the number of positive examples and therefore can be searched for a consistent hypothesis. However this pruning technique, making assumptions about the target concept, is tailored to the problem at hand and is not a general learning technique.

The state of the art on evaluation functions used in learning shows that they are all, without loss of generality, based on two main parameters that are the rate of positive and negative examples covered. As these two parameters are inherited from the definition of the learning task, it is unlikely that a solution to overcome these plateau phenomena consists in designing new evaluation functions. The generate-and-test learning paradigm seems doomed to fail to identify a good approximation of the target concept when facing non-trivial plateaus.

5 Crossing plateaus using near-misses with the TDD strategy

In this section, we show that near-misses can be added to a learning problem without changing the plateau profile.[3] In doing so, a TGT learner cannot take advantage of the addition of the most informative negative examples whereas a TDD learner is guided to the target concept without search and then overcomes the plateau problems. For the sake of readability, we quickly present the TDD strategy in ILP, limited to our setting, and we refer to (Alphonse and Rouveirol 2006) for a detailed presentation.

5.1 The TDD strategy in ILP

The TDD strategy is biased toward the covering of a seed example, s, and then amounts to mapping the initial search space of the learning algorithm into the space of generalizations of the seed example. Additionally, it is required that this space of generalizations be isomorphic to a boolean lattice and therefore the TDD refinement operator (see Definition 2) is an algebraic resolution of the learning problem. This requirement is naturally met in ILP under the "object identity" partial order, but also under other partial orders as θ-subsumption as classically handled in systems like PROGOL or Aleph. Indeed, as it is known that no (ideal) refinement operators exist for this partial order (van der Laag and Nienhuys-Cheng 1994), the hypothesis space is often restricted to the power set of a seed example, which is isomorphic to a boolean lattice. Looking for a hypothesis of $\mathcal{L}_h^m$ that both covers s and rejects a negative example e^- can be equivalently performed by looking for a generalization of s that rejects $lgg(s, e^-)$, the least general generalization of s and e^-. By definition of the lgg (Plotkin 1970), we have $h \geq s \wedge h \geq e^- \Leftrightarrow h \geq lgg(s, e^-)$. Equivalently, by transposition, we have $h \not\geq s \vee h \not\geq e^- \Leftrightarrow h \not\geq lgg(s, e^-)$. As the TDD strategy is biased toward the covering of s, $h \not\geq e^- \Leftrightarrow h \not\geq lgg(s, e^-)$. Note that in ILP the lggs are not unique and several lggs need to be rejected before rejecting the initial negative example.

By reformulating the whole negative example set into a boolean search space by means of lggs, we can show that only the most specific set of lggs, which are incomparable by definition, are useful for learning. By making use of this partial order between negative examples,

[3]The scripts used in this paper to add the near-misses examples are available on demand.

the TDD strategy can reject the most informative negative examples only and following that way efficiently prune the refinement graph. Precisely, if a negative example is such that one of its lggs with the seed misses only one literal of the seed, then this literal necessarily belongs to the hypothesis that discriminate them. This particular example, or its particular lgg, is a Winston's near-miss. The TDD strategy implemented in PROPALmakes the most of this property by rejecting the closest examples first, named in this context the nearest-misses examples. A Winston's near-miss is a nearest-miss by definition, being the closest.

5.2 Generating near-misses in the artificially generated problems

To guide the search with near-misses, we have to create new negative examples which differ from the seed example only by one literal in the reformulated space by means of lggs. We will have as many near-misses as literals in the target concept to guide the refinement process literal by literal.[4] Before presenting the generation procedure, we have to set the seed example, or equivalently the bottom clause of the search space. For efficiency reasons, and without loss of generality, we do not choose a positive example as seed. The reason is that the examples provided in the set of problems can be very large with up to 3000 literals. Instead, we choose as a bottom clause the union of the most specific clauses in the hypothesis space, taking into account the fact that they are all built from the same set of variables. Then, for a target concept of size m with n variables, the bottom clause has $m \times \frac{n(n-1)}{2}$ literals. It is necessarily more specific than any hypothesis and it is covered by the target concept.

Algorithm 1 *GenerateProblemWithNearMisses*(h, m, E)

begin

1 $NM = \emptyset$ *% the set of near-misses*
2 Create the bottom clause B of $\mathcal{L}_h^m$
3 For $k = 1$ to m do *% one near-miss per concept's literal*
4 Extract a negative example e^- from E *%* $E = E \setminus e^-$
5 Add to e^-'s definition a skolemised version of the bottom clause
6 While $\exists\theta$ such that $h\theta \subseteq e^-$ do
7 Remove $p_k(X_i, X_j)\theta$ from e^- *% the kth literal of h*
8 $NM = NM \cup e^-$
9 Return $E \cup NM$

End

The generation procedure is given in Algorithm 1. It takes as input a target concept h of size m, the corresponding learning problem E and outputs a new learning problem. This new learning problem has the same number of positive and negative examples as the input problem. At line 5, we see that a negative example e^- is augmented with the addition of a skolemized version of the bottom clause. Hence, it is now covered by all the hypotheses in $\mathcal{L}_h^m$and, supposing that a solution of the learning problem exists in $\mathcal{L}_h^m$, is now turned into a positive example: for each $h \in \mathcal{L}_h^m$, we can exhibit a substitution θ, involving only the skolem constants of the added bottom clause, such that $h\theta \subseteq e^-$. It has to be modified into a negative example which is done at lines 6 and 7. It guarantees that the new negative example is a near-miss: the longest lgg between the bottom clause and the example will differ only

[4] One can note in the following that only one negative example can represent all near-misses. However, to make a parallel with the near-misses in attribute-value learning, we consider here that one near-miss is associated with one negative example.

in the relevant literal of the target concept. When this near-miss example will be used by the TDD refinement operator to specialize the incorrect hypothesis, only the relevant literal will be produced as refinement. For the sake of completeness, note that in the case where there are several possible matchings of the target concept onto the bottom clause the near-miss will have as many missing literals. It means that there exist several syntactic variants of the target concept in the power set of the bottom clause and each unmatched or missing literal belongs to a different syntactic version. A greedy search can still be applied successfully as any literal is a valid refinement toward a definition of the target concept. The initial definition of near-miss by Winston does not take into account this particularity of relational learning but we leave this consideration out, as it is beyond the scope of the paper. Practically, we do not observe that in the learning problems we consider here: there is always a single possible matching substitution between the target concept and the bottom clause.

The most important point of this problem generation is to note that the number of models of a hypothesis in the near-misses can only be greater than in the original negative examples: if a hypothesis covers the original negative example, it covers the corresponding near-miss and the converse is not true. This property is necessary to show that a near-miss can be added without changing the plateau profile. At lines 6 and 7, during the computation of substitutions between the target concept and the positive example, only substitutions of the variables onto the skolem constants of the bottom clause part of the example can be found as the original example is a negative example. This property is due to the fact that a phase transition is shown in a space of hypotheses where the resulting matching problems cannot be decomposed into simpler sub-matching problems as explained in Sect. 4.1 (see Giordana and Saitta 2000). Therefore, the computed substitutions can only involve the skolem constants and not a mix of some constants from the original example and some skolem constants. In the next section, we show the resulting plateau profiles and discuss the behavior of the TGT and the TDT learners on these new problems.

5.3 Experiments

We ran FOIL on the same problems as before, augmented with near-misses as described above. On all of these problems, FOIL is unable to find any good approximation of the target concepts, being in the "yes", "no" or "pt" regions. As we can see in Fig. 4, the plateau profile does not change and a TGT learner cannot take advantage of the addition of the most informative negative examples. The problems of the "no" region exhibit a different behavior of FOIL which cannot even output a correct theory for any of them. If we look at the rate of covered negative examples for the maximum hypothesis size, we see that the most specific hypotheses are not correct on average. These hypotheses cover the near-misses which by construction are more specific than all hypotheses but merely the ones that are on the path to the target concept. This is an illustration of the performance of FOIL as bad as random guessing on the original problems in the "no" region described in Sect. 4.2. As noted in (Ales-Bianchetti et al. 2002), these types of problems are equivalent to finding the needle in the haystack: almost only the target concept is the solution of the consistency problem.

The opposite behavior is exhibited by the TDD learner PROPALwhich, by construction, solves all of the problems being in the "yes", "pt" or "no" regions. The learner makes the most of the learning data by exploiting only the information provided by the near-misses to guide its search. Note that as the branching factor is reduced to one thanks to the near-misses, the target concept is exactly identified each time as opposed to evaluating the quality of the approximation on a test set as for FOIL. Interestingly, the resolution of the problems is quite fast, taking below 20 minutes on a desktop computer up to problems on the line $m = 14$ to several hours for the hardest ones.

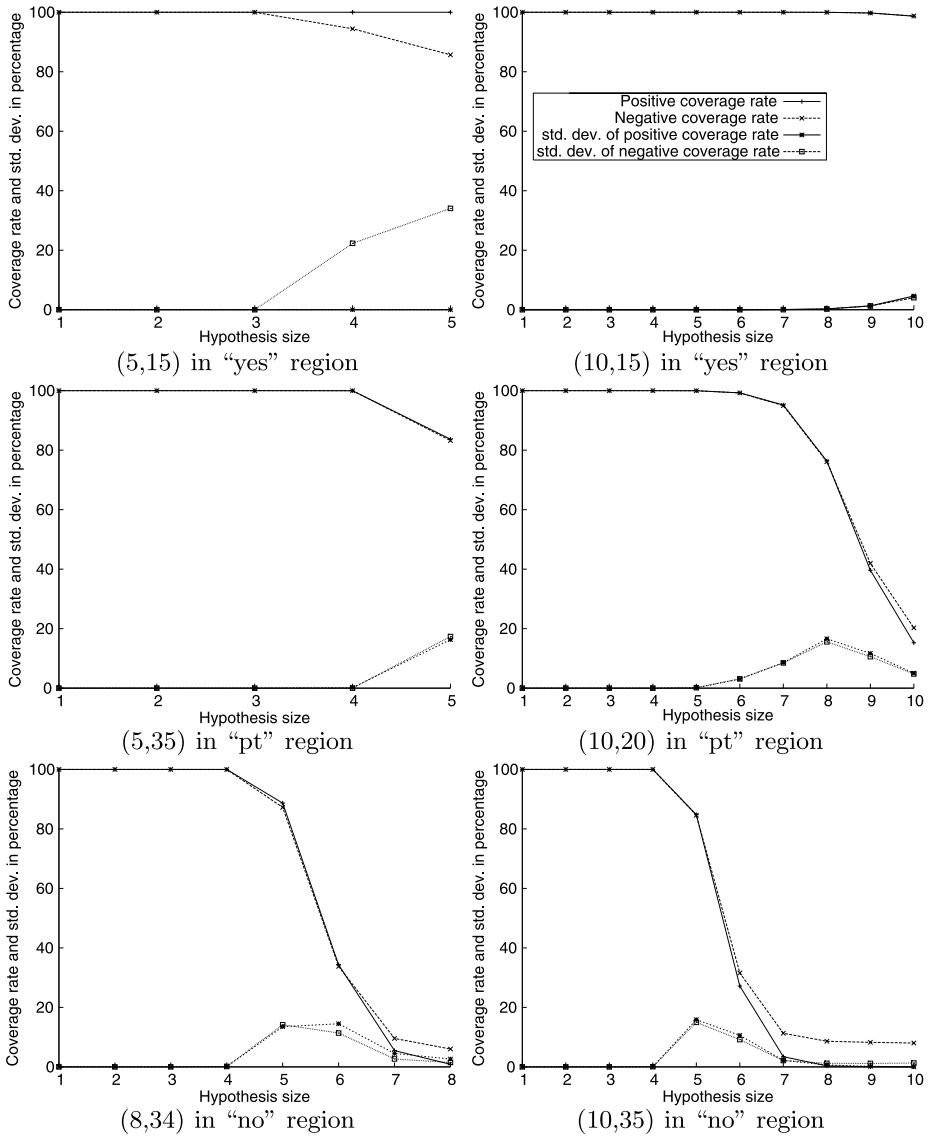

Fig. 4 Coverage rates and plateau profiles for representative learning problems in the "yes", "pt" and "no" regions with-near misses

6 Conclusions

Plateau phenomena in ILP have been studied recently in the phase transition framework and an important work has been done on identifying the criteria of success of learning algorithms (Botta et al. 2003). The conclusion drawn from this work was that the location of the target concept with respect to the phase transition of the covering test was conclusive of the difficulty of the learning problems. A failure region was identified for all the tested learners, starting from the "pt" region to the beginning of the "no" region. We performed

additional experiments that strengthen this result. When the top-down search is conducted in the hypothesis space that exhibits a phase transition in its covering test, the "yes" region acts as a plateau for the heuristic search. This is the pathological case of heuristic search, whether complete or not, as the plateau must be crossed without being able to differentiate between refinements. In such a case, the greedy TGT learner, FOIL, cannot solve any of the problems. We showed, however, that this criterion alone is not reliable. As a main result, we showed that a top-down data-driven learning algorithm (Alphonse and Rouveirol 2006), supplied with near-miss examples was able to solve all problems, although the near-miss examples are still non-informative for generate-and-test algorithms. The plateau phenomena exhibited in the phase transition framework is a pathological case of the generate-and-test learners as they only rely on an evaluation function to guide their search, but it is not, alone, a reliable complexity measure for all learners.

In future work, we are going to further investigate the complexity of ILP within the phase transition framework by studying the impact of the location of a learning problem with respect to the phase transition of the search problem. The learning problems considered in this paper are instances of what is known as the bounded ILP-consistency problem in (Gottlob et al. 1997). That is finding a (single) horn clause, with a size bounded by an integer k polynomial in the problem size, that is consistent with the examples. It is known that this problem is NP^{NP} (or equivalently Σ_2^P-complete): the search is NP-complete and it is guided by the covering test which is NP-complete. Therefore, ILP potentially has to face two phase transitions: the phase transition of the covering test and the phase transition of search. We believe that the study of the latter is a better complexity measure relevant to both strategies as it translates into the inherent complexity of search.

Acknowledgements We are very grateful to Henry Soldano and Céline Rouveirol for discussing the impact of the TDD strategy on heuristic search. We would also like to thank Lorenza Saitta and Attilio Giordana for discussing their work with us.

References

Ales-Bianchetti, J., Rouveirol, C., & Sebag, M. (2002). Constraint-based learning of long relational concepts. In M. Kaufmann (Ed.), *Nineteenth international conference on machine learning (ICML-2002)* (pp. 35–42), Sydney, NSW, Australia. Los Altos: Kaufmann.

Alphonse, E. (2004). Macro-operators revisited in inductive logic programming. In *Proceedings of the conference on inductive logic programming* (pp. 8–25), Porto, Portugal. Berlin: Springer.

Alphonse, E., & Rouveirol, C. (2006). Extension of the top-down data-driven strategy to ILP. In *Proceedings of the conference on inductive logic programming*, Santiago de Compostela, Spain. Berlin: Springer.

Blockeel, H., & Raedt, L. D. (1997). Lookahead and discretization in ILP. In N. Lavrač & S. Džeroski (Eds.), *Proceedings of the conference on inductive logic programming* (Vol. 1297, pp. 77–84), 17–20 September 1997. Berlin: Springer.

Blockeel, H., & Raedt, L. D. (1998). Top-down induction of first order decision trees. *Artificial Intelligence*, *101*, 285–297.

Botta, M., Giordana, A., Saitta, L., & Sebag, M. (2003). Relational learning as search in a critical region. *Journal of Machine Learning Research*, *4*, 431–463.

Cheeseman, P., Kanefsky, B., & Taylor, W. (1991). Where the really hard problems are. In R. Myopoulos (Ed.), *Proceedings of the 12th international joint conference on artificial intelligence* (pp. 331–340), Sydney, Australia, August 1991. Los Altos: Kaufmann.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–283.

Cohen, W. W. (1993). Learnability of restricted logic programs. In S. Muggleton (Ed.), *Proceedings of the conference on inductive logic programming* (pp. 41–72). Szeged: J. Stefan Institute.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the 12th international conference on machine learning* (pp. 115–123), Tahoe City, CA. Los Altos: Kaufmann.

Fürnkranz, J., & Flach, P. (2005). Roc'n' rule learning-towards a better understanding of covering algorithms. *Machine Learning*, *58*, 39–77.

Giordana, A., & Saitta, L. (2000). Phase transitions in learning relations. *Machine Learning*, *41*, 217–25.
Giordana, A., Saitta, L., Sebag, M., & Botta, M. (2000). Analyzing relational learning in the phase transition framework. In *17th international conference on machine learning* (pp. 311–318), Stanford, CA, USA. Los Altos: Kaufmann.
Gottlob, G., Leone, N., & Scarcello, F. (1997). On the complexity of some inductive logic programming problems. In N. Lavrač & S. Džeroski (Eds.), *Proceedings of the 7th international workshop on inductive logic programming* (Vol. 1297, pp. 17–32). Berlin: Springer.
Haussler, D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning*, *4*(1), 7–40.
Hayes-Roth, F., & McDermott, J. (1977). Knowledge acquisition from structural descriptions. In R. Reddy (Ed.), *Proceedings of the 5th international joint conference on artificial intelligence* (pp. 356–362). Cambridge: Kaufmann.
Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory*. Cambridge: MIT Press.
Korf, R. E. (1985a). Depth-first iterative-deepening: an optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109.
Korf, R. E. (1985b). Macro-operators: a weak method for learning. *Artificial Intelligence*, *26*, 35–77.
Laird, P. D. (1986). Inductive inference by refinement. In T. Kehler & S. Rosenschein (Eds.), *Proceedings of the 5th national conference on artificial intelligence* (Vol. 1, pp. 472–476), August 1986. Los Altos: Kaufmann.
Michalski, R. S. (1983). *A theory and methodology of inductive learning*. Palo Alto: Kaufmann.
Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203–226.
Muggleton, S. (1995). Inverse entailment and PROGOL. *New Generation Computing*, *13*, 245–286.
Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs: Prentice-Hall.
Peña-Castillo, L., & Wrobel, S. (2002). On the stability of example-driven learning systems: a case study in multirelational learning. In *MICAI 2002: advances in artificial intelligence* (pp. 321–330). Berlin: Springer.
Pearl, J. (1985). *Heuristics*. Reading: Addison-Wesley.
Plotkin, G. (1970). A note on inductive generalization. In *Machine intelligence* (pp. 153–163). Edinburgh: Edinburgh University Press.
Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.
Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.
Quinlan, J. R. (1991). Determining literals in inductive logic programming. In *Proceedings of the 12th international joint conference on artificial intelligence* (pp. 746–750), Sydney, New South Wales, Australia. Berlin: Springer.
Quinlan, J. R. (1993). *C4.5: programs for machine learning*. San Mateo: Kaufmann.
Richards, B., & Mooney, R. (1992). Learning relations by pathfinding. In *Proceedings of the tenth national conference on artificial intelligence* (pp. 723–738). San Jose: AAAI Press/MIT Press.
Russell, S., & Norvig, P. (1995). *Artificial intelligence: a modern approach*. Englewood Cliffs: Prentice Hall.
Serra, A., Giordana, A., & Saitta, L. (2001). Learning on the phase transition edge. In B. Nebel (Ed.), *Proceedings. of the 7th int. conference. on artificial intelligence (IJCAI-01)* (pp. 921–926), Seattle, Washington, USA. Los Altos: Kaufmann.
Shapiro, E. (1983). *Algorithmic program debugging*. Cambridge: MIT Press.
Silverstein, G., & Pazzani, M. J. (1991). Relational cliches: constraining constructive induction during relational learning. In L. Birnbaum & G. Collins (Eds.), *Proceedings of the 8th international workshop on machine learning* (pp. 203–207), University of California, Irvine. Los Altos: Kaufmann.
Smith, B. D., & Rosenbloom, P. S. (1990). Incremental non-backtracking focusing: a polynomially bounded generalization algorithm for version spaces. In *Proceedings of the 8th national conference on artificial intelligence* (pp. 848–853). Boston: AAAI Press/MIT Press.
Srinivasan, A. (1999). A learning engine for proposing hypotheses (Aleph). http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph.
van der Laag, P., & Nienhuys-Cheng, S. H. (1994). A note on ideal refinement operators in ILP. In S. Wrobel (Ed.), *Proceedings of the 4th international workshop on inductive logic programming* (Vol. 237, pp. 247–262). Bad Honnef/Bonn: Gesellschaft für Mathematik und Datenverarbeitung MBH.
Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision* (pp. 157–209). New York: McGraw-Hill.