Appears in the
Proceedings of the Ninth Canadian Conference on Artificial Intelligence (CSCSI-92),
Vancouver, May 1992.

[Awarded "Best Paper Prize"]

Probabilistic Hill-Climbing: Theory and Applications

Russell Greiner*
Siemens Corporate Research, Princeton, NJ 08540
greiner@learning.siemens.com (609) 734-3627

Abstract

Many learning systems search through a space of possible performance elements, seeking an element with high expected utility. As the task of finding the globally optimal element is usually intractable, many practical learning systems use hill-climbing to find a local optimum. Unfortunately, even this is difficult, as it depends on the distribution of problems, which is typically unknown. This paper addresses the task of approximating this hill-climbing search when the utility function can only be estimated by sampling. We present an algorithm that returns an element that is, with provably high probability, essentially a local optimum. We then demonstrate the generality of this algorithm by sketching three meaningful applications, that respectively find an element whose efficiency, accuracy or completeness is nearly optimal. These results suggest approaches to solving the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

1 Introduction

Many learning tasks can be viewed as a search through a space of possible performance elements seeking an element that is optimal, based on some utility measure. As examples, many inductive systems seek a function whose classification is optimal, *i.e.*, which labels correctly as many examples as possible; and many explanation-based learning [DeJ88, MCK+89] and chunking [LNR87] systems seek a problem solving system that is optimally efficient [Min88, Gre91]. In each of these cases, the utility function used to compare the different elements is defined as the expected value of a particular scoring function, averaged over the distribution of samples (or goals, queries, problems, ...) that will be seen [Hau88, OG90, GO91].

There are two problems with implementing such a learning system: First, we need to know the distribution of samples to determine which element is optimal; unfortunately, this information is usually unknown. There are, of course, standard statistical techniques that use a set of observed samples to estimate the needed information; and several classes of learning systems have incorporated these techniques. For example, many "PAC-learning" systems [Val84] use these estimates to identify an element that is approximately a global optimum.

This leads to the second problem: unfortunately, the task of identifying the globally optimal element, even given the correct distribution information, is intractable for many spaces of elements [Gre91, Hau88]. A common response is to build a system that hill-climbs towards a *local* optimum. Many well-known inductive learning systems, including BACKPROP [Hin89] and ID3 [Qui86], use this approach, as do many speedup learning methods; see especially [GD91]. Unfortunately, few existing systems guarantee that each hill-climbing step is even an improvement, meaning the final element is not always even superior to the initial one, much less an optimum in the space of elements. Moreover, fewer systems include a stopping criterion to determine when the learning has reached a point of diminishing returns.

The work presented here draws ideas from both of these themes: In particular, it describes a general learning algorithm, PALO, that hill-climbs to a local optimum, using a utility metric that is estimated by sampling. Given any parameters $\epsilon, \delta > 0$, PALO efficiently produces an element whose expected utility is, with probability greater than $1 - \delta$, an ϵ-local optimal.[1] Moreover, PALO can work unobtrusively [MMS85], passively gathering the statistics it needs by simply watching a performance element solve problems relevant to a user's applications. Here, the incremental cost of PALO's hill-climbing, over the cost of simply solving performance problems, can be very minor.

*Most of this work was performed at the University of Toronto, where it was supported by the Institute for Robotics and Intelligent Systems and by an operating grant from the National Science and Engineering Research Council of Canada. I also gratefully acknowledge receiving many helpful comments from William Cohen, Dale Schuurmans and the anonymous referees.

[1] Theorem 1 below defines both our sense of efficiency, and "ϵ-local optimality".

Section 2 motivates the use of "expected utility" as a quality metric for comparing performance elements. Section 3 then describes a statistical tool for evaluating whether the result of a proposed modification is better (with respect to this metric) than the original performance element PE; this tool can be viewed as a mathematically rigorous version of [Min88]'s "utility analysis". We use this tool to define the general PALO algorithm, that incrementally produces a series of performance elements $\mathrm{PE}_1, \ldots, \mathrm{PE}_m$ such that each PE_{i+1} is statistically likely to be an incremental improvement over PE_i and, with high confidence, the performance of the final PE_m is a local optimal in the space searched by the learner. Section 4 demonstrates the generality of this approach by presenting three different instantiations of the PALO system, each using its own set of transformations to find a near-optimal element within various sets of performance elements, where optimality is defined in terms of efficiency, accuracy, or completeness, respectively.

2 Framework

We assume as given a (possibly infinite) set of performance elements $\mathcal{PE} = \{\mathrm{PE}_i\}$, where each $\mathrm{PE} \in \mathcal{PE}$ is a system that returns an answer to each given problem (or query or goal, etc.) $q_i \in \mathcal{Q}$, where $\mathcal{Q} = \{q_1, q_2, \ldots\}$ is the set of all possible queries. We also use the utility function $c : \mathcal{PE} \times \mathcal{Q} \mapsto \Re$, where $c(\mathrm{PE}, q)$ measures how well the element PE does at solving the problem q. (Section 4 defines $c_s(\mathrm{PE}, q)$, which quantifies the time PE requires to solve q; $c_a(\mathrm{PE}, q)$, the accuracy of PE's answer; and $c_c(\mathrm{PE}, q)$, PE's categoricity.)

This utility function specifies which PE_i is best for a single problem. Our performance elements, however, will have to solve an entire ensemble of problems. As we obviously prefer the element that is best overall, we must therefore consider the distribution of problems that our performance elements will encounter. We model this using a probability function, $Pr : \mathcal{Q} \mapsto [0, 1]$, where $Pr[\,q_i\,]$ denotes the probability that the problem q_i is selected.[2] We then define the *expected utility* of a performance element:

$$\mathrm{C}[\,\mathrm{PE}\,] \stackrel{def}{=} E[\,c(\mathrm{PE}, \mathbf{q})\,] = \sum_{q \in \mathcal{Q}} Pr[\,q\,] \times c(\mathrm{PE}, q) \quad (1)$$

Our underlying challenge is to find the performance element whose expected utility is maximal. As mentioned above, there are two problems: First, the problem distribution, needed to determine which element is optimal, is usually unknown. Second, even if we knew that distribution information, the task of identifying the optimal element is often intractable.

3 The PALO Algorithm

This section presents a learning system, PALO (for "<u>P</u>robably <u>A</u>pproximately <u>L</u>ocally <u>O</u>ptimal") that sidesteps the above problems by using a set of sample queries to estimate the distribution, and by hill-climbing efficiently from a given initial PE_0 to one that is, with high probability, essentially a local optimum. This section first states the fundamental theorem that specifies PALO's functionality, then summarizes PALO's code and sketches a proof of the theorem.

In more detail, PALO takes as arguments an initial PE_0 and parameters $\epsilon, \delta > 0$. It uses a set of sample problems drawn at random from the $Pr[\,\cdot\,]$ distribution to climb from the initial PE_0 to a final PE_m, using a particular set of possible transformations $\mathcal{T} = \{\tau_j\}$, where each τ_j maps one performance element to another; see Section 4. PALO then returns this final PE_m. Theorem 1 states our main theoretical results.[3]

Theorem 1 *The* PALO$(\mathrm{PE}_0, \epsilon, \delta)$ *process incrementally produces a series of performance elements* $\mathrm{PE}_0, \mathrm{PE}_1, \ldots, \mathrm{PE}_m$, *staying at a particular* PE_j *for only a polynomial number of samples before either climbing to* PE_{j+1} *or terminating. With probability at least* $1 - \delta$, PALO *will terminate. It then returns an element* PE_m *whose expected utility* $C[\,\mathrm{PE}_m\,]$ *is, with probability at least* $1 - \delta$, *both*

1. *at least as good as the original* PE_0; *i.e.,* $C[\,\mathrm{PE}_m\,] \geq C[\,\mathrm{PE}_0\,]$; *and*
2. *an* ϵ*-local optimum — i.e.,* $\forall \tau_j \in \mathcal{T}.\ C[\,\mathrm{PE}_m\,] \geq C[\,\tau_j(\mathrm{PE}_m)\,] - \epsilon$ $\square$.

Algorithm PALO(PE_0, ϵ, δ)

- $i \leftarrow 0 \qquad j \leftarrow 0$

L1: Let $S \leftarrow \{\}$ $\quad$ Neigh $\leftarrow \{\, \tau_k(\mathrm{PE}_j) \,\}_k$

$\Lambda_{max} = \max\{\, \Lambda[\mathrm{PE}', \mathrm{PE}_j] \mid \mathrm{PE}' \in \text{Neigh} \,\}$

L2: Get query q (from the user).

Let $S \leftarrow S \cup \{q\} \qquad i \leftarrow i + |\text{Neigh}|$

- If there is some $\mathrm{PE}' \in$ Neigh such that
$$\Delta[\mathrm{PE}', \mathrm{PE}_j, S] \geq \Lambda[\mathrm{PE}', \mathrm{PE}_j]\sqrt{\tfrac{|S|}{2} \ln\left(\tfrac{i^2 \pi^2}{3\,\delta}\right)} \quad (2)$$
then let $\mathrm{PE}_{j+1} \leftarrow \mathrm{PE}'$, $\quad j \leftarrow j+1$.
Return to **L1**.
- If $|S| \geq \frac{2\Lambda_{max}^2}{\epsilon^2} \ln\left(\frac{i^2\pi^2}{3\,\delta}\right)$ and $\forall\, \mathrm{PE}' \in$ Neigh. $\Delta[\mathrm{PE}', \mathrm{PE}_j, S] \leq \frac{\epsilon\,|S|}{2}$, $\quad (3)$
then halt and return as output PE_j.
- Otherwise, return to **L2**.

Figure 1: Code for PALO

The basic code for PALO appears in Figure 1. In essence, PALO will climb from PE_j to a new PE_{j+1} if PE_{j+1} is likely to be better than PE_j; *i.e.*, if we are highly confident that $\mathrm{C}[\,\mathrm{PE}_{j+1}\,] > \mathrm{C}[\,\mathrm{PE}_j\,]$. To determine this, define

$$d_i = \Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, q_i] \stackrel{def}{=} c(\mathrm{PE}_\alpha, q_i) - c(\mathrm{PE}_\beta, q_i)$$

[2] We assume that $|\mathcal{Q}|$ is finite for purely pedagogical reasons, as it allows us to define this probability function. There are obvious ways of extending this analysis to handle an infinite set of problems.

[3] This proof, and others, appear in the expanded version of this paper [Gre92].

to be the difference in cost between using PE_α to deal with the problem q_i, and using PE_β. As each query q_i is selected randomly according to some fixed distribution, these d_is are independent, identically distributed random variables whose common mean is $\mu = \mathrm{C}[\mathrm{PE}_\alpha] - \mathrm{C}[\mathrm{PE}_\beta]$. (Notice PE_α is better than PE_β if $\mu > 0$.)

Let $Y_n \stackrel{def}{=} \frac{1}{n}\Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, \{q_i\}_{i=1}^n]$ be the sample mean over n samples, where $\Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, S] \stackrel{def}{=} \sum_{q\in S} c(\mathrm{PE}_\alpha, q) - c(\mathrm{PE}_\beta, q)$ for any set of queries S. This average tends to the true population mean μ as $n \to \infty$; *i.e.*, $\mu = \lim_{n\rightsquigarrow\infty} Y_n$. Chernoff bounds [Che52] describe the probable rate of convergence: the probability that "Y_n is more than $\mu + \gamma$" goes to 0 exponentially fast as n increases; and, for a fixed n, exponentially as γ increases. Formally,[4]

$$\begin{aligned} Pr[Y_n > \mu + \gamma] &\leq e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2} \\ Pr[Y_n < \mu - \gamma] &\leq e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2} \end{aligned}$$

where Λ is the range of possible values of $c(\mathrm{PE}_\alpha, q_i) - c(\mathrm{PE}_\beta, q_i)$. This $\Lambda = \Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta]$ is also used in both the specification of Λ_{max} under Line **L1** and in Equation 2.

The PALO algorithm uses these equations and the values of $\Delta[\mathrm{PE}', \mathrm{PE}_j, S]$ to determine both how confident we should be that $\mathrm{C}[\mathrm{PE}'] > \mathrm{C}[\mathrm{PE}_j]$ (Equation 2) and whether any "$\mathcal{T}$-neighbor" of PE_j (*i.e.*, any $\tau_k(\mathrm{PE}_j)$) is more than ϵ better than PE_j (Equation 3).

We close this section with some general comments on the PALO framework and algorithm.

N-PALO1. The samples that PALO uses may be produced by a user of the performance system, who is simply asking questions relevant to his current applications; here, PALO is unobtrusively gathering statistics as the user is solving his own problems [MMS85]. This means that the total cost of the overall system, that both solves performance problems and "learns" by hill-climbing to successive performance elements, can be only marginally more than the cost of only running the performance element to simply solve the performance problems.

We are using these user-provided samples as our objective is to approximate the average utility values of the elements, over the distribution of problems that the performance element will actually address. This "average case analysis" differs from several other approaches as, for example, we do not assume that this distribution of problems will be uniform [Gol79], nor that it will necessarily correspond to any particular collection of "benchmark challenge problems" [Kel87].

N-PALO2. All three $c_\alpha(\mathrm{PE}, q)$ functions discussed in this paper are "bounded"; *i.e.*, satisfy

$$\forall\, \mathrm{PE} \in \mathcal{PE},\, q \in \mathcal{Q}.\; c_\ell \;\leq\; c_\alpha(\mathrm{PE}, q) \;\leq\; c_\ell + \lambda$$

for some constants $c_\ell \in \Re$ and $\lambda \in \Re^+$. Here, we can guarantee that $\Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta] \leq \lambda$. For certain transformations τ_k, we can find yet smaller values for $\Lambda[\tau_k(\mathrm{PE}), \mathrm{PE}]$; see [GJ92].

N-PALO3. Although Theorem 1 bounds the number of samples per iteration, it is impossible to bound the number of iterations of the overall PALO algorithm without making additional assumptions about the search space defined by the $\mathcal{T}$ transformations. The theorem's guarantee that PALO will terminate with probability at least $1 - \delta$ requires that the space of performance elements be finite; this is true in all three situations considered in this paper.

N-PALO4. Notice that a "0-local optimum" corresponds exactly to the standard notion of local optimum; hence our "ϵ-local optimum" generalizes local optimality. Notice that PALO's output, PE_m, will (probably) be a real local optimum if the difference in cost between every two distinct performance elements, PE and $\tau(\mathrm{PE})$, is always larger than ϵ. Thus, for sufficiently small values of ϵ, PALO will always produce a *bona fide* local optimum.

N-PALO5. We can view PALO as a variant on *anytime algorithms* [BD88, DB88] as, at any time, PALO provides a usable result (here, the performance element produced at the j^{th} iteration, PE_j), with the property that later systems are (probably) better than earlier ones; *i.e.*, $i > j$ means $\mathrm{C}[\mathrm{PE}_i] > \mathrm{C}[\mathrm{PE}_j]$ with high probability. PALO differs from standard anytime algorithms by terminating on reaching a point of diminishing returns.

Notice finally that PALO will (probably) process more samples using later elements than using the earlier ones, as its tests (Equations 2 and 3) are increasingly more difficult to pass. This behaviour is desirable, as it means that the overall system is dealing with increasing numbers of samples using later, and therefore better, elements.

4 Instantiations of the PALO Algorithm

This section demonstrates the generality of the PALO algorithm by presenting three different instantiations of this framework. For each instantiation, we specify (1) the set of possible performance elements $\mathcal{PE} = \{\mathrm{PE}_i\}$, (2) the set of transformations $\mathcal{T}$ used in the hill-climbing process, and (3) the scoring function $c(\cdot, \cdot)$ used to specify the expected utility. We will also discuss how to obtain the values of $\Lambda[\tau(\mathrm{PE}), \mathrm{PE}]$. (The instantiations of these parameters are also summarized in Table 1.) For pedagogical reasons, each subsection begins with a quick simplistic description of the application, and then provides notes that describe how to build a more comprehensive system.

4.1 Improving Efficiency

Many derivation processes can be viewed as a satisficing search [SK75] through a given graph structure. As an example, notice that using the information shown in Figure 2 to find an answer to the `hep(κ)` query, for some ground individual κ, corresponds naturally to a search

[4] See [Bol85, p. 12]. *N.b.*, these inequalities holds for essentially *arbitrary distributions*, not just normal distributions, subject only to the minor constraint that the random variables $\{d_i\}$ is bounded.

Rule Set

```
R1:  hep(X) :- jaun(X).
R2:  hep(X) :- badB(X).
R3:  badB(X) :- bt#1(X).
R3:  badB(X) :- bt#2(X).
```

Fact Set

```
jaun(b2), jaun(b3), jaun(b5), ...
bt#1(b1), bt#1(b3), bt#1(b5), ...
bt#2(b6), ...
```

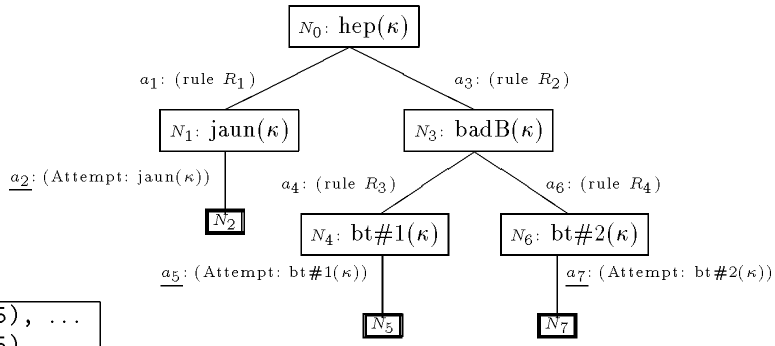

Figure 2: "Inference Graph" G_A, used by Θ_0 and Θ_1

through the G_A inference graph (formed from a given set of rules) seeking a successful database retrieval.[5] A *strategy* specifies the order in which to perform the various rule-based reductions (*e.g.*, the a_1 arc reduces the N_0: `hep(`κ`)` goal to the N_1: `jaun(`κ`)` subgoal, based on the rule R_1) and the database retrievals (*e.g.*, the a_2 arc from N_1 to N_2 corresponds to the attempted database retrieval `jaun(`κ`)`). We can express each strategy as a sequence of G_A's arcs; *e.g.*, the strategy

$$\Theta_0 = \langle a_1, a_2, a_3, a_4, a_5, a_6, a_7 \rangle$$

corresponds to the obvious depth-first left-to-right traversal, with the understanding that the performance element using this strategy will stop whenever it reaches a "success node" (*e.g.*, if the a_2 retrieval succeeds, then Θ_0 reaches the success node N_2 and so stops with success), or has exhausted all of its reductions. (Figure 2 doubly-boxes G_A's success nodes, N_2, N_5 and N_7.) There are many other possible strategies, including

$$\Theta_1 = \langle a_3, a_4, a_5, a_6, a_7, a_1, a_2 \rangle,$$

as well as non–depth-first strategies, etc.

Each strategy will find an answer, if one exists. As this is a satisficing search, all answers are equally acceptable [SK75], which means that all strategies are equally accurate. We can therefore consider the costs of the strategies, preferring the one whose expected cost is minimal.

Letting $f_i \in \Re^+$ be the positive cost of traversing the a_i, we can compute the $c_s(\Theta, q)$, the cost of using strategy Θ to find an answer to the query q. For example, $c_s(\Theta_0, \texttt{hep(b2)}) = f_1 + f_2$, $c_s(\Theta_0, \texttt{hep(b1)}) = f_1 + f_2 + f_3 + f_4 + f_5$, and $c_s(\Theta_1, \texttt{hep(b1)}) = f_3 + f_4 + f_5$. (These different strategies have different costs for a given query as each stops as soon as it finds an answer.) The *expected cost*, of course, depends on the distribution of queries; *i.e.*, on how often the query will be `hep(b1)`, versus `hep(b2)`, etc. Moreover, the task of finding the *globally* optimal strategy is NP-hard [Gre91].

This looks like a job for PALO.[6] We first define the set of reordering transformations $\mathcal{T}^{RO} = \{\tau_{\alpha 1, \alpha 2}\}$, where each $\tau_{\alpha 1, \alpha 2}$ maps one strategy to another by moving the subgraph under the $\alpha 1$ arc to before $\alpha 2$ and its subgraph. For example, $\tau_{a_3, a_1}(\Theta_0) = \Theta_1$, and $\tau_{a_6, a_4}(\Theta_0) = \langle a_1, a_2, a_3, \boxed{a_6, a_7}, a_4, a_5 \rangle$. PALO also needs to compute $\Lambda[\tau(\Theta_\alpha), \Theta_\alpha]$; these values are bounded by $c(G) = \sum_i f_i$, the sum of the costs of all of the arcs in the inference graph G; see Note 2 below.

N-Eff1. This class of performance elements corresponds to many standard problem solvers, including PROLOG [CM81]; see also [GN87]. We can also use these inference graphs to describe operators working in state spaces; here each internal arc of the inference graph corresponds to an operator invocation and each leaf arc to a general "probabilistic experiment". Using G_A, for example, a_3 could encode the "take some blood" operator, and a_5, the experiment that succeeds if the patient tests positive on `bt#1`, etc.

N-Eff2. The companion paper [GJ92] provides more formal descriptions of inference graphs and strategies. That article also presents an efficient analytic way of computing upper and lower bounds of $\Delta[\tau_{\alpha 1, \alpha 2}(\Theta_j), \Theta_j, S]$ (which can be used in Equation 2 and 3, respectively), based only on running Θ_j; this provides a way of obtaining good estimates of $\Delta[\tau_{a1, a2}(\Theta_j), \Theta_j, S]$ *without* first constructing and then executing each $\tau_{a1, a2}(\Theta_j)$ over all $S = \{q_i\}$ queries. It also presents empirical evidence that a system that uses those estimates can still work effectively.

That paper also discusses how this instantiation of the PALO algorithm fits into the framework of "explanation-based learning" systems, and in particular, argues that it provides a mathematical basis for [Min88]'s "utility analysis".

[5] Here, `hep(`χ`)` means χ has hepatitis, `jaun(`χ`)` means χ is jaundiced, and `badB(`χ`)` means χ has "bad blood", `bt#i(`χ`)` means χ tests positive for blood test #i.

[6] Of course, all of the signs in Figure 1 should be flipped, as we are here measuring *cost* rather than utility, and so prefer the element with *minimal*, rather than maximal, cost. Note also that we are viewing each strategy as a performance element.

4.2 Improving Accuracy

A nonmonotonic system can be ambiguous, in that it can produce many individually plausible but collectively incompatible solutions to certain queries [Rei87]. Unfortunately, only (at most) one of these solutions is correct; the challenge then is to determine which one. This is the essence of the "multiple extension problem" in knowledge representation [Rei87, HM86, Mor87], and corresponds to the "bias problem" in machine learning [Mit80, Utg84, RG87, Hau88]. This subsection addresses this problem by seeking a credulous system, related to the given initial nonmonotonic system, that is "optimally correct"; *i.e.*, which produces the correct answer most often.

In more detail, we assume there is a correct answer to each query q, denoted $\mathcal{O}[\,q\,]$; hence $\mathcal{O}[\,2+2\ =\ \texttt{?x}\,]\ =\ \texttt{Yes}[\texttt{?x} \leadsto \texttt{4}]$. Each correct answer is either "Yes" (possibly with a binding list, as shown here) or "No". Using PE(q) to represent the answer returned by the credulous performance element PE, we can define the utility function

$$c_a(\mathrm{PE},\, q) \;\stackrel{def}{=}\; \begin{cases} +1 & \text{if } \mathrm{PE}(q) = \mathcal{O}[\,q\,] \\ 0 & \text{if } \mathrm{PE}(q) = \texttt{IDK} \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

where IDK represents "I don't know".

We focus on stratified THEORIST-style performance elements [PGA86] [Prz87, Bre89, vA90], where each element $\mathrm{PE} = \langle \mathcal{F}, \mathcal{H}, \Theta\rangle$ corresponds to a set of factual information $\mathcal{F}$, a set of allowed hypotheses $\mathcal{H}$ (each a simple type of default [Rei87]) and a specific ordering of the hypotheses. As a specific example, consider $\mathrm{PE}_A = \langle \mathcal{F}_0, \mathcal{H}_0, \Upsilon_A\rangle$, where[7]

$$\mathcal{F}_0 \;=\; \left\{ \begin{array}{l} \forall x.\ \texttt{E}(x)\ \&\ \texttt{N}_E(x) \Rightarrow \texttt{S}(x,\texttt{G}) \\ \forall x.\ \texttt{A}(x)\ \&\ \texttt{N}_A(x) \Rightarrow \texttt{S}(x,\texttt{W}) \\ \forall x.\ \neg\texttt{S}(x,\texttt{G}) \vee \neg\texttt{S}(x,\texttt{W}) \\ \texttt{A(Z)},\quad \texttt{E(Z)},\quad \ldots \end{array} \right\} \tag{5}$$

is the fact set;

$$\mathcal{H}_0 \;=\; \left\{ \begin{array}{ll} h_1: & \texttt{N}_E(x) \\ h_2: & \texttt{N}_A(x) \end{array} \right\}$$

is the hypothesis set, and $\Upsilon_A = \langle h_1,\, h_2\rangle$ is the hypothesis ordering.

To explain how PE_A would process a query, imagine we want to know the color of Zelda — *i.e.*, we want to find a binding for ?c such that σ = "S(Z, ?c)" holds. PE_A would first try to prove σ from the factual information $\mathcal{F}_0$ alone. This would fail, as we do not know if Zelda is a normal elephant or if she is a normal albino (*i.e.*, whether $\texttt{N}_E(\texttt{Z})$ or $\texttt{N}_A(\texttt{Z})$ holds, respectively). PE_A then considers using some hypothesis — *i.e.*, it may assert an instantiation of some element of $\mathcal{H}_0$ if that proposition is both consistent with the known facts $\mathcal{F}_0$ and if it allows us to reach a conclusion to the query posed. Here, PE_A could consider asserting either $\texttt{N}_E(Z)$ (meaning that Zelda is a "normal" elephant and hence is colored *G*ray) or $\texttt{N}_A(Z)$ (meaning that Zelda is a "normal" albino and hence is colored *W*hite). Notice that either of these options, individually, is consistent with everything we know, as encoded by $\mathcal{F}_0$. Unfortunately, we cannot assume both options, as the resulting theory $\mathcal{F}_0 \cup \{\, \texttt{N}_E(Z),\, \texttt{N}_A(Z)\,\}$ is inconsistent.

We must, therefore, decide amongst these options. PE_A's hypothesis ordering Υ_A specifies the priority of the hypotheses. Here $\Upsilon_A = \langle h_1, h_2\rangle$ means that $h_1 :\ \texttt{N}_E(x)$ takes priority over $h_2 :\ \texttt{N}_A(x)$, which means that PE_A will return the conclusion associated with $\texttt{N}_E(Z)$ — *i.e.*, Gray, encoded by $\texttt{Yes}[\texttt{?c} \mapsto G]$, as $\mathcal{F}_0 \cup \{\texttt{N}_E(Z)\} \models \texttt{S(Z, G)}$.[8]

Now consider the $\mathrm{PE}_B = \langle \mathcal{F}_0, \mathcal{H}_0, \Upsilon_B\rangle$ element, which differs from PE_A only in terms of its ordering: As PE_B's $\Upsilon_B = \langle h_2, h_1\rangle$ considers the hypotheses in the opposite order, it will return the answer $\texttt{Yes}[\texttt{?c} \mapsto \texttt{W}]$ to this query; *i.e.*, it would claim that Zelda is white.

Which of these two elements is better? If we are only concerned with this single Zelda query, then the better (read "more accurate") PE_i is the one with the larger value for $c_a(\mathrm{PE}_i, \texttt{S(Z, ?c)})$; *i.e.*, the PE_i for which $\mathrm{PE}_i(\texttt{S(Z, ?c)}) = \mathcal{O}[\,\texttt{S(Z, ?c)}\,]$. In general, however, we will have to consider a less trivial distribution of queries. To illustrate this, imagine Equation 5's "…" corresponds to $\{\texttt{A}(\texttt{Z}_1), \texttt{E}(\texttt{Z}_1), \ldots, \texttt{A}(\texttt{Z}_{100}), \texttt{E}(\texttt{Z}_{100})\}$, stating that each $\texttt{Z}_i$ is an albino elephant; and that the queries are of the form "$\texttt{S}(\texttt{Z}_i\texttt{, ?c})$", for various $\texttt{Z}_i$s.

The best PE_i now depends on the distribution of queries (*i.e.*, how often each "$\texttt{S}(\texttt{Z}_i\texttt{, ?c})$" query is posed) and also on the correct answers (*i.e.*, for which $\texttt{Z}_i$s $\mathcal{O}[\,\texttt{S}(\texttt{Z}_i\texttt{, ?c})\,] = \texttt{Yes}[\texttt{?c} \mapsto \texttt{W}]$ as opposed to $\mathcal{O}[\,\texttt{S}(\texttt{Z}_i\texttt{, ?c})\,] = \texttt{Yes}[\texttt{?c} \mapsto \texttt{G}]$, or some other answer). That is, it depends on the expected accuracy of each system $\mathrm{C}_a[\,\mathrm{PE}_i\,]$, which is defined by plugging Equation 4's $c_a(\cdot,\,\cdot)$ function into Equation 1. We would then select the PE_i system with the larger $\mathrm{C}_a[\,\cdot\,]$ value.

In general, $\mathrm{PE} = \langle \mathcal{F}, \mathcal{H}, \Upsilon\rangle$ can include a much larger set of hypotheses $\mathcal{H} = \{h_1, \ldots h_n\}$. As before, each ordering $\Upsilon = \langle h_{\pi(1)}, \ldots h_{\pi(n)}\rangle$ is a sequence of $\mathcal{H}$'s elements. PE's uses this information when answering queries: Let i be the smallest index such that $\mathcal{F} \cup \{h_i\}$ is consistent and $\mathcal{F} \cup \{h_i\} \models q/\lambda_i$ for some answer λ_i; here PE returns this λ_i. If there are no such i's, then PE returns IDK.

Our goal is identifying the ordering that is accurate most often. Unfortunately, the task of identifying this optimal ordering of the hypotheses is NP-complete even for the simplistic situation we have been considering (where every derivation involves exactly one hypothesis, etc.); see [Gre92].

Once again, PALO is designed to deal with this situation. We first define the set of transformations $\mathcal{T}^A = \{\,\tau_{ij}\,\}_{i,j}$, where each τ_{ij} moves the j^{th} term in the ordering to just before the i^{th} term — *i.e.*, given any ordering $\Upsilon = \langle h_1, h_2, \ldots, h_n\rangle$,

[7] Here Z refers to Zelda, $\texttt{A}(\chi)$ means χ is an albino, $\texttt{E}(\chi)$ means χ is an elephant, and $\texttt{S}(\chi,\ \phi)$ means χ's color is ϕ. The first three clauses in Equation 5 state that normal elephants are gray, normal albinos are white, and (in effect) that S is a function.

[8] This uses the instantiation $\texttt{S(Z, G)} = \texttt{S(Z, ?c)/Yes}[\texttt{?c} \mapsto G]$. We will also view "$q$/No" as "$\neg q$".

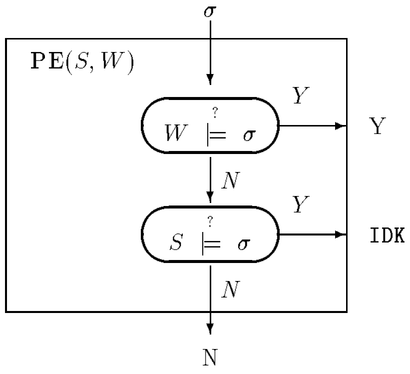

Figure 3: Flow Diagram of PE(S, W) addressing $\Sigma \overset{?}{\models} \sigma$

$\tau_{ij}(\Upsilon) = \langle h_1, \ldots, h_{i-1}, \underline{h_j}, h_i, \ldots, h_{j-1},\ h_{j+1}, \ldots, h_n \rangle$.
We can compute the value of $\Delta[\tau_{ij}(\Upsilon_k),\ \Upsilon_k,\ S]$ for each τ_{ij} transformation and each set of queries S based on whether $\mathcal{F} \cup \{h_\ell\} \overset{?}{\models} q/\mathcal{O}[q]$ for each hypothesis h_ℓ. (Hence, each step of the overall PALO computation is efficient if this test is polynomial time, *e.g.*, if we are dealing with propositional Horn theories, etc.) Observe finally that $\Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta] \leq 2$ for all $\mathrm{PE}_\alpha, \mathrm{PE}_\beta$.

N-Acc1. In many situations, we may want to consider each hypothesis to be the conjunction of a *set* of sub-hypotheses, which must all collectively be asserted to reach a conclusion. Here, we can view $\mathcal{H} = \mathcal{P}[H]$ as the power set of some set of "sub-hypotheses", H.

N-Acc2. Our descriptions have assumed that every ordering of hypotheses is meaningful. In some contexts, there may already be a meaningful partial ordering of the hypotheses, perhaps based on specificity or some other criteria [Gro91]. Here, we can still use PALO to complete the partial ordering, by determining the relative priorities of the initially incomparable elements.

N-Acc3. The motivation underlying this work is similar to the research of [Sha89] and others, who also use probabilistic information to order the various default rules. Our work differs by providing a way of obtaining the relevant statistics, rather than assume that they are known *a priori*.

4.3 Improving Categoricity

The task of determining whether a query is entailed by a theory is known to be intractable if the theory is a general propositional theory. It can, however, be performed efficiently if the theory contains only Horn clauses. Selman and Kautz [SK91] use this observation to define a particular "knowledge compilation" method: Given a general propositional theory Σ, their compiler computes a pair of "bracketing" Horn theories S and W, with the property $S \models \Sigma \models W$.[9] The resulting "compiled system" PE $=$ PE(S, W) uses these bracketing theories to determine whether a query σ follows from Σ, as shown in Figure 3: If $W \models \sigma$, PE terminates with "yes"; otherwise, if $S \not\models \sigma$, then PE terminates with "no". (Notice that these are the correct answers, in that $W \models \sigma$ guarantees that $\Sigma \models \sigma$, and $S \not\models \sigma$ guarantees that $\Sigma \not\models \sigma$. Moreover, these tests are linear in the sizes of σ and S (respectively, σ and W) [DG84].) Otherwise, if $W \not\models \sigma$ and $S \models \sigma$, PE returns `IDK`. Notice this compiled system is usually tractable,[10] yet can deal with an arbitrary propositional theory. It may, however, no longer be completely categoric; hence, we have (potentially) sacrificed complete accuracy for tractability [SK91].

We of course would like to find an approximation $\langle S, W \rangle$ that minimizes the probability that the associated PE(S, W) system will return `IDK`. To state this more precisely: Given any approximation $\langle S,\ W \rangle$ and query σ, let $c_c(\langle S,\ W \rangle,\ \sigma) \overset{def}{=} d(W, \sigma) + (1 - d(S, \sigma))$ where

$$d(S,\ \sigma) \quad \overset{def}{=} \quad \begin{cases} 1 & \text{if } S \models \sigma \\ 0 & \text{otherwise .} \end{cases}$$

Hence, $c_c(\langle S,\ W \rangle,\ \sigma) = 1$ if σ is "covered" by $\langle S,\ W \rangle$, in that either $W \models \sigma$ or $S \not\models \sigma$. Using Equation 1, we can then define $\mathrm{C}_c[\,\langle S,\ W \rangle\,]$ to be the expected value of $c_c(\langle S,\ W \rangle, \cdot)$. Our goal is to determine the approximation $\langle S,\ W \rangle$ with the largest $\mathrm{C}_c[\,\cdot\,]$ value. As before, this task is NP-hard (see [Gre92]) and depends on the distribution, suggesting yet again that we use the PALO system.

Observe that the set of queries covered by a strengthening and a weakening are disjoint — *i.e.*, for any approximation $\langle S, W \rangle$, there is no query σ such that both $W \models \sigma$ and $S \not\models \sigma$. This means an approximation $\langle S_i, W_j \rangle$ is, with probability at least $1 - \delta$, within ϵ of a local optimum if S_i (resp., W_j) is within $\epsilon/2$ of a locally optimal strengthening (resp., weakening) with probability at least $1 - \delta/2$. We can therefore decouple the task of finding a good strengthening from that of finding a good weakening, and handle each separately. This paper discusses only how to finding a good strengthening; [Gre92] merges this with the algorithm that computes a good weakening.

We are seeking a strengthening S_{opt} whose $\mathrm{D}[\,S_{opt}\,]$ value is minimal, where $\mathrm{D}[\,S_{opt}\,] = E[\,d(S, \cdot)\,]$ is the expected value of $d(S, \cdot)$. (Recall we want $S_{opt} \models \sigma$ to *fail* for as many queries as possible.) It is easy to see that this S_{opt} should be a weakest strengthening; *i.e.*, satisfy $\mathrm{OptS}(\Sigma, S_{opt})$ where

$$\begin{aligned} \mathrm{OptS}(\Sigma, S) \quad \Longleftrightarrow \quad & S \models \Sigma \ \&\ \mathrm{Horn}(S)\ \& \\ & [\neg\exists T.\ [S \models T \models \Sigma\ \&\ \mathrm{Horn}(S)\ \&\ S \not\equiv T] \end{aligned}$$

To compute these OptSs: Define a "horn-strengthening" of the clause $\gamma = \{a_1, \ldots, a_k,\ \neg b_1, \ldots, \neg b_\ell\}$ to be any maximal clause that is a subset of γ and is Horn — *i.e.*, each horn-strengthening is formed by simply discarding all but one of γ's positive literals. Here, there are k horn-strengthenings of γ, each of the form $\gamma_j = \{a_j,\ \neg b_1, \ldots, \neg b_\ell\}$.

[9] We call each such S a "<u>S</u>trengthening" of the initial theory, and each such W an "<u>W</u>eakening". This subsection deal with clausal theories; each such theory is a set (conjunction) of clauses, where each clause is a set (disjunction) of atomic literals, each either positive or negative. A theory is Horn if each clause includes at most one positive literal.

[10] Note 2 below explains this caveat.

	Efficiency	Accuracy	Categoricity
Performance Elements $\mathcal{PE}$	satisficing strategies	hypothesis orderings	Horn-strengthenings
Utility function $c(\cdot,\cdot)$	computation time	$\mathrm{PE}(q) \stackrel{?}{=} \mathcal{O}[q]$	$S \stackrel{?}{\models} q$
Tranformations $\mathcal{T}$	reorder arcs	reorder priority	change 1 clause
Range $\Lambda[\tau(\mathrm{PE}), \mathrm{PE}]$	$\leq c(G)$	≤ 2	≤ 1

Table 1: Summary of Applications

Now write $\Sigma = \Sigma_H \cup \Sigma_N$, where Σ_H is the subset of Σ that are Horn and $\Sigma_N = \{\gamma^i\}_{i=1}^m$ is its non-Horn subset. [SK91] proves that each optimal strengthening is of the form $S_o = \Sigma_H \cup \Sigma'_N$, where each $\gamma' \in \Sigma'_N$ is a horn-strengthening of some $\gamma \in \Sigma_N$. By identifying each Horn-strengthened theory with the "index" of the positive literal used (*i.e.*, ${\gamma^i}_j = \{a^i_j, \neg b^i_1, \ldots, \neg b^i_{\ell(i)}\}$), we can consider any Horn-strengthened theory to be a set of the form $S_{\langle j(1), j(2), \ldots, j(m)\rangle} = \Sigma_H \cup \{\gamma^1_{j(1)}, \gamma^2_{j(2)}, \ldots, \gamma^m_{j(m)}\}$.

We can navigate about this space of Horn-strengthened theories by incrementing or decrementing the index of a specific non-Horn clause: That is, define the set of $2m$ transformations $\mathcal{T}^{ID} = \{\tau_k^+, \tau_k^-\}_{k=1}^m$ where each τ_k^+ (resp., τ_k^-) is a function that maps one strengthening to another, by incrementing (resp., decrementing) the "index" of k^{th} clause — *e.g.*, $\tau_k^+(S_{\langle 3,\,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,\,9,\ldots,i_k+1,\ldots,5\rangle}$, and $\tau_i^-(S_{\langle 3,\,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,\,9,\ldots,i_k-1,\ldots,5\rangle}$. (Of course, the addition and subtraction operations wrap around.)

This instantiation of the PALO process starts with any given Horn-strengthened theory (perhaps $S_{\langle 1,\,1,\,\ldots,\,1\rangle}$) and hill-climbs in the space of Horn-strengthened theories, using this set of $\mathcal{T}^{ID}$ transformations. As $\Delta[\tau_k^{\pm}(S_i),\, S_i,\, \sigma]$ depends only on whether $\tau_k^{\pm}(S_i) \models \sigma$ and $S_i \models \sigma$, it can be answered efficiently, as S_i and all $\tau_k^{\pm}(S_i)$ are Horn. (In fact, this process can also use the support of σ from S_i to further improve its efficiency.) Notice finally that $\Lambda[\tau_k^{\pm}(S_i),\, S_i] \leq 1$ for all strengthenings S_i and all $\tau_k^{\pm} \in \mathcal{T}^{ID}$.

N-Cat1. The $\mathrm{PE}(S_i, W_j)$ systems discussed here each return `IDK` if $W \not\models \sigma$ and $S \models \sigma$. [Gre92] proposes several other options for this situation — *e.g.*, perhaps the PE should "guess" at an answer here, or perhaps spend as long as necessary to compute whether $\Sigma \stackrel{?}{\models} \sigma$, etc. — and discusses their relative advantages.

N-Cat2. [Gre92] also presents an algorithm that finds a good weakening. For subtle reasons, that process is slightly different from PALO, and computes a W_g that is close to the *global* optimal, with high probability.

(Unfortunately, the size of the optimal weakening can be exponential in the size of the initial theory, meaning the linear bounds mentioned above are not meaningful. [Gre92] considers ways of finding weakenings that are good with respect to a utility metric that combines both categoricity and efficiency [GE91], to produce a polynomially-sized weakening.)

5 Conclusion

This paper first poses two of the problems that can arise in learning systems that seek a performance element whose expected utility is optimal [Hau90, Vap82]: *viz.*, that the distribution information (which is required to determine which element is optimal) is usually unknown, and that finding a globally optimal performance element can be intractable. It then presents the PALO algorithm that side-steps these shortcomings by using statistical techniques to approximate the distribution, and by hill-climbing to produce a near locally optimal element. After defining this algorithm and specifying its behaviour, we demonstrate its generality by showing how it can be used to find a near-optimal element in three very different settings, based on different spaces of performance elements and different criteria for optimality: efficiency, accuracy and categoricity. (See Table 1.) These results suggest approaches to solving the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

References

[BD88] M. Boddy and T. Dean. Solving time dependent planning problems. Technical report, Brown University, 1988.

[Bol85] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[Bre89] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of IJCAI-89*, Detroit, 1989.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[CM81] W. Clocksin and C. Mellish. *Programming in Prolog*. Springer-Verlag, New York, 1981.

[DB88] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, 1988.

[DeJ88] G. DeJong. AAAI workshop on Explanation-Based Learning. Sponsored by AAAI, 1988.

[DG84] W. Dowling and J. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–84, 1984.

[GD91] J. Gratch and G. Dejong. A hybrid approach to guaranteed effective control strategies. In *Proceedings of IWML-91*, 1991.

[GE91] R. Greiner and C. Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, 1991.

[GJ92] R. Greiner and I. Jurišica. a statistical approach to solving the EBL utility problem. In *Proceedings of AAAI-92*, 1992.

[GN87] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1987.

[GO91] R. Greiner and P. Orponen. Probably approximately optimal derivation strategies. In *Proceedings of KR-91*, 1991.

[Gol79] A. Goldberg. An average case complexity analysis of the satisfiability problem. In *Proceedings of CADE-79*, 1979.

[Gre91] R. Greiner. Finding the optimal derivation strategy in a redundant knowledge base. *Artificial Intelligence*, 50(1):95–116, 1991.

[Gre92] R. Greiner. Probabilistic hill-climbing: Theory and applications. Technical report, Siemens Corporate Research, 1992.

[Gro91] B. Grosof. Generalizing prioritization. In *Proceeding of KR-91*, April 1991.

[Hau88] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 1988.

[Hau90] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. Technical Report UCSC-CRL-91-02, UC Santa Cruz, 1990.

[Hin89] G. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989.

[HM86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, 1986.

[Kel87] R. Keller. Defining operationality for explanation-based learning. In *Proceedings of AAAI-87*, 1987.

[LNR87] J. Laird, A. Newell, and P. Rosenbloom. SOAR: An architecture of general intelligence. *Artificial Intelligence*, 33(3), 1987.

[MCK+89] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–119, 1989.

[Min88] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, 1988.

[Mit80] T. Mitchell. The need for bias in learning generalizations. Technical Report CBM-TR-117, Laboratory for Computer Science Research, May 1980.

[MMS85] T. Mitchell, S. Mahadevan, and L. Steinberg. LEAP: A learning apprentice for VLSI design. In *Proceedings of IJCAI-85*, 1985.

[Mor87] P. Morris. Curing anomalous extensions. In *Proceedings of AAAI-87*, 1987.

[OG90] P. Orponen and R. Greiner. On the sample complexity of finding good search strategies. In *Proceedings of COLT-90*, 1990.

[PGA86] D. Poole, R. Goebel, and R. Aleliunas. Theorist: A logical reasoning system for default and diagnosis. Technical Report CS-86-06, University of Waterloo, 1986.

[Prz87] T. Przymusiński. On the declarative semantics of stratified deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, 1987.

[Qui86] J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Rei87] R. Reiter. Nonmonotonic reasoning. In *Annual Review of Computing Sciences*, volume 2, Annual Reviews Incorporated, 1987.

[RG87] S. Russell and B. Grosof. A declarative approach to bias in concept learning. In *Proceedings of AAAI-87*, 1987.

[Sha89] L. Shastri. Default reasoning in semantic networks: A formalization of recognition and inheritance. *Artificial Intelligence*, 39:283–355, 1989.

[SK75] H. Simon and J. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence*, 6:235–247, 1975.

[SK91] B. Selman and H. Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, 1991.

[Utg84] P. Utgoff. *Shift of Bias for Inductive Concept Learning.* PhD thesis, Rutgers, Laboratory for Computer Science Research, October 1984.

[vA90] P. van Arragon. Nested default reasoning with priority levels. In *Proceedings of CSCSI-90*, 1990.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.

[Vap82] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 1982.