PALO Algorithms

Russell Greiner*

April 2, 1993

Abstract: In general, a "PALO-algorithm" is a learning system that hill-climbs to a locally-optimal performance element, based on information gathered by observing performance elements as they solve performance tasks. Several recent papers have shown that such algorithms can be used effectively for many diverse performance tasks, including finding the optimally efficient satisficing search strategy [GJ92], finding the optimally categorical Horn approximation [GS92], and finding the optimally accurate ordering of default rules [Gre92]. In each case, the quality metric used to measure the worth of an element is its expected utility, based on a given utility function. [Gre92] argues that this learning algorithm is useful when (1) the distribution of samples, which is required to determine these expected values, is not known *a priori*, and (2) the task of computing the optimal element based on this quality metric, even given the distribution information, is intractable. This short note defines these algorithms more precisely, and discusses ways of implementing them efficiently. It also provides a short simulation study, to provide evidence that these algorithms do work effectively.

1 Introduction

A "PALO-algorithm" is a system that hill-climbs through a given space of performance elements $\mathcal{S}_\Theta = \{\Theta_i\}$, using a given set of transformations $\mathcal{T} = \{\tau_j\}$, where each $\tau_j : \mathcal{S}_\Theta \mapsto \mathcal{S}_\Theta$. Each such PALO$_i$ system will (under specified conditions) climb from the given initial Θ_1 performance element to one of Θ_1's neighbors, $\Theta_2 = \tau_{k_1}(\Theta_1)$, and then possibly from this Θ_2 to a different $\Theta_3 = \tau_{k_2}(\Theta_2)$, and so on, until (eventually) reaching a Θ_m, which is returned. The quality function used to rank the different elements is the expected value $\mathrm{C}[\,\Theta\,] \stackrel{def}{=} E[c(\Theta,\, q)]$ of a given utility function $c\colon \mathcal{S}_\Theta \times \mathcal{Q} \mapsto \Re$, over the distribution of samples q, where $c(\Theta,\, q)$ is the "utility" of using the performance element Θ to deal with the individual sample $q \in \mathcal{Q}$. We insist that $c(\cdot,\, \cdot)$ have a bounded range of possible values $\lambda = \lambda(c) \in \Re^+$; i.e.,

$$\exists a_c \in \Re,\ \forall \Theta \in \mathcal{S}_\Theta,\ \forall q \in \mathcal{Q}\colon\ a_c \ \le\ c(\Theta,\, q)\ \le\ a_c + \lambda\ . \tag{1}$$

PALO has access to an oracle that draws from a set of samples $\mathcal{Q} = \{q_j\}$ at random, according to a fixed (but unknown) distribution $P\colon \mathcal{Q} \mapsto [0, 1]$. The definition below specifies such "PALO-algorithms":

*This note is written for researchers who have expressed an interested in using a PALO-like system for some application. It is *not* as a stand-alone document; we assume its readers have already read at least one of the articles mentioned in the abstract. We are actively seeking feedback on both these ideas, and these algorithms; please send any relevant comments to `greiner@learning.scr.siemens.com`. Finally, I gratefully thank William Cohen and Dale Schuurmans for their earlier contributions to this work; and Michael Braverman for his recent comments on this document.

Algorithm PALO$_1$(Θ_1, $\mathcal{T}$, ϵ, δ)
  For $j = 1..\infty$ do
    Let $\delta_j \leftarrow \frac{\delta}{j^2}\frac{6}{\pi^2}$,
      $\mathcal{T}[\Theta_j] \leftarrow \{\tau(\Theta_j) \in \mathcal{S}_\Theta | \tau \in \mathcal{T} \ \&\ \tau(\Theta_j) \neq \Theta_j\}$,   *% $\mathcal{T}[\Theta_j]$ are Θ_j's neighbors*
      $L_j \leftarrow \left\lceil 2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{2|\mathcal{T}[\Theta_j]|}{\delta_j}\right\rceil$ *% L_j is max # of samples, Θ_j iteration* $\langle L1\rangle$
    ForEach $\Theta' \in \mathcal{T}[\Theta_j]$ **do** **Let** $\Delta(\Theta_j, \Theta', 0) \leftarrow 0$.
    For $i = 1..L_j$ do   *% Get and process i^{th} query*
      Get sample q_i (from oracle)
      ForEach $\Theta' \in \mathcal{T}[\Theta_j]$ **do**
        Let $\Delta(\Theta_j, \Theta', i) \leftarrow \Delta(\Theta_j, \Theta', i{-}1) + [c(\Theta', q_i) - c(\Theta_j, q_i)]$. $\langle L2\rangle$
      If $i < L_j$
      Then
        If $\exists \Theta' \in \mathcal{T}[\Theta_j]$ s.t. $\frac{1}{i}\Delta(\Theta_j, \Theta', i) > \Lambda(\Theta_j, \Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2(L_j-1)|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ $\langle L3\rangle$
        Then **Let** $\Theta_{j+1} \leftarrow \Theta'$
          Exit For (inner loop)
        Else If $\forall \Theta' \in \mathcal{T}[\Theta_j]$: $\frac{1}{i}\Delta(\Theta_j, \Theta', i) < \epsilon - \Lambda(\Theta_j, \Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2(L_j-1)|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ $\langle L4\rangle$
          Then **Return**[Θ_j] (Exiting both inner and outer For Loops)
      Else
        If $\exists \Theta' \in \mathcal{T}[\Theta_j]$ s.t. $\frac{1}{L_j}\Delta(\Theta_j, \Theta', L_j) > \frac{\epsilon}{2}$ $\langle L5\rangle$
        Then **Let** $\Theta_{j+1} \leftarrow \Theta'$
          Exit For (inner loop)
        Else **Return**[Θ_j]. (Exiting both inner & outer For loops)
    End For (inner loop)
  End For (outer loop)
End PALO$_1$

Figure 1: Code for PALO$_1$ Algorithm

Definition 1 *An algorithm P is a "PALO-Algorithm" iff $P(\Theta_1, \epsilon, \delta, \mathcal{T})$ incrementally produces a series of performance elements $\Theta_1, \Theta_2, \ldots, \Theta_m$, such that each $\Theta_{j+1} = \tau(\Theta_j)$ for some $\tau \in \mathcal{T}$ and, with probability at least $1-\delta$,*

1. *the expected utility of each performance element is strictly better than its predecessors* i.e.,
   $\forall 1 \leq i < j \leq m: \ C[\Theta_j] > C[\Theta_i]$; *and*
2. *the final performance element returned by P, Θ_m, is an "ϵ-local optimum"* — i.e.,
   $\neg\exists \tau_j \in \mathcal{T}: \ C[\tau_j(\Theta_m)] \geq C[\Theta_m] + \epsilon$.

Moreover, P will stay at any Θ_j (before either terminating or climbing to a new Θ_{j+1}) for a number of samples that is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\lambda(c)$ and $|\mathcal{T}[\Theta_j]|$ where $\mathcal{T}[\Theta_j] = \{\tau(\Theta_j) \,|\, \tau \in \mathcal{T} \ \&\ \tau(\Theta_j) \neq \Theta_j\}$ is the set of Θ_j's neighbors, and will terminate with probability 1, provided $|\mathcal{T}[\Theta]|$ is finite for all reached Θ. □.

2 PALO$_1$ Algorithm, and Variants

This short summary focuses on one particular PALO-algorithm: PALO$_1$ shown in Figure 1. It then discusses various ways of implementing such a system, and presents two other PALO-algorithms, viewing each as a variant of PALO$_1$.

To define the terms used by this algorithm: For any performance element $\Theta \in \mathcal{S}_\Theta$ and transformation $\tau \in \mathcal{T}$, let

$$\begin{array}{rcl} \Lambda(\Theta,\Theta') & = & \max_{q\in\mathcal{Q}}\{\ c(\Theta'),\ q) - c(\Theta,\ q)\ \} \quad - \quad \min_{q\in\mathcal{Q}}\{\ c(\Theta',\ q) - c(\Theta,\ q)\ \} \\ \Lambda[\Theta] & = & \max_{\Theta'\in\mathcal{T}[\Theta]}\{\ \Lambda(\Theta,\Theta')\ \} \end{array} \tag{2}$$

Hence, $\Lambda(\Theta,\Theta')$ is the size of the range of possible values for $c(\Theta',\ q) - c(\Theta,\ q)$, over all samples $q \in \mathcal{Q}$; and $\Lambda[\Theta]$ is the largest such range for a given Θ, over all $\Theta' \in \mathcal{T}[\Theta]$. Notice that $\Lambda[\Theta] \le 2\lambda$, for each Θ (using the λ defined in Equation 1). We also define

$$\mathcal{T}[\Theta] \quad = \quad \{\ \tau(\Theta)\ |\ \tau \in \mathcal{T}\ \&\ \tau(\Theta) \neq \Theta\ \} \tag{3}$$

to be the set of Θ's non-degenerate neighbors. *N.b.*, we are only considering the *set* of such $\tau(\Theta)$ elements; hence, $\mathcal{T}[\Theta]$ will only include $\tau(\Theta)$ once, even if $\tau(\Theta) = \tau'(\Theta)$ for some other $\tau' \neq \tau$. This means that $|\mathcal{T}[\Theta_j]|$ can be much smaller than $|\mathcal{T}|$, as there can be many $\tau \in \mathcal{T}$ that map Θ_j to Θ_j, and many other pairs $\langle \tau_k, \tau_\ell \rangle$ where $k \neq \ell$ such that $\tau_k(\Theta_j) = \tau_\ell(\Theta_j)$.

To summarize the code: PALO$_1$ examines a sequence of queries, one by one. For each sample, PALO$_1$ computes the utility of the given Θ_1 over the queries seen so far, and compares that value with comparable values for each of Θ_1's neighbors (Line $\langle L2 \rangle$). If any neighbor appears to be significantly better (Line $\langle L3 \rangle$), it become the new performance element Θ_2, and PALO$_1$ then compares Θ_2's performance with that of Θ_2's neighbors, over the next set of samples. Once again, if any of Θ_2's neighbors appears much better, PALO$_1$ will climb to this apparently-superior element Θ_3, and so forth. On the other hand, if no neighbor looks much better, PALO$_1$ will exercise other portions of its code: If none of Θ_j's neighbors look much better than the current Θ_j (Line $\langle L4 \rangle$), PALO$_1$ will terminate, returning Θ_j. If neither of these conditions holds, PALO$_1$ will, in general, simply loop around to process another query, then observe how the current Θ_j compares with its neighbors, using this query in additional to the previous ones. However, if PALO$_1$ has stayed on this current Θ_j for a large enough number of queries, PALO$_1$ will use easier thresholds to decide whether to climb to some Θ_{j+1} or terminate (Line $\langle L4 \rangle$), and will necessarily perform one of those actions. (This number is based on the L_j from Line $\langle L1 \rangle$.)

Theorem 1, in Appendix A, proves that PALO$_1$ qualifies as a PALO-algorithm.

Comments and Efficiency Hints: First, each Θ performance system will produce an answer/response when processing each sample q. We assume that the PALO$_i$ learning system will make available the answer that is produced by the current Θ_j system, perhaps as a side effect.

Second, notice that PALO$_1$ can accommodate an infinite number of transformations $\mathcal{T}$; for it to work effectively, we require only that $|\mathcal{T}[\Theta]|$ be finite for each Θ reached; see [CGS91].

Third, PALO$_1$ has an awkward set of tests to perform: First checking if $i < L_j$, and then performing different tests depending on the result of that initial comparison. We could omit that test if we had used a L_j such that $\Lambda[\Theta_j]\sqrt{\frac{1}{2L_j}\ln\left(\frac{2\,(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)} \le \frac{\epsilon}{2}$. However, the value of such a L_j is much larger than the value defined on line $\langle L1 \rangle$; we estimate this would cost at least a multiplicative factor of about $7\ln(\frac{\lambda}{\epsilon})$, which was deemed too high a price to pay only for an increase in elegance.

Fourth, the only relevant property of the $\delta_j \leftarrow \frac{\delta\ 6}{j^2\ \pi^2}$ values is that $\sum_{j=1}^{\infty}\delta_j = \delta$. We could have used any other sequence that has this property, including for example, Rissanen's $\delta'_j \leftarrow \frac{\delta}{2.865064\, j\, \log^*(j)}$ sequence [Ris83].

Finally, Line $\langle L2 \rangle$ of the PALO system uses the values $c(\Theta',\ q_i) - c(\Theta_j,\ q_i)$ for each $\Theta' \in \mathcal{T}[\Theta_j]$. Of course, we can compute these values by first constructing each neighboring $\tau_k(\Theta_j)$ element, and then computing the $c(\tau_k(\Theta_j),\, q_i)$ quantity for each $q_i \in S$ sample, etc. This forces PALO_1 to run $(|\mathcal{T}|+1)$ performance elements on each sample; and hence $(|\mathcal{T}|+1)\times i$ performance-element–sample runs to deal with i samples.

These Δ values can often be computed more efficiently, however, based only on the results obtained from running the single Θ_j on the elements q_i. For this, we need a $\varphi(\Theta, \tau, q)$ function that analytically computes the value of $c(\tau(\Theta),\, q) - c(\Theta,\, q)$, based only on the statistics gathered by observing Θ (alone) deal with q; see [GJ92].

Two relevant observations about the $\varphi(\,\Theta_j,\, \tau_k,\, q_i\,)$ computation:
[1] In many situations, it is easy to see that the result of the τ_k transformation, $\tau_k(\Theta)$, will have the same performance on a particular sample as the base Θ; here, the value of $\varphi(\,\Theta_j,\, \tau_k,\, q_i\,)$ will necessarily be 0. If this occurs frequently, one could use a variant of PALO_1 that does not consider accumulate statistics for $\Delta(\,\Theta_j,\, \underline{\tau_k}(\Theta_j),\, \cdot\,)$ unless it observes a sample for which $\varphi(\,\Theta_j,\, \underline{\tau_k},\, q_i\,) \neq 0$. Of course, each distinct $\tau_k(\Theta_j)$ must still be included in the size of $|\mathcal{T}[\Theta_j]|$ which is used in computing L_j, even if $\varphi(\,\Theta_j,\, \tau_k,\, q_i\,) = 0$ for all L_j samples.

[2] In some situations, it is much easier to compute bounds for $c(\tau(\Theta),\, q) - c(\Theta,\, q)$, rather than compute its exact value. Here, we can allow $\varphi(\Theta_j, \tau, q)$ to return a *pair* of values that bracket the correct answer; $\varphi(\Theta, \tau, q) = \langle \varphi_L(\Theta, \tau, q),\ \varphi_U(\Theta, \tau, q)\rangle$, where

$$\varphi_L(\Theta, \tau, q) \quad \leq \quad c(\tau(\Theta),\, q) - c(\Theta,\, q) \quad \leq \quad \varphi_U(\Theta, \tau, q)\ .$$

An obvious variant of PALO_1 — call it PALO_{1U} — could then use these bracketing values appropriately. In particular, PALO_{1U}'s Line $\langle L2\rangle$ would use $\varphi_U(\Theta, \tau, q)$, rather than $c(\tau(\Theta),\, q) - c(\Theta,\, q)$, in computing its $\Delta(\,\Theta_j,\, \tau,\, i\,)$ values, here called $\Delta^{(U)}(\,\Theta_j,\, \tau(\Theta_j),\, i\,)$. As $\varphi_U(\Theta, \tau, q) \geq c(\tau(\Theta),\, q) - c(\Theta,\, q)$ means $\Delta^{(U)}(\Theta_j,\, \tau(\Theta_j),\, i) \geq \Delta(\Theta_j,\, \tau(\Theta_j),\, i)$, finding any $\Delta^{(U)}(\Theta_j, \tau(\Theta_j), i)$ either passing Line $\langle L4\rangle$ or failing Line $\langle L5\rangle$ is sufficient to conclude that the corresponding $\Delta(\Theta_j,\, \tau(\Theta_j),\, i)$ would have performed similarly; hence, we know that the probability that PALO_{1U} will terminate "incorrectly" is bounded by the probability that PALO_1 will terminate incorrectly, as desired. Unfortunately, this PALO_{1U} may climb too soon — *i.e.*, we cannot bound the probability that it will climb to $\tau(\Theta_j)$ even though $\mathrm{C}[\,\tau(\Theta_j)\,] < \mathrm{C}[\,\Theta_j\,]$. We can similarly define the more cautious PALO_{1L} algorithm which uses φ_L rather than φ, and PALO_{1LU}, which uses the arithmetic mean $\frac{1}{2}[\varphi_L(\Theta, \tau, q) + \varphi_U(\Theta, \tau, q)]$. Finally, the PALO_{1P} variant uses the statistics gathered so far to produce an estimate $\varphi_P(\Theta, \tau, q)$ that is between the lower and upper bounds, $\varphi_L(\Theta, \tau, q)$ and $\varphi_U(\Theta, \tau, q)]$. (Section 4 presents concrete examples of these functions.)

Implementation Hints: Our implementation of PALO_1 uses a few subroutines, which depend on the particular class of performance elements $\mathcal{S}_\Theta$ and transformations $\mathcal{T}$ used.

INIT_DELTA$(\Theta_j,\, \mathcal{T})$ takes a performance element Θ_j and set of transformations $\mathcal{T}$ and produces two $m = |\mathcal{T}[\Theta_j]|$ element vectors: $\Gamma = \langle \gamma_1,\, \ldots,\, \gamma_m\rangle$ and $\Upsilon = \langle \upsilon_1,\, \ldots,\, \upsilon_m\rangle$ where each γ_k represents the value of $\Delta[\Theta_j,\, \tau_k(\Theta_j),\, \cdot]$, each initialized to 0, and each $\upsilon_k = \Lambda(\Theta_j, \tau_k(\Theta_j))$. Our subroutine also computes and returns two other quantities: the largest of these $\Lambda(\Theta_j, \tau(\Theta_j))$ values $\Lambda[\Theta_j] = \max_\tau\{\Lambda(\Theta_j, \tau(\Theta_j)\}$, and $m = |\mathcal{T}[\Theta_j]|$.

UPDATE_DELTA$(\Theta_j,\, \langle q, a\rangle,\, \Gamma^{(i-1)},\, \Upsilon)$ takes a performance element Θ_j, a sample q along with the "answer" returned a (and whatever other data may be useful), and the $\Gamma = \Gamma^{(i-1)}$ and Υ vectors defined above, where each $\gamma_k^{(i-1)}$ in $\Gamma^{(i-1)}$ is now the $\Delta[\Theta_j, \tau_k, i-1]$ value obtained after the previous $i-1$ samples. (Hence, view this q as the i^{th} sample.) It returns the updated $\Gamma^{(i)}$ vector, whose elements $\gamma_k^{(i)} = \Delta[\Theta_j, \tau_k(\Theta_j), i] \;=\; \Delta[\Theta_j, \tau_k(\Theta_j), i-1] + \varphi(\Theta_j, \tau_k, q)$. As it is processing these vector elements, UPDATE_DELTA can also determine

- if any $\frac{1}{i}\gamma_k^{(i)}$ exceeds the $\epsilon_i(\tau_k) = \Lambda(\Theta_j, \tau_k(\Theta_j))\sqrt{\frac{1}{2i}\ln\left(\frac{2(L_j-1)|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ threshold (see Theorem 1 below)
- if, for every τ_k, $\frac{1}{i}\gamma_k^{(i)}$ is below $\epsilon - \epsilon_i(\tau_k)$

and return that information as well. (If there are several τ_ks for which $\frac{1}{i}\gamma_k^{(i)} > \epsilon_i(\tau_k)$, our algorithm will select the k for which $\frac{1}{i}\gamma_k^{(i)} - \epsilon_i(\tau_k)$ is largest.) Each $\epsilon_i(\tau_k)$ value is based on the relevant $v_k = \Lambda(\Theta_j, \tau_k(\Theta_j))$ value.

Notice UPDATE_DELTA needs to compute new $\gamma_k^{(i)}$ values, and test if $\frac{1}{i}\gamma_k^{(i)} > \epsilon_i(\tau)$, only if the corresponding $\varphi(\Theta_j, \tau_k, q)$ value is non-0. Notice, however, that it must perform the $\frac{1}{i}\gamma_k^{(i)} <^? \epsilon - \epsilon_i(\tau_k)$, test for all τ_ks.

NEW_PE($\langle\Theta\rangle, j$) takes a description $\langle\Theta\rangle$ of the performance element Θ_j and the iteration number (which is the j using in Figure 1), and does the proper initializations. In particular, it uses INIT_DELTA to compute the appropriate $\Gamma^{(1)}$ and Υ vector as well as $\Lambda[\Theta_j]$ and $|\mathcal{T}[\Theta_j]|$ which is used to compute the value of L_j.

3 Other PALO-Algorithms

The simpler "on-line but batched" PALO$_0$ system differs from PALO$_1$ by completely skipping the climb test in Line $\langle L3\rangle$ and termination test in Line $\langle L4\rangle$; hence, the inner **For** loop is used simply to collect the L_j samples and compute the relevant $\Delta(\Theta_j, \tau_k, L_j)$ values. PALO$_0$ therefore requires a slightly smaller number of samples on each iteration; PALO$_0$'s Line $\langle L1\rangle$ is reduced to[1]

$$\textbf{Let} \quad L_j^{(0)} \leftarrow \left\lceil 2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{|\mathcal{T}[\Theta_j]|}{\delta_j}\right\rceil \qquad \langle L1^{(0)}\rangle$$

It is trivial to prove that the resulting PALO$_0$ system is a PALO-algorithm; see [CGS91].

The second variant, PALO$_2$, resembles PALO$_0$ as it is also on-line but batched. It differs by hill-climbing only a bounded number of times: at most only $K = \left\lceil\frac{2\lambda}{\epsilon}\right\rceil$ times, based on the λ defined in Equation 1 above. At the beginning of the j^{th} stage (for $j \geq 2$), this algorithm will already have an estimate of $\mathrm{C}[\,\Theta_j\,]$, call it $\hat{\mathrm{C}}[\,\Theta_j\,]$, that is, with probability at least $1 - \frac{\delta}{K|\mathcal{T}[\Theta_j]|}$, within $\frac{\epsilon}{4}$ of the corresponding $\mathrm{C}[\,\Theta_j\,]$; *i.e.*, $Pr[\,|\hat{\mathrm{C}}[\,\Theta_j\,] - \mathrm{C}[\,\Theta_j\,]| \leq \frac{\epsilon}{4}\,] \quad \geq \quad 1 - \frac{\delta}{K|\mathcal{T}[\Theta_j]|}$. PALO$_2$ will then use

$$L_j^{(2)} \quad = \quad 2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{2K|\mathcal{T}[\Theta_j]|}{\delta} \tag{4}$$

samples to estimate the expected value of each neighboring $\tau_k(\Theta_j)$ to within $\frac{\epsilon}{4}$, again with probability at least $1 - \frac{\delta}{K|\mathcal{T}[\Theta_j]|}$. PALO$_2$ will climb to one such $\Theta_{j+1} = \tau_k(\Theta_j)$ if that estimate is at least $\epsilon/2$ larger than the estimated value for Θ_j; *i.e.*, if $\hat{\mathrm{C}}[\,\Theta_{j+1}\,] > \hat{\mathrm{C}}[\,\Theta_j\,] + \frac{\epsilon}{2}$; and will terminate, returning Θ_j, if there are no such $\tau_k(\Theta_j)$s. To deal with the first (*i.e.*, $j = 1$) iteration, PALO$_2$ will use

$$L_1^{(2)} \quad = \quad 2\left(\frac{\Lambda[\Theta_1]}{\epsilon}\right)^2 \ln\frac{2\,K\,(1 + |\mathcal{T}[\Theta_1]|)}{\delta} \tag{5}$$

[1] Here, and below, we will use superscripts to indicate variables and line labels in PALO$_i$ systems. For example, PALO$_0$'s line $\langle L1^{(0)}\rangle$ corresponds to PALO$_1$'s line $\langle L1\rangle$, and PALO$_0$'s variable $L_j^{(0)}$ corresponds to PALO$_1$'s L_j.

samples to obtain empirical estimates of the $1 + |\mathcal{T}[\Theta_1]|$ values, $\mathrm{C}[\,\Theta_1\,]$ and each $\mathrm{C}[\,\tau_k(\Theta_1)\,]$.)

Notice that $\hat{\mathrm{C}}[\,\Theta_{j+1}\,] \;\geq\; \hat{\mathrm{C}}[\,\Theta_1\,] \;+\; j\frac{\epsilon}{2}$, as the empirical average of each $\hat{\mathrm{C}}[\,\Theta_{j+1}\,]$ is at least $\frac{\epsilon}{2}$ larger than $\hat{\mathrm{C}}[\,\Theta_j\,]$. As each $c(\Theta_j,\,q)$ is bounded (see Equation 1), PALO$_2$ can climb at most K times. (If it climbed $K+1$ times, then the empirical average of the resulting $\Theta_{(K+1)+1}$ would be at least $\hat{\mathrm{C}}[\,\Theta_{K+2}\,] \;\geq\; \hat{\mathrm{C}}[\,\Theta_1\,] \;+\; (K+1)\frac{\epsilon}{2} \;>\; a_c \;+\; \frac{2\lambda}{\epsilon}\frac{\epsilon}{2} \;=\; a_c + \;\lambda$, which is too large.) To see that the resulting PALO$_2$ system qualifies as a PALO-algorithm, notice that it climbs, or terminates, appropriately if each of its estimates is correct (*i.e.*, within $\frac{\epsilon}{4}$), and the probability that any of its estimates is incorrect is bounded by

$$(1 + |\mathcal{T}[\Theta_1]|)\frac{\delta}{K\,(1 + |\mathcal{T}[\Theta_1]|)} \;+\; \sum_{i=2}^{K} |\mathcal{T}[\Theta_i]|\frac{\delta}{K\,|\mathcal{T}[\Theta_i]|} \quad=\quad \frac{\delta}{K} \;+\; (K-1)\frac{\delta}{K} \quad=\quad \delta$$

as desired.

Each of the above variants works effectively for *any* distribution of "samples", where each sample is of the form $X_\ell = c(\tau(\Theta),\,q_\ell) - c(\Theta,\,q_\ell)$, for some given element Θ and transformation τ. We now define the PALO$_{1N}$ variant, which is only guaranteed to qualify as a PALO-algorithm if these samples are normally distributed.[2] To state this more precisely, we continue to assume that these $\{X_\ell\}$ are independent, identically-distributed random variables, with common mean $\mu = \mathrm{C}[\,\tau(\Theta)\,] - \mathrm{C}[\,\Theta\,]$; here, we also assume that each X_ℓ is normally distributed; *i.e.*, $X_\ell \sim \mathcal{N}(\mu, \sigma)$.

PALO$_{1N}$ differs from PALO$_1$ only by using different constants:

$$\textbf{Let}\quad L_j^{(N)} \;\leftarrow\; \left\lceil \left(\frac{\Lambda[\Theta_j]}{\epsilon}\, z\!\left(1 - \frac{\delta_j}{2|\mathcal{T}[\Theta_j]|}\right)\right)^2 \right\rceil \qquad \langle L1^{(N)}\rangle$$

where $z(p) \;=\; \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{p} e^{-\frac{x^2}{2}}\,dx$ is the p^{th} quantile of the standard normal distribution $\mathcal{N}(0,1)$ (*i.e.*, the area under the standard normal distribution function to the left of $z(p)$ is p); and the $\Lambda(\Theta_j, \tau(\Theta_j))\sqrt{\frac{1}{2i}\ln\left(\frac{2\,(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ value used in both $\langle L3\rangle$ and $\langle L4\rangle$ is replaced with

$$\epsilon_i^{(N)} \quad=\quad \sqrt{\frac{s^2(i)}{i}}\;\; t_{i-1}\!\left(1 - \frac{\delta_j}{2(L_j^{(N)} - 1)|\mathcal{T}[\Theta_j]|}\right) \qquad (6)$$

where

$$s^2(i) = \frac{1}{i-1}\left[\sum_{\ell=1}^{i} X_\ell^2 - \frac{1}{i}\left(\sum_{\ell=1}^{i} X_\ell\right)^2\right]$$

is an unbiased estimator of the variance of the X_ℓ variables after i samples, and $t_k(p)$ is the p^{th} quantile of the Student's T distribution with k degrees of freedom, $\mathcal{T}_k$. For subtle reasons, PALO$_{1N}$ will skip the early climb and early termination tests (on Lines $\langle L3\rangle$ and $\langle L4\rangle$) until $i = 4$.[3] (Theorem 2 in the appendix proves that PALO$_{1N}$ qualifies as a PALO-algorithm.)

Comparison of PALO$_i$ Algorithms: Table 1 explicit shows of the major differences between the PALO$_i$ systems. We have presented only 4 of the $2^3 = 8$ possible systems defined by this table;

[2]This system was motivated by the work of Gratch and deJong [GD92], who also built a hill-climbing system which assumed that the sample mean would be normally distributed.

[3]We also experimented with larger minimal values of i, up to $i = 30$, but found that they did not significant affect PALO$_{1N}$'s performance.

	Arbitrary Distributions	Climb/Terminate Early	Bounded # of Climbs
PALO_0	+	−	−
PALO_1	+	+	−
PALO_2	+	−	+
PALO_{1N}	−	+	−

Table 1: Different Features for the Different PALO_i Systems

there are obvious ways of defining the obvious remaining systems. Of course, there are yet other possible PALO-ish systems which, for example, might consider climbing early, but not terminating early (*i.e.*, it would keep Line $\langle L3\rangle$'s test, but eliminate Line $\langle L4\rangle$'s). While this PALO_3 would have essentially the sample complexity as PALO_1, it might have a slightly reduced computational cost. Furthermore, any of these PALO_i systems can trivially use the various modifications discussed for PALO_1; in particular, any can use $\langle \varphi_L, \varphi_U\rangle$ rather than $c(\tau(\Theta_j), q_i) - c(\Theta_j, q_i)$, etc.

Notice first that PALO_{1N} differs from the other PALO_i systems as it is not guaranteed to apply to arbitrary distributions. To compare the other three algorithms: As they all qualify as PALO-algorithms, all have essentially the same functionality. They will, however, have different computational costs. The primary cost is the time required to collect the relevant samples, as the only other costs are trivial: just gathering simple statistics and performing trivial arithmetic computations. We will therefore focus on their respective "sample complexities". The obvious measure would consider the total number of samples required before the PALO_i system terminates. Notice, however, that a learning+performance system based on any of the PALO_is will probably continue to use the locally-optimal Θ_m, even after PALO_i has terminated. We therefore use a slightly different measure, defined to be the number of samples PALO_i uses *before it reaches its final performance element* Θ_m; *n.b.*, this measure ignores the samples used afterwards, to establish that this Θ_m is an ϵ-local optimum.

The following comparisons are based on this measure. We use terms like "usually" in a very vague way, and only suggest plausible reasons for why certain claims may hold; the empirical results in the next section provide additional evidence supporting our intuitions.

It seems clear that PALO_1 will usually be superior to PALO_0, as on each iteration, PALO_1 has the potential of climbing much earlier, and hence of reaching the final Θ_m after fewer total samples. While there is a cost for this potential increased efficiency, that cost is only an additive $2(\ln 2)\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \approx 1.39\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2$ constant in the *worse* case, which is usually quite small.

It is less obvious which of PALO_0 and PALO_2 is superior. On one hand, the guarantees associated with PALO_2 are stronger, as we see that it will terminate in polynomial time. However, it may take longer than PALO_0 to reach a good performance element as PALO_2 initially has a higher "barrier" at each stage; in fact, the sample complexity associated with PALO_2's j^{th} stage, $L_j^{(2)}$, will be larger than the sample complexity associated with PALO_0's j^{th} stage L_j, until j grows larger than $\frac{2}{\pi}\sqrt{\frac{6\lambda}{\epsilon}}$, after which PALO_2 will require fewer samples per iteration. Hence, as PALO_0 initially climbs faster, it will spend fewer samples dealing with inferior elements than PALO_2 will.

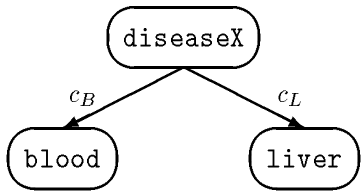

Figure 2: Doctor's "Disjunctive Decision Tree"

4 Simulation Studies

To get some feel for these different systems, we implemented various different PALO$_i$-algorithms, and ran some simulations on a simple performance task. To explain this underlying task: Assume a person has `diseaseX` iff she has either a positive `blood` test or a positive `liver` test. The physician could then diagnose a patient by first performing a `blood` test, and if that test is positive, stop, knowing the patient has `diseaseX`. If that test fails, however, the doctor could then perform the `liver` test, and know that the patient has `diseaseX` iff that test is positive. This sequence, of performing the blood test and then, if necessary, performing the liver test, is one possible strategy, designated $\Theta_1 = \langle B, L\rangle$. There is another possible strategy, $\Theta_2 = \langle L, B\rangle$, which involves first performing the liver test and then, if necessary, the blood test.

Our goal is to find the strategy whose expected cost, in practice, is minimal.[4] We assume we know the costs of the individual experiments, designated c_B and c_L in Figure 2. Hence,

$$c(\Theta_1,\, q) \quad = \quad \begin{cases} c_B & \text{if } q\text{'s } \texttt{blood} \text{ test is positive} \\ c_B + c_L & \text{otherwise} \end{cases}$$

The expected cost of Θ_1 is

$$\mathrm{C}[\,\Theta_1\,] \quad = \quad E[c(\Theta_1,\, q)] \quad = \quad (p_B \times c_B) + [(1-p_B)\times(c_B+c_L)] \quad = \quad c_B + (1-p_B)\times c_L$$

where p_B is the probability that a patient's blood will test positive, over the distribution of patients that the doctor will examine. (Similarly, $\mathrm{C}[\,\Theta_2\,] = c_L + (1-p_L)\times c_B$.)

Of course, the physician will prefer the strategy whose cost is less; hence, he will use Θ_1 iff $\frac{p_B}{c_B} \leq \frac{p_L}{c_L}$. As he, unfortunately, does not know the probability values $p = \langle p_L, p_B\rangle$, he may use some statical sampling technique to obtain good estimates. In this particular context, he can then trivially determine the optimal strategy (read "performance element"), once he learns this distribution information. In general, however, the task of computing the optimal strategy for more complicated structures is intractable; *cf.*, [Gre91]. These are exactly the motivations for our PALO approach.

To exercise our PALO$_i$-algorithms, we used a slight extension of the doctor's situation, in which there are k experiments $W = \{e_i\}_{i=1}^k$, each with a positive cost $c_i = c(e_i) \in \Re^+$, and also a success probability $p_i = p(e_i)$. (Of course, these p_i values are not known to the learning algorithm. Notice we are also implicitly assuming that these experiments are independent; that is, the probability that e_j succeeds will not change if some other experiment e_i ($e_i \neq e_j$) fails.)

Each transformation τ_{ij} re-arranges a given strategy by moving the experiment occurring in the i^{th} position to become the j^{th} experiment. Hence, $\tau_{24}(\langle e_1, e_2, e_3, e_4\rangle) = \langle e_1, e_4, e_2, e_3\rangle$, and

[4] As our metric is trying to find *smaller*, not larger, values, we need to flip the signs on the various algorithms.

$\tau_{13}(\langle e_1, e_4, e_2, e_3\rangle) = \langle e_4, e_2, e_1, e_3\rangle$, etc. The space of transformations $\mathcal{T} = \{\tau_{ij}\}_{i,j}$ is all $(k-1)^2$ distinct transformations.

Our simulations focused on the following situation: Our disjunctive decision structure used the ten experiments, $\{e_i\}_{i=1}^{10}$, each with success probability $p_i = 1/2$, where $c_i = 2^{(10-i)}$, hence $c_1 = 512$, $c_2 = 256$, …, $c_{10} = 1$. Notice the worst possible strategy is $\Theta_{1..10} = \langle e_1, \ldots, e_{10}\rangle$, as its expected cost is $\mathrm{C}[\,\Theta_{1..10}\,] = 512 + \frac{1}{2}\left[256 + \frac{1}{2}\left[\ldots + \frac{1}{2}1\right]\right] \approx 683$; the best strategy is $\Theta_{10..1} = \langle e_{10}, \ldots, e_1\rangle$ with expected cost $\mathrm{C}[\,\Theta_{10..1}\,] = 1 + \frac{1}{2}\left[2 + \frac{1}{2}\left[\ldots + \frac{1}{2}512\right]\right] = 10$.

We then used several different PALO$_i$-algorithms, in each case beginning with the pessimal $\Theta_{1..10}$ and using $\delta = 0.05$ (hence, our learning algorithms should make no more than one mistake in 20 runs). We also tried four different ϵ values, ranging from $\epsilon = 400$ down to $\epsilon = 25$. In general, we used the PALO$_{\chi U}$ variant of the different algorithms, based on the opportunistic φ_U estimate; we also used PALO$_{1LU}$ and PALO$_{1P}$, discussed below.

To explain how we estimate the values of $\varphi(\Theta, \tau, q)$: We compute the upper bound $\varphi_U(\Theta, \tau, q)$, by assuming that every untried experiment in the sequence $\tau(\Theta)$ will fail. For example, imagine using $\Theta = \langle e_1, e_2, e_3, e_4\rangle$ on sample q, and observing that experiment e_1 failed and e_2 succeeded; hence $c(\Theta, q) = c(e_1) + c(e_2)$. Now let $\Theta' = \tau(\Theta) = \langle e_3, e_1, e_2, e_4\rangle$ be a neighbor of Θ. The value of $\varphi(\Theta, \tau, q) = c(\Theta, q) - c(\Theta, q')$[5] is either $[c(e_1) + c(e_2)] - [c(e_3) + c(e_1) + c(e_2)] = -c(e_3)$ if e_3 fails, or $[c(e_1) + c(e_2)] - [c(e_3)]$ if e_3 succeeds. The value is $\varphi_U(\Theta, \tau, q)$ is the larger of these values, $[c(e_1) + c(e_2)] - [c(e_3)]$, which corresponds to the situation where e_3 fails. The value of $\varphi_L(\Theta, \tau, q)$ is the smaller of these values, $-c(e_3)$, which corresponds to e_3's failure. We can trivially compute $\varphi_{LU}(\Theta, \tau, q)$ from these $\varphi_U(\Theta, \tau, q)$ and $\varphi_L(\Theta, \tau, q)$ values. Computing $\varphi_P(\Theta, \tau, q)$ is slightly more complicated. By the time we have encountered this $q = q_i$ sample, we will have processed a sequence of earlier samples $(q_1, q_2, \ldots, q_{i-1})$, each of which requires performing a set of experiments. We will maintain a set of statistics during these executions, recording the number of times each e_j experiment was performed $t(e_j)$ and succeeded $s(e_j)$. We will use these statistics to compute $\varphi_P(\Theta, \tau, q)$. Here, $\varphi_P(\Theta, \tau, q) = \hat{p}(e_3)[-c(e_3)] + (1 - \hat{p}(e_3))[c(e_1) + c(e_2) - c(e_3)] = (1 - \hat{p}(e_3))[c(e_1) + c(e_2)] - c(e_3)$, where $\hat{p}(e_3) = \frac{s(e_3)}{t(e_3)}$ is the ratio of the observed quantities (or $\frac{1}{2}$ if $t(e_3) = 0$).

Performance of Individual PALO$_i$ Systems: Before comparing the different PALO systems on average, we first present a quick description of how each PALO$_\chi$ worked. On one run, PALO$_{1U}$, when given the $\Theta_{1..10} = [e_1, e_2, \ldots, e_{10}]$ strategy (with expected cost $\mathrm{C}[\,\Theta_{1..10}\,] = 682.7$) and $\epsilon = 400$, first climbed to the strategy $\Theta_2 = [e_{10}, e_1, e_2, \ldots, e_9]$ (where $\mathrm{C}[\,\Theta_2\,] = 342.3$) after 12 samples, and then climbed to $\Theta_3 = [e_{10}, e_9, e_1, e_2, \ldots, e_8]$ (with cost 172.7) after another 67 samples (for a total of $12 + 67 = 79$ samples). PALO$_{1U}$ then required an additional 102 samples to decide that this strategy was a 400-local optimal and terminate its hill-climbing. (Of course, the underlying performance+learning system would then continue running the performance task, using the performance element based on this Θ_3 strategy.) As the cost of the globally optimal element $\Theta_{10..1}$ is 10, the Θ_3 element returned is necessarily a 400-local optimal; in fact, it is a $172.7 - 10 = 162.7$ local optimal. (Here, PALO$_{1U}$ would have been correct even if it had terminated one stage earlier and returned Θ_2, as Θ_2 is a 400-local optimal; in fact, it is a 333-local optimal. We will see this over-achieving behavior often.) In this particular run, all three decisions (*i.e.*, both the two hill-climbs, and the termination) were performed "early", based on the tests on Lines $\langle L3\rangle$ and $\langle L4\rangle$. In other situations, PALO$_{1U}$ would make corresponding decisions "late", based on the test on Line $\langle L5\rangle$.

When we used $\epsilon = 200$, PALO$_{1U}$ usually climbed three times, for example reaching the Θ_2 shown above after 19 samples, then Θ_3 after 56 additional samples, and finally terminating with the final

[5] As we are trying to *minimize* cost, we are computing the additive inverse of the $\varphi(\Theta, \tau, q)$ term defined above.

$\Theta_4 = [e_{10}, e_9, e_8, e_1, e_2, \ldots, e_7]$ (whose cost is 88.3) after 245 additional samples. It then examined 494 more samples before terminating "late" (*i.e.*, based on Line $\langle L5 \rangle$'s test).

The comparable numbers for $\epsilon = 100$ are 20 samples to climb to Θ_2, then 69 additional samples to climb to Θ_3, 287 more samples to climb to Θ_4, and finally 1249 more samples to climb to $\Theta_5 = [e_{10}, e_9, e_8, e_7, e_1, e_2, \ldots, e_6]$ (whose score is 46.7), followed by 2325 samples to know that Θ_5 is good enough. For $\epsilon = 50$, PALO$_{1U}$ performed 5 climbs, through Θ_2 (18 samples), Θ_3 (109 more samples), Θ_4 (309 more samples) and Θ_5 (1419 more samples), before reaching $\Theta_6 = [e_{10}, e_9, e_8, e_7, e_6, e_1, \ldots, e_5]$ (cost 26.3) after 5229 more samples, and terminating after 10184 more samples.

Four quick comments before discussing the other PALO$_\chi$ systems: First, different runs for the same ϵ value used different numbers of samples, and often involved different strategies, both at intermediate points and in conclusion. In fact, some runs terminated after fewer climbs, and others, after additional climbs. Second, while essentially all of these climbs occurred "early", based on the test on Line $\langle L3 \rangle$ (rather than the test on Line $\langle L5 \rangle$), the majority of the termination tests were based on (the failure of) Line $\langle L5 \rangle$'s test, rather than Line $\langle L4 \rangle$. Third, notice PALO$_{1U}$ would have been correct, even if it had stopped one stage earlier in all four situations, as Θ_2 is a 400-local optimal, Θ_3 is a 200-local optimal, Θ_4 is a 100-local optimal and Θ_5 is a 50-local optimal. Fourth, notice that PALO$_{1U}$ is almost always "perfect climber", in that each of its climbs is, in fact, the best possible climb; *i.e.*, it is the best element in the neighborhood. (For $\epsilon = 50$, occasionally climbed to second-best neighbor.)

PALO$_{0U}$ had essentially the same behavior as PALO$_{1U}$, in that it climbs through the same intermediate strategies before reaching and returning the same strategies. It usually required more samples, though, as it was not able to stop "early". For $\epsilon = 400$, PALO$_{0U}$ reached Θ_2 after 103 samples, then Θ_3 after 121 more samples, and terminating after seeing an additional 131 samples. For $\epsilon = 200$, PALO$_{0U}$ required 412, 485 and 527 samples to reach Θ_2, Θ_3 and Θ_4 respectively, and an additional 557 samples before terminating. And so on for the other ϵ values. (These results are summarized below.)

Notice that each of the PALO-algorithms presented so far requires a different number of samples for each climb; hence, each would require one number of samples to climb from Θ_1 to Θ_2, and then a different number of additional samples to climb from this Θ_2 to the new Θ_3, and so forth. By contrast, PALO$_{2U}$ requires a constant number for each climb, which depends on the ϵ value.[6] For $\epsilon = 400$, PALO$_{2U}$ would process a given Θ_j for 127 samples, before either climbing to a new Θ_{j+1} or terminating. Hence it would reach Θ_3 after $127 \times 2 = 254$ samples, and terminate after 127 more samples. For $\epsilon = 200$ (resp., 100, 50), that constant was 544 (resp., 2323, 9875).

Notice that PALO$_{0U}$ and PALO$_{2U}$ were also almost "perfect climbers"; their only differences were in terms of the number of samples observed before climbing, and the total number of climbs. That is not true for the other PALO$_i$ systems, discussed below.

In one run using $\epsilon = 400$, the PALO$_{1P}$ system climbed from $\Theta_{1..10}$ to $\Theta_7 = \langle e_3, e_1, e_2, e_4, \ldots, e_{10} \rangle$ (cost 458.7) after 34 samples, and then on to $\Theta_8 = \langle e_6, e_3, e_1, e_2, e_4, \ldots, e_{10} \rangle$ (cost 245.2) after a total of 75 samples, before terminating (late) after a total of 215 samples. On other runs, PALO$_{1P}$ climbed first to $\langle e_6, e_1, \ldots, e5, e7, \ldots, e_{10} \rangle$ (cost 357.2), then stopped; or first to $\langle e_5, e_1, \ldots, e_4, e_6, \ldots, e_{10} \rangle$ (cost 372.7) then to $\langle e_6, e_5, e_1, \ldots, e_4, e_7, \ldots, e_{10} \rangle$ (cost 202.2) and finally to $\langle e_7, e_6, e_5, e_1, \ldots, e_4, e_8, e_9, e_{10} \rangle$ (cost 109.0); each of the other runs used yet other strategies. The runs using $\epsilon = 200$ were equally varied, as with $\epsilon = 100$ and $\epsilon = 50$. While PALO$_{1P}$ was not a perfect climber, our empirical data confirmed that it was almost always correct: in all 4×25 runs,

[6] In our current situation, the difference between the values defined in Equations 4 and 5 was insignificant.

	Values of ϵ															
	400				200				100				50			
PALO_{1U}	2 3	84.1 229.5	230.4	159.2	3 4	406.0 1,065.7	1,045.0	83.4	4	1,819.1	4,256.6	46.7	5	8,491.4	18,635.4	26.3
PALO_{1LU}	1 2	59.3 182	194.2	335.9	2	361.0	924.0	177.4	3	1,642.2	4,018.2	90.7	4	7,586.8	17,465.8	47.2
PALO_{1P}	1 2	26.2 117.1	203.8	236.4	2–4	555.0	1,011.5	120.1	3–6	2,190.9	4,198.4	70.4	4–7	10,362.3	18,690.6	37.5
PALO_{0U}	2 3	224 355	382.8	156.1	3 4	1,424 1,981	2,027.5	85.6	4	7,932	10,256	47.2	5	41,033	50,637	27.0
PALO_{2U}	2 3	254 381	391.2	166.2	3 4	1,632 2,176	2,219.5	85.4	4 5	9,292 11,615	11,707.9	46.3	5	4,9375	59,250	27.0
PALO_{1NU}	4–66	1296.4	1,403.2	191.6	8–206	7,704	7,832.1	215.2	7–195	8,800.3	8,870.2	225.2	7–277	8,474.6	8,479.6	218.2

Table 2: Results of different PALO_i-Algorithms

it never once climbed to an inferior element.

The PALO_{1N} system was not so effective; it fairly often climbed to inferior neighbors. On one $\epsilon = 400$ run, it climbed first to $\langle e_1, \ldots, e_8, e_{10}, e_9 \rangle$ (cost 682.7) after 10 samples, then to $\langle e_2, e_3, e_4, e_1, e_5, \ldots, e_8, e_{10}, e_9 \rangle$ (cost 402.7) after 5 more samples, then $\Theta_9 = \langle e_2, e_3, e_4, e_1, e_5, e_7, e_8, e_6, e_{10}, e_9 \rangle$ (cost 402.4) on sample 68 and onto $\Theta_{10} = \langle e_2, e_3, e_4, e_1, e_5, e_7, e_6, e_{10}, e_8, e_9 \rangle$ (cost 402.5) on sample 110. Notice this last transition is a mistake, as $C[\,\Theta_{10}\,] > C[\,\Theta_9\,]$. PALO_{1N} climbed 7 more times before terminating; 3 of those transitions were mistakes, climbing to elements that represented *increases* in cost of 43, 9.3 and 13.8.

Summary of Average Performances: This subsection extends the earlier anecdotes to present a summary of how the different PALO_i systems performed, using a set of 25 different runs for each of ϵ values mentioned above. (Each run involves running the system until the PALO_i-component reaches a local optimal and terminates itself.) The relevant statistics appear in Table 2: To interpret an entry of the form "$\mathtt{n}_1\ \ \mathtt{n}_2\ \ \mathtt{n}_3\ \ \underline{\mathtt{n}_4}$" associated with running the particular system PALO_i using epsilon value ϵ: $\mathtt{n}_1$ is number of times PALO_i climbed on each run; $\mathtt{n}_2$ is the average number of samples required to reach final performance element (this is the metric discussed above); $\mathtt{n}_3$ is the average number of samples required before the PALO_i-algorithm terminated; and $\mathtt{n}_4$ is the average expected-cost of resulting system.[7] An entry is displayed as an integer only if all 25 runs produced the same integral value. We use these entries if PALO_i climbed the same number of times on each run, for a given ϵ value. For certain ϵ values, however, PALO_i climbed a different number of times on different runs; here, we used the slightly more complicated "$\begin{Bmatrix} \mathtt{n}_{1A} & \mathtt{n}_{2A} \\ \mathtt{n}_{1B} & \mathtt{n}_{2B} \end{Bmatrix}\ \ \mathtt{n}_3\ \ \underline{\mathtt{n}_4}$" entries: The meanings of $\mathtt{n}_3$ and $\mathtt{n}_4$ are unchanged: they remain the average number of samples until termination and the average expected cost of the resulting system, respectively. Here, however, we separate the different average number of samples required to find the final elements. To illustrate using PALO_{1U} and $\epsilon = 400$: Of the runs when PALO_{1U} required exactly 2 hill-climbing steps, it used on average 84.1 samples before reaching this Θ_3. PALO_{1U} required exactly 3 hill-climbing steps on the remaining runs; here, PALO_{1U} required an average of 230.4 samples. (We split up these values to keep these incomparable values distinct. Btw, PALO_{1U} required 2 steps on 21 runs, and 3 on the remaining 4 runs.) Finally, certain PALO_i systems performed many different numbers of climbs on

[7]Of course, this $\mathtt{n}_4$ value is based on our knowledge of the probabilities, which is not available to the PALO_i learning systems.

different runs, associated with the same ϵ values. Here, $\mathtt{n}_1$ is simply the upper and lower bounds on the number of climbs, and $\mathtt{n}_2$ is overall average.

There are several obvious observations one can make: Consider first just the three PALO$_{\chi U}$ systems: PALO$_{1U}$, PALO$_{0U}$ and PALO$_{2U}$. Notice each of these PALO$_i$ system always produced (at least) an ϵ-local optimal in all $3 \times 4 \times 25$ runs. In fact, as mentioned above, each of these system climbed at least one extra time in many situations to produce answers that are better than required. (The PALO$_{1LU}$, PALO$_{1P}$ and PALO$_{1NU}$ systems were not such overachievers.)

The final $\mathtt{n}_4$ entry indicates the quality of the performance element that each produces; notice that all three PALO$_{\chi U}$ system produce comparable final element, for all four ϵ values, as they all undertake approximately the same number of climbs (corresponding to the $\mathtt{n}_{2\chi}$ entries). The $\mathtt{n}_{1\chi}$ entries show that PALO$_{1U}$ consistently required the fewest number of samples to produce this final element, as anticipated. While it is not surprising that PALO$_{0U}$ did poorly in this measure, we were slightly surprised that it also did poorly with respect to the $\mathtt{n}_{3\chi}$ entries, which measure the total number of samples before termination. (This is no doubt due to the additional cost of PALO$_{0U}$'s earliest steps.) PALO$_{2U}$ was uniformly dominated, over all of these situations. Of course, this is not always the case: PALO$_{2U}$ will be preferable in contexts which more require more climbing. Similarly, it is easy to find situations in which any of the other PALO$_i$ systems will be preferable.

Now to discuss the other PALO$_i$ systems. While PALO$_{1LU}$ generally required fewer samples than any PALO$_{\chi U}$ before reaching its final performance element, PALO$_{1LU}$'s final element tends to be worse than the elements produced by the PALO$_{\chi U}$ systems, as these other systems generally climb one time *more* that PALO$_{1LU}$. Notice, however, that these returned elements are still ϵ-local optimal. The behavior of the PALO$_{1P}$ system is more complicated to describe. It was similar to PALO$_{1LU}$'s behaviour, as PALO$_{1P}$ usually climbed sooner than PALO$_{1U}$, but it did not climb as many times. The behaviour was much more variable, though: Each of the PALO$_\chi$ systems discussed so far would either climb n times or $n+1$ times, over each trial of 25 runs for a given value of ϵ. By contrast, PALO$_{1P}$ climbed either between 2 and 4 times for $\epsilon = 200$, between 3 and 7 times for $\epsilon = 100$, and between 4 and 8 times for $\epsilon = 50$. (As shown, it climbed either 1 or 2 times for $\epsilon = 400$.) While the cost of the final element was always an ϵ-local optimal (in each of the 4×25 examples), the actual costs were much higher than in the PALO$_{\chi U}$ cases. (In fact, several of the final elements just barely qualified as an ϵ-local optimal.) Also, as mentioned above, PALO$_{1P}$ never climbed in error; *i.e.*, each time PALO$_{1P}$ climbed from Θ_i to Θ_{i+1}, it was always true that $C[\,\Theta_i\,] < C[\,\Theta_{i+1}\,]$.

The behaviors of the PALO$_{1NU}$ system is even more complicated to describe. The number of times it climbed is much more varied: Table 3 shows the distribution for the 25 $\epsilon = 400$ runs. All told, PALO$_{1NU}$ climbed a total of 475 times in those 25 runs; of those, 148 climbs were mistakes! In each case, however, PALO$_{1NU}$ did return a 400-local optimal. Similarly 833 of PALO$_{1N}$'s 1992 climbs for $\epsilon = 200$ were mistakes, and in 10 runs, PALO$_{1NU}$ returned a value that was *not* a 200-local optimal. The comparable numbers for $\epsilon = 100$ were 823 climbing mistakes out of 2095 climbs with 20 premature terminations; and for $\epsilon = 50$: 738 bad climbs in 1845, and 24 premature terminations.

In general, PALO$_{1NU}$ took relatively few samples before climbing; unfortunately, it would often climb to inferior elements, and often stopped too early. Moreover, even when PALO$_{1NU}$ made no mistakes, it still often performed worse than the other systems as it would often climb to neighbors that were only slight improvements, rather than to the optimal neighbor. We did find, however, that PALO$_{1NU}$ was often superior to PALO$_{1U}$ for *small problems*; e.g., when dealing with a performance element the corresponds to only the first three experiments $\langle e_1, e_2, e_3 \rangle$. Here, PALO$_{1NU}$ would climb quickly, and almost always correctly.

(We also implemented the PALO$_{1L}$ system, and tested it in this same context. Unfortunately, our estimating techniques appeared to be too sloppy for PALO$_{1L}$, as it failed to climb in any of the

n_c	4	5	7	8	10	11	15	17	19	21	22	23	27	32	33	48	66
n_t	1	1	2	2	2	2	2	3	1	1	2	1	1	1	1	1	1

Table 3: The number of times that PALO$_{1NU}$ climbed n_c times, is n_t

situations discussed! We did, of course, test it in other contexts in which it would work effectively.)

All told, this evidence tends to corroborate our assumption that the PALO$_{1U}$ system is the one that should be used, as the element it returns is comparable to the elements returned by the others, but PALO$_{1U}$ requires fewer samples to find this answer.

Comparison to PAO: These PALO systems have a similar objective to the PAO system defined in [GO91]. PAO also uses observed samples to estimate the unknown distribution, and then uses these estimates to identify a good performance element. PAO differs by producing a performance element that is close to the *global* optimum, with high probability. (Recall that a PALO-algorithm can only guarantee that the results are essentially a *local* optimal.) There is a price for this increased ability: PAO only applies when the underlying performance task involves processing a set of "experiments", which must be independent; it also requires an algorithm (which we hope is efficient) for producing the optimal element, once given estimates of the distribution. Notice that the previous above $\Theta_{1..10}$ task satisfies these constraints. We can therefore apply [GO91]'s algorithm, which basically requires first sampling each experiment at least

$$M \quad = \quad \left\lceil 2\left(\frac{n\,C}{\epsilon}\right)^2 \ln\frac{2n}{\delta}\right\rceil$$

times, where n is the number of experiments (here, $n = 10$) and C is the total cost of traversing this entire graph, which corresponds to the λ defined above (here, $C = 1023$). Using the above $\delta = 0.05$, the comparable values are

ϵ	400	200	100	50
M	7,838	31,352	125,405	501,620

Now observe that, in every case, the PALO$_i$-algorithm produced a performance element whose cost is within the prescribed ϵ of the cost of the *global* optimal (which we know is 10, based on $\Theta_{10..1}$). Hence, in our particular situation, the PALO-algorithms strictly dominate PAO, as they are not just more versatile, but will produce an equally good answer, with fewer trials required.

A Proofs

Theorem 1 PALO$_1$ *qualifies as a* PALO*-algorithm.*

Proof: To prove Parts 1 and 2 of Definition 1, consider first a single stage of the PALO$_1$ algorithm, when it is dealing with Θ_j. Notice there are four types of mistakes that PALO$_1$ could make:

A. After seeing i samples (for some $i = 1..L_j - 1$), PALO$_1$ could climb from Θ_j to some $\Theta' = \tau(\Theta_j)$ as Θ' appears to be better than Θ_j (based on Line $\langle L3\rangle$ test), but in reality, Θ' is not better; or

B. After seeing i samples (for some $i = 1..L_j - 1$), PALO_1 could terminate as no $\Theta' = \tau(\Theta_j)$ appears to be more than ϵ better than Θ_j (based on Line $\langle L4 \rangle$ test), but there is some Θ' that is better; or

C. After seeing all L_j samples, PALO_1 could climb from Θ_j to some $\Theta' = \tau(\Theta_j)$ as Θ' appears to be better than Θ_j (based on Line $\langle L5 \rangle$ test), but in reality, Θ' is not better; or

D. After seeing all L_j samples, PALO_1 could terminate as no $\Theta' = \tau(\Theta_j)$ appears to be more than ϵ better than Θ_j, but there is some Θ' that is better.

Using $\epsilon_i(\Theta') = \Lambda(\Theta_j, \Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ as the "barrier" used to decide whether to climb to the neighboring $\Theta' = \tau(\Theta_j)$ after seeing i samples, the respective probabilities of these events are

$$\begin{aligned}
a_j^i &= Pr\left[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]\colon\ \tfrac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \geq \epsilon_i(\Theta') \text{ and } \mathrm{C}[\,\Theta'\,] < \mathrm{C}[\,\Theta_j\,]\,\right]\\
b_j^i &= Pr\left[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]\colon\ \tfrac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] < \epsilon - \epsilon_i(\Theta') \text{ and } \mathrm{C}[\,\Theta'\,] > \mathrm{C}[\,\Theta_j\,] + \epsilon\,\right]\\
c_j &= Pr\left[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]\colon\ \tfrac{1}{L_j}\Delta[\Theta_j,\,\Theta',\,L_j] \geq \tfrac{\epsilon}{2} \text{ and } \mathrm{C}[\,\Theta'\,] < \mathrm{C}[\,\Theta_j\,]\,\right]\\
d_j &= Pr\left[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]\colon\ \tfrac{1}{L_j}\Delta[\Theta_j,\,\Theta',\,L_j] < \tfrac{\epsilon}{2} \text{ and } \mathrm{C}[\,\Theta_j\,] > \mathrm{C}[\,\Theta_j\,] + \epsilon\,\right]
\end{aligned}$$

Of course, each existential within a $Pr[\cdot]$ above is really a finite disjunction, including only elements of $|\mathcal{T}[\Theta_j]|$. Moreover, a_j^i is only considering the subset of Θ's for which $\mathrm{C}[\,\Theta_j)\,] < \mathrm{C}[\,\Theta_j\,]$; hence using $\mathcal{T}^< \;=\; \{\,\Theta' \in \mathcal{T}[\Theta_j]\,|\,\mathrm{C}[\,\Theta'\,] < \mathrm{C}[\,\Theta_j\,]\,\}$,

$$a_j^i = Pr\left[\bigvee_{\Theta' \in \mathcal{T}^<} \frac{1}{i}\Delta[\Theta_j, \Theta', i] \;\geq\; \epsilon_i(\Theta')\right]$$

and using $\mathcal{T}^> \;=\; \{\,\Theta' \in \mathcal{T}[\Theta_j]\,|\,\mathrm{C}[\,\Theta'\,] > \mathrm{C}[\,\Theta_j\,] + \epsilon\,\}$,

$$b_j^i = Pr\left[\bigvee_{\Theta' \in \mathcal{T}^>} \frac{1}{i}\Delta[\Theta_j, \Theta', i] \;\leq\; \epsilon - \epsilon_i(\Theta')\right]$$

and so on.

Now observe that

$$\begin{aligned}
a_j^i &\leq \sum_{\Theta' \in \mathcal{T}^<} Pr\left[\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;\geq\; \epsilon_i(\Theta')\right]\\
&\leq \sum_{\Theta' \in \mathcal{T}^<} Pr\left[\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;\geq\; (\mathrm{C}[\,\Theta'\,] - \mathrm{C}[\,\Theta_j\,]) \,+\, \epsilon_i(\Theta')\,\right] && (7)\\
&\leq \sum_{\Theta' \in \mathcal{T}^<} e^{-2i\left(\frac{\epsilon_i(\Theta')}{\Lambda(\Theta_j,\Theta')}\right)^2} && (8)\\
&\leq |\mathcal{T}^<|\; exp\left\{-2i\left(\frac{\Lambda(\Theta_j,\Theta')\sqrt{\frac{1}{2i}\,\ln\left(\frac{2(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}}{\Lambda(\Theta_j,\Theta')}\right)^2\right\}\\
&= |\mathcal{T}^<|\,\frac{\delta_j}{2\,(L_j-1)\,|\mathcal{T}[\Theta_j]|}
\end{aligned}$$

Line 7 follows from the observations that $\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;\geq\; \epsilon_i(\Theta')$ and $\mathrm{C}[\,\tau(\Theta_j)\,] - \mathrm{C}[\,\Theta_j\,] \;<\; 0$ implies $\frac{1}{i}\Delta[\Theta_j,\,\tau,\,i] \;\geq\; (\mathrm{C}[\,\tau(\Theta_j)\,] - \mathrm{C}[\,\Theta_j\,]) + \epsilon_i(\Theta')$, and $A \Rightarrow B$ implies $Pr[A] \leq Pr[B]$, for any events A and B. Line 8 uses Hoeffding's inequality (which bounds the probability that the sample mean of a set of iid random variables will be far from the population mean [Hoe63]) using the facts that each $\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i]$ is the empirical average of a set of i independent, identically-distributed random values $\{c(\Theta',\,q_\ell) - c(\Theta_j,\,q_\ell)\}_{\ell=1}^{i}$, whose (common) mean is $\mathrm{C}[\,\Theta'\,] - \mathrm{C}[\,\Theta_j\,]$, and whose range of possible values is $\Lambda(\Theta_j, \Theta')$.

Similarly,

$$
\begin{aligned}
b_j^i &\leq \sum_{\Theta' \in \mathcal{T}^>} Pr\left[\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;<\; \epsilon - \epsilon_i(\Theta')\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^>} Pr\left[\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;\leq\; (\mathrm{C}[\,\Theta'\,] - \mathrm{C}[\,\Theta_j\,]) \;-\; \epsilon_i(\Theta')\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^>} e^{-2i\left(\frac{-\epsilon_i(\Theta')}{\Lambda(\Theta_j,\Theta')}\right)^2} \\
&\leq |\mathcal{T}^>|\,\frac{\delta_j}{2\,(L_j - 1)\,|\mathcal{T}[\Theta_j]|}
\end{aligned}
$$

In a similar manner, we can bound

$$
\begin{aligned}
c_j &\leq \sum_{\Theta' \in \mathcal{T}^<} Pr\left[\frac{1}{L_j}\Delta[\Theta_j,\,\Theta',\,L_j] \;\geq\; \frac{\epsilon}{2}\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^<} e^{-2L_j\left(\frac{\epsilon/2}{\Lambda(\Theta_j,\Theta')}\right)^2} \\
&\leq |\mathcal{T}^<|\,exp\left\{-2\left(2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{2\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)\left(\frac{\epsilon}{2\,\Lambda[\Theta_j]}\right)^2\right\} \qquad (9) \\
&= |\mathcal{T}^<|\frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|}
\end{aligned}
$$

(Line 9 uses the fact that $\Lambda(\Theta_j, \Theta') \leq \Lambda[\Theta_j]$); and likewise

$$
d_j \;\leq\; \sum_{\Theta' \in \mathcal{T}^>} Pr\left[\frac{1}{L_j}\Delta[\Theta_j,\,\Theta',\,L_j] \;<\; \frac{\epsilon}{2}\right] \;\leq\; \sum_{\Theta' \in \mathcal{T}^>} e^{-2L_j\left(\frac{\epsilon/2}{\Lambda(\Theta_j,\Theta')}\right)^2} \;\leq\; |\mathcal{T}^>|\frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|}
$$

Hence, the probability of making any type of mistake at the j^{th} stage is bounded by

$$
\left[\sum_{i=1}^{L_j-1} a_j^i + b_j^i\right] + c_j + d_j \;\leq\; (L_j - 1)\frac{\delta_j}{2\,(L_j - 1)\,|\mathcal{T}[\Theta_j]|}\left(|\mathcal{T}^<| + |\mathcal{T}^>|\right) + \frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|}\left(|\mathcal{T}^<| + |\mathcal{T}^>|\right) \;\leq\; \delta_j
$$

as $\mathcal{T}^<$ and $\mathcal{T}^>$ are disjoint subsets of $\mathcal{T}[\Theta_j]$, and so $|\mathcal{T}^<| + |\mathcal{T}^>| \leq |\mathcal{T}[\Theta_j]|$.

The probability that PALO$_1$ will make a mistake (of any kind) on any step is at most

$$
\leq \sum_{j=1}^{\infty} \delta_j \;=\; \sum_{j=1}^{\infty} \frac{\delta\;6}{j^2\pi^2} \;=\; \frac{\delta\;6}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{j^2} \;=\; \frac{\delta\;6}{\pi^2}\frac{\pi^2}{6} \;=\; \delta
$$

as desired.

To deal with PALO_1's efficiency: Notice it can stay at any Θ_j performance element for at most L_j samples, a quantity that is clearly polynomial in $|\mathcal{T}[\Theta_j]|$, $\Lambda[\Theta_j] \leq 2\lambda(c)$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

To show that PALO_1 will terminate with probability 1, notice that the only way that PALO_1 can fail to terminate is if it cycles infinitely often: each time thinking first that some Θ_i is strictly better than Θ_j and so switching to it, and later, thinking that Θ_i is better, switching back. Of course, one of these inequalities is necessarily false; and the probability that PALO_1 will make such a mistake is $\frac{\delta_i}{3} < \frac{\delta}{3}$. The proability that PALO_1 will make this type of mistake an infinite number of times is therefore (at most) $\lim_{m\to\infty} \prod_{i=1}^{m} \frac{\delta}{3} = 0$. □ (Theorem 1)

Theorem 2 PALO_{1N} *qualifies as a* PALO*-algorithm, provided the samples it uses are normally distributed.*

Proof: As $z(1-p) \leq \ln(1/p)$ for the relevant values of p PALO_{1N}'s sample complexity, per climb, is clearly polynomial. Now to bound the probability that PALO_{1N} will either climb to an inferior element, or will terminate when there is a much better element. Borrowing from the above proof, we need only show that ...

1. ...after $L_j^{(N)}$ samples,

$$Pr\left[\left|\frac{1}{L_j^{(N)}}\Delta[\Theta_j,\,\Theta',\,L_j^{(N)}-1] \;-\; (\mathrm{C}[\,\tau(\Theta_j)\,] - \mathrm{C}[\,\Theta_j\,])\right| \;\leq\; \frac{\epsilon}{2}\right] \quad\geq\quad 1 - \frac{\delta_j}{|\mathcal{T}[\Theta_j]|}$$

as this means that observing $\Delta[\Theta_j,\Theta',L_j^{(N)}-1] > \frac{\epsilon}{2}$ (resp., $\Delta[\Theta_j,\Theta',L_j^{(N)}-1] \leq \frac{\epsilon}{2}$) is sufficient to conclude that $\mathrm{C}[\,\Theta'\,] > \mathrm{C}[\,\Theta_j\,]$ (resp., $\mathrm{C}[\,\Theta'\,] < \mathrm{C}[\,\Theta_j\,] - \epsilon$) with probability at least $1 - \frac{\delta_j}{2|\mathcal{T}[\Theta_j]|}$.[8]

As the range of each $X_\ell = c(\tau(\Theta_j),\,q_\ell) - c(\Theta_j,\,q_\ell)$ variable is $\Lambda[\Theta_j]$, X_ℓ's variance is necessarily less than $\frac{1}{4}\Lambda[\Theta_j]^2$. From [BD77, p158], after $\frac{z^2(1-\frac{\alpha}{2})\,\sigma_1^2}{\beta^2}$ samples, we can be at least $1-\alpha$ confident that our empirical average $\bar{X}$ will be within β of the population mean μ; here we use $\alpha = \frac{\delta_j}{|\mathcal{T}[\Theta_j]|}$ and $\beta = \frac{\epsilon}{2}$.

2. ...using the $\epsilon_i^{(N)}$ values defined in Equation 6,

$$Pr\left[\left|\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i] \;-\; (\mathrm{C}[\,\Theta'\,] - \mathrm{C}[\,\Theta_j\,])\right| \leq \epsilon_i^{(N)}\right] \quad\geq\quad 1 - \frac{\delta_j}{(L_j^{(N)}-1)|\mathcal{T}[\Theta_j]|}$$

as this means that observing $\frac{1}{i}\Delta[\Theta_j,\Theta',i] > \epsilon_i^{(N)}$ (resp., $\frac{1}{i}\Delta[\Theta_j,\Theta',i] \leq \epsilon - \epsilon_i^{(N)}$) is sufficient to conclude that $\mathrm{C}[\,\Theta'\,] > \mathrm{C}[\,\Theta_j\,]$ (resp., $\mathrm{C}[\,\Theta'\,] < \mathrm{C}[\,\Theta_j\,] - \epsilon$) with probability at least $1 - \frac{\delta_j}{2(L_j^{(N)}-1)|\mathcal{T}[\Theta_j]|}$.

This follows from the fact, stated in [BD77, p157–58], that $\frac{\sqrt{n}(\bar{X}-\mu)}{s}$ has a $\mathcal{T}_{n-1}$ distribution (student's t, with $n-1$ degrees of freedom), and so

$$Pr\left[\,\frac{1}{n}\Delta[\Theta_j,\,\Theta',\,n] \;>\; \mu + \frac{s(n)\,t_{n-1}(1-\alpha)}{\sqrt{n}}\right] \;=\; Pr\left[\,\frac{1}{n}\Delta[\Theta_j,\,\Theta',\,n] \;<\; \mu - \frac{s(n)\,t_{n-1}(1-\alpha)}{\sqrt{n}}\right] \;=\; \alpha$$

□ (Theorem 2)

[8] Of course, we are assuming that $Pr[\hat{X} > \mu+\lambda] = Pr[\hat{X} < \mu-\lambda] = \frac{1}{2}Pr[|\mu-\hat{X}| > \lambda]$, which is true in this case.

Further Extensions: We can generalize from this example to define yet other PALO variants that apply to other classes of distribution. We need only define

1. L'_j such that

$$Pr\left[\left|\frac{1}{L'_j}\Delta[\Theta_j,\,\Theta',\,L'_j-1]\;-\;(\mathrm{C}[\,\Theta'\,]-\mathrm{C}[\,\Theta_j\,])\right|\leq\frac{\epsilon}{2}\right]\quad\geq\quad 1-\frac{\delta_j}{|\mathcal{T}[\Theta_j]|}$$

2. ϵ'_i such that

$$Pr\left[\left|\frac{1}{i}\Delta[\Theta_j,\,\Theta',\,i]\;-\;(\mathrm{C}[\,\Theta'\,]-\mathrm{C}[\,\Theta_j\,])\right|\leq\epsilon'_i\right]\quad\geq\quad 1-\frac{\delta_j}{(L'_j-1)|\mathcal{T}[\Theta_j]|}$$

in each case, where $\Theta'\in\mathcal{T}[\Theta_j]$.

References

[BD77] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics.* Holden-Day, Inc., Oakland, 1977.

[CGS91] William Cohen, Russell Greiner, and Dale Schuurmans. Probabilistic hill climbing. In *Proceedings of CLNL-91*, Berkeley, September 1991.

[GD92] Jonathan Gratch and Gerald Dejong. COMPOSER: A probabilistic solution to the utility problem in speed-up learning. In *Proceedings of AAAI-92*, 1992.

[GJ92] Russell Greiner and Igor Jurišica. A statistical approach to solving the EBL utility problem. In *Proceedings of AAAI-92*, San Jose, 1992.

[GO91] Russell Greiner and Pekka Orponen. Probably approximately optimal derivation strategies. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of KR-91*, San Mateo, CA, April 1991. Morgan Kaufmann.

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

[Gre92] Russell Greiner. Probabilistic hill-climbing: Theory and applications. In *Proceedings of CSCSI-92*, Vancouver, June 1992.

[GS92] Russell Greiner and Dale Schuurmans. Learning useful horn approximations. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of KR-92*, San Mateo, CA, October 1992. Morgan Kaufmann.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[Ris83] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.