Appears in the
Workshop on Knowledge Compilation and Speedup Learning,
Amherst, June 1993.

Adaptive Derivation Processes

Russell Greiner*
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540-6632
greiner@learning.scr.siemens.com

1 Introduction

Many reasoning systems must reach conclusions based on stored information; we can often model this as deriving logical conclusions from a given knowledge base of facts. We of course prefer derivation systems that draw all and only the correct conclusions, and that reach these conclusions as quickly as possible. Unfortunately, a sound and complete derivation process can be intractable, if not undecidable, in the worse case [LB85]. This position paper discusses the general challenge of producing an derivation process that is as effective as possible, and argues for using a (cautious) adaptive derivation process here. Section 2 first provides a trivial example to explain the ideas and motivate our "adaptive process" approach; Section 3 then explains adaptive systems in general, and focuses on one implementation of this idea, PALO. Section 4 concludes by suggesting some of the extensions and applications relevant to knowledge compilation and presenting some relevant future work.

2 Framework (and Example)

Extending [Lev84], we model the reasoner as a parameterized derivation process $\textsc{Ask}_\alpha(\cdot,\cdot)$, where $\textsc{Ask}_\alpha(KB, q)$ returns the answers to the query q based on the knowledge base KB.[1] The α subscript is used to denote the various parameters of this derivation process, which can include a derivation strategy that specifies the order in which to consider the clauses in the knowledge base; *cf.*, [Smi89, GO91]. As a simple example, consider the knowledge base

$$KB_c = \left\{ \begin{array}{l} \texttt{child(X) :- boy(X).} \\ \texttt{child(X) :- girl(X).} \\ \texttt{boy(abe).\ \ boy(bob).\ \ ...} \\ \texttt{girl(ann).\ \ girl(carla).\ \ ...} \end{array} \right\}$$

and assume the $\alpha = \langle bg \rangle$ parameter-setting specifies that each (sub)goal is matched against the KB's clauses in a top-to-bottom order (*á la* PROLOG [CM81], assuming "top" is oldest). Hence, $\textsc{Ask}_{\langle bg \rangle}(KB_c, \texttt{child(bo)})$ first reduces `child(bo)` to `boy(bo)` and checks if this proposition is in KB_c's set of atomic facts; if not, it then uses the second rule to reduce `child(bo)` to `girl(bo)` and checks if this proposition is in KB_c. By contrast, the $\alpha = \langle gb \rangle$ parameter-setting uses a different ordering of the rules: $\textsc{Ask}_{\langle gb \rangle}(KB_c, \texttt{child(bo)})$ first searches for the `girl(bo)` proposition, and then (if necessary) for `boy(bo)`. Notice this $\textsc{Ask}_{\langle gb \rangle}(KB_c, \texttt{child(bo)})$ process will take *less* time than $\textsc{Ask}_{\langle bg \rangle}(KB_c, \texttt{child(bo)})$ if `girl(bo)` is in KB_c but will take more time if $\texttt{boy(bo)} \in KB_c$.[2]

When should we use the $\textsc{Ask}_{\langle bg \rangle}(KB_c, \cdot)$ process, rather than $\textsc{Ask}_{\langle gb \rangle}(KB_c, \cdot)$? As both systems produce the same set of answers to each query, the only difference can be computational cost. If we knew the distribution of queries — which here corresponds to "the respective probabilities of $\texttt{boy}(\kappa_j) \in KB_c$ and $\texttt{girl}(\kappa_j) \in KB_c$, over the set of $\texttt{child}(\kappa_j)$ queries" — we could then compute the *average time* required to answer $\textsc{Ask}_{\alpha_i}(KB_c, \texttt{child}(\kappa_j))$ queries for $\alpha_i \in \{\langle gb \rangle, \langle bg \rangle\}$, and then chose the α_i whose average cost is less.[3]

To scale up from this trivial example: In general,

*I gratefully acknowledge receiving helpful comments on this research from William Cohen and Dale Schuurmans.

[1] In the propositional case, an "ideal" $\texttt{Ask}_\alpha(KB, q)$ process will return "Yes" iff $KB \models q$ and "No" otherwise. In the predicate calculus case, such processes could return a set of binding lists for the free variables in q; etc.

[2] Here, we assume that `bo` is known to be either a boy xor a girl.

[3] This assumes we know (at least an approximation to) the cost model for such individual computations — *i.e.*, the cost of each rule-based reduction and database retrieval. We are also assuming that the appropriate quality measure is *expected* cost, as opposed to "best worst-case cost", or some other combining relation; see [GE91].

there can be a set of possible "performance elements" $\mathcal{S}_\Theta = \{\textsc{Ask}_{\alpha_i}(KB_i, \cdot)\}_i$, perhaps each formed from an initial $\textsc{Ask}_{\alpha_0}(KB_0, \cdot)$ by modifying the initial knowledge base KB_0 and/or the parameters α_0. (The $\textsc{Ask}_{\alpha_i}(KB_c, \cdot)$s in our example differ only in the order in which they considered the various rule-based reductions; notice they all use the same KB_c.) Our goal is to find the best of these $\mathcal{S}_\Theta$ elements.

To define this more precisely: We assume as given a set of performance elements $\mathcal{S}_\Theta = \{\Theta_k\}$ and a set of tasks (or problems or queries or ...) $\mathcal{Q} = \{q_j\}$ that will be drawn according to some stationary (but unknown) distribution $P\colon \mathcal{Q} \mapsto [0,1]$. There is also a cost function $c\colon \mathcal{S}_\Theta \times \mathcal{Q} \mapsto \Re$, which measures the time Θ requires to produce an answer to q. The quality measure for comparing different elements is their respective *expected cost*,

$$\mathrm{C}[\Theta] \;=\; E[\,c(\Theta, \mathbf{q})\,] \;=\; \sum_{q \in \mathcal{Q}} P[q] \times c(\Theta, q)$$

We are seeking the "optimal element" $\Theta_{opt} \in \mathcal{S}_\Theta$, which is the element whose expected cost is minimal:

$$\forall \Theta \in \mathcal{S}_\Theta,\; \mathrm{C}[\Theta_{opt}] \;\le\; \mathrm{C}[\Theta]\;.$$

There are two immediate issues: First, we need to know the distribution of queries $P[\cdot]$ to compute the expected cost of any given $\Theta_i = \textsc{Ask}_{\alpha_i}(KB_i, \cdot)$ system, and hence to determine which is optimal. Unfortunately, this distribution information is usually not available *a priori*. Second, even given the distribution information, the task of computing the globally optimal system is often intractable; *cf.*, [Gre91].

3 Adaptive Derivation Process

One way around these obstacles is to build an *adaptive derivation process*, which begins with one performance element $\Theta_1 = \textsc{Ask}_{\alpha_1}(KB_1, \cdot)$ and slowly "adapts" — as Θ_1 is solving the queries posed by the user, a "learning element" [BMSJ78] monitors the performance, and can eventually replace Θ_1 with another performance element Θ_2 that is better for this environment. (In the above example, if the learner observed that all of the queries dealt with girls, it would replace the $\Theta_1 = \textsc{Ask}_{\langle bg\rangle}(KB_c, \cdot)$ performance element with $\Theta_2 = \textsc{Ask}_{\langle gb\rangle}(KB_c, \cdot)$.) The learner could then observe how well this new Θ_2 works over another set of queries, and possibly replace it with a third, superior performance element Θ_3, and so on; in essence hill-climbing in the space of performance elements. Eventually the learner may reach a locally optimal element, and terminate. Hence, the learner is using its observations (of the user's queries) to obtain an estimate of the distribution $P[\cdot]$ of the queries, and then using this information to hill-climb to successive (apparently better) performance elements.

Algorithm PALO(Θ_1, ϵ, δ)
  For $k = 1..\infty$ **do**
    Let $\mathcal{T}[\Theta_k] \;\leftarrow\; \{\tau(\Theta_k) \mid \tau \in \mathcal{T}\}$
      $L_k \;\leftarrow\; \left\lceil 2\left(\frac{\lambda}{\epsilon}\right)^2 \ln \frac{k^2 |\mathcal{T}[\Theta_k]|\, \pi^2}{3\,\delta} \right\rceil$
    Draw L_k sample queries from $P[\cdot]$ distribution,
      $S_k = \{q_1, \ldots, q_{L_k}\}$
    ForEach $\Theta' \in \mathcal{T}[\Theta_k]$ **do**
      Let $\Delta[\Theta', \Theta_k] \;\leftarrow\; \frac{1}{L_k}\sum_{i=1}^{L_k} c(\Theta_k, q) - c(\Theta', q)\;.$
    If there is $\Theta' \in \mathcal{T}[\Theta_k]$ s.t. $\Delta[\Theta', \Theta_k] \;>\; \frac{\epsilon}{2}$
      Then **Let** $\Theta_{k+1} \leftarrow \Theta'$
      Else **Return**[Θ_k].
  End For
End PALO

Figure 1: PALO Algorithm

Many speed-up learning systems fit into this framework; *cf.*, [MKKC86, DM86], [LRN86]. Most of these systems, however, climb to a new performance element (by incorporating a new macro, or an additional control heuristic, etc.) after observing a *single* query; in each case, forming a new performance system that would work better *on that specific query*. Unfortunately, the resulting system may not work well over *the entire distribution of queries*, as this one query is unlikely to be representative; this issue leads to the *utility problem* [Min90]. One way to address this problem is to include a pruning process: *After* adding in the rules that appeared useful on single queries, this pruner would use a set of subsequent queries to identify, and then remove, the rules that have "negative utility".

This position paper, however, advocates a more cautious approach: Only climb to a new element (*e.g.*, modify the derivation strategy, or add a new macro) if we are confident that the resulting element is better than the current one. Such cautious adaptive systems (*e.g.*, COMPOSER [GD92] and PALO [Gre92, GJ92]) first observe a statistically significant set of queries, implicitly computing the empirical expected costs of an element with, versus without, a proposed modification. They then climb to the modified element if it is, with high probability, superior to the original one.

PALO **Adaptive System:** The rest of this section discusses the adaptive system, PALO, that is shown in Figure 1.[4] PALO takes as arguments an initial perfor-

[4]This procedure uses the value λ, which is the largest value of the c cost function: $\forall \Theta \in \mathcal{S}_\Theta, q \in \mathcal{Q}, 0 \le c(\Theta, q) \le \lambda$.

mance element Θ_1, along with error and confidence parameters $\epsilon, \delta > 0$. It also uses a set of possible transformations $\mathcal{T} = \{\tau_i\}_i$, where each τ_i maps one performance element to another; here by performing one re-ordering of the possible reductions. The set $\mathcal{T}[\Theta] = \{\tau_i(\Theta)\}_i$ defines the set of Θ's neighbors. PALO will climb from Θ_k to one of its neighbors, $\Theta' \in \mathcal{T}[\Theta_k]$, if this Θ' is statistically likely to be superior to Θ_k; *i.e.*, if we are highly confident that $C[\Theta_{k+1}] < C[\Theta_k]$. This constitutes one hill-climbing step; in general, PALO will perform many such steps, climbing from Θ_1 to Θ_2 to Θ_3, and so on, until terminating on reaching Θ_m. At this point, we are confident that none of Θ_m's neighbors $\mathcal{T}[\Theta_m]$ is more than ϵ better than Θ_m. Theorem 1 specifies PALO's behavior more precisely; its proof appears in the [Gre93b].

Theorem 1 *The* PALO$(\theta_1, \epsilon, \delta)$ *algorithm incrementally produces a series of performance elements* $\Theta_1, \Theta_2, \ldots, \Theta_m$ *such that, with probability at least* $1-\delta$, *both*

1. *the expected cost of each successive element in the series is strictly better than its predecessors'; i.e.,*
$$\forall i > j,\ C[\Theta_i] < C[\Theta_j]$$
2. *(once* PALO *terminates) the final ordering* Θ_m *is an "*ϵ*-local optimum"; i.e.,*
$$\forall \tau \in \mathcal{T},\ C[\Theta_m] \leq C[\tau(\Theta_m)] + \epsilon\ .$$

Moreover, PALO *terminates with probability* 1, *and on each iteration, requires only a number of samples that is polynomial in* $1/\epsilon$, $1/\delta$ *and* $|\mathcal{T}|$. □

4 Issues

Notice the overall "performance+adaptation" system is both solving relevant problems q_i (at each instant, using the Θ_k performance element) and collecting the statistical data required to decide whether to climb to a new Θ_{k+1}. Our objective is for the "adaptive component" to be quite efficient, so the overall performance+adaptation system is essentially as efficient as the underlying performance system. The only challenge, here, is in computing $\Delta[\Theta', \Theta_k] \leftarrow \frac{1}{|S_k|}\sum_{q \in S_k} c(\Theta_k, q) - c(\Theta', q)$ for each $\Theta' \in \mathcal{T}[\Theta_k]$. The obvious way of computing this value involves first running each Θ' on each $q \in S_k$, for each individual $\Theta' \in \mathcal{T}[\Theta_k] \cup \{\Theta_k\}$. This can be expensive, as it involves $(|\mathcal{T}[\Theta_k]| + 1) \times |S_k|$ (non-trivial) computations, of which only $|S_k|$ are required to solve the performance task. A partial solution to this, explored in [GJ92], is to approximate this $\Delta[\Theta', \Theta_k]$ quantity using guaranteed upper and lower bounds, $L(\Theta', \Theta_k) \leq \Delta[\Theta', \Theta_k] \leq U(\Theta', \Theta_k)$, which can be computed efficiently, based only on information obtained by observing the current Θ_k solve each problem $q \in S_k$.

Extensions: There are many obvious extensions, both to the PALO algorithm in particular and to the idea of a "cautious adaptor" in general. The PALO algorithm shown here works in a "batched incremental" mode, as it iteratively uses a *set* of samples to decide whether to climb to a new theory, or to terminate. There is also a strictly-incremental variant of this algorithm, which observes samples one-by-one, and decides after each individual sample, whether to climb, terminate, or simply draw an additional sample; see [Gre92]. There are other variants that will climb only a fixed number of steps [GS92a]. All of these systems are guaranteed to work appropriately for an arbitrary distribution of queries; there are yet other PALO-variants that are designed to handle certain specific distributions. For example, [Gre93a] describes a system that is guaranteed to work effectively if the distribution of $c(\Theta, \mathbf{q})$ values are normally distributed. [Gre93a] also provides a preliminary empirical study of these different PALO-ish systems, which suggests when each works most effectively.

Notice that this PALO system is described in terms of a given set of transformations $\mathcal{T} = \{\tau_i\}$. While our earlier example deals only with one type of transformation (namely, by rearranging the order in which the rule-based reductions are performed), there are many other ways of modifying a given $\text{Ask}_{\alpha_i}(KB_i, \cdot)$ performance element: We could change the knowledge base by adding in entailed clauses [Gre91] or removing redundant ones [MCK+89], or even "reformulating" by adding in clauses that involve newly-defined terms [Sub89, KS92]. We could also adjust other parameters of the derivation process, for example, by adding checks for loops, etc. Each of these types of modifications leads to its own space of performance elements. [Gre93a] discusses how to find a good space of such transformations, and specifies when these transformations will work effectively.

So far, we have been considering only "symbol level" modifications [Die86], whose objective is to improve the efficiency of the computation, but not to modify the set of answers returned. (*I.e.*, we have implicitly insisted that $\forall q\ \text{Ask}_{\alpha_i}(KB_i, q) = \text{Ask}_{\alpha_0}(KB_0, q)$. We can, however, use this same type of adaptive process for knowledge-level learning as well; [GS92a], for example, presents a PALO-like system that finds an optimally-*accurate* prioritized default theory. (See also [OM90].)

Finally, we can consider yet other ways of evaluating a given performance element. In general, we can use an (essentially) arbitrary user-specified utility function, which could perhaps quantify how much categoricity we are willing to sacrifice for an increase in efficiency [GE91]. As an example, [GS92b] describes an algorithm that re-represents a given theory into a new form (as a pair of horn theories), from which queries can be always be answered efficiently. However, the answers to some queries will be "I don't know", rather than the categorical "Yes" or "No". (Hence, while this new system is never incorrect, it may be silent on some queries.)

Future Work: There are several remaining challenges. First, the PALO framework assumes we have access to an arbitrarily large number of sample queries. It does not address the important, but distinct, challenge of producing the best possible element, given only a *specified number of samples*. Second, this framework assumes that the error and confidence terms ϵ and δ are given initially. We have not explored how to determine appropriate values for these terms. (Clearly ideas from both decision theory and the two-armed-bandit problem [BF85, NT89] are relevant.) Third, PALO can easily land in a *local* optimum that is not the globally best element. One obvious approach, which we yet to explore, is to combine PALO with some ideas from simulated annealing [KGV83]. Finally, PALO is designed to work in a *discrete* space of elements. It is not clear if it would apply to domains with an infinite number of elements, such as weights in a neural net.

References

[BF85] D. A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, London, 1985.

[BMSJ78] Bruce G. Buchanan, Thomas M. Mitchell, Reid G. Smith, and C. R. Johnson, Jr. Models of learning systems. In *Encyclopedia of Computer Science and Technology*, volume 11. Dekker, 1978.

[CM81] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog.* Springer-Verlag, New York, 1981.

[Die86] Thomas G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3):287–315, 1986. (Reprinted in "Readings in Machine Learning").

[DM86] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–76, 1986.

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

[GE91] Russell Greiner and Charles Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, pages 518–24, Sydney, Australia, August 1991.

[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.

[Gre93a] Russell Greiner. Palo algorithms. Technical report, Siemens Corporate Research, 1993.

[Gre93b] Russell Greiner. Probabilistic hill-climbing: Theory and applications. Technical report, Siemens Corporate Research, 1993.

[GS92a] Russell Greiner and Dale Schuurmans. Learning an optimally accurate representational system. In *ECAI Workshop on Theoretical Foundations of Knowledge Representation and Reasoning*, Vienna, August 1992.

[GS92b] 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.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[KS92] Henry Kautz and Bart Selman. Speeding inference by acquiring new concepts. In *Proceedings of AAAI-92*, San Jose, July 1992.

[LB85] Hector Levesque and Ron Brachman. A fundamental tradeoff in knowledge representation and reasoning. In Ron Brachman and Hector Levesque, editors, *Readings in Knowledge Representation*, pages 41–70, Los Altos, CA, 1985. Morgan Kaufmann Publishers, Inc.

[Lev84] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[LRN86] John E. Laird, Paul S. Rosenbloom, and Allan Newell. *Universal Subgoaling and Chunking: The Automatic Generation and*

Learning of Goal Hierarchies. Kluwer Academic Press, 1986.

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

[Min90] Steven Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2–3):363–391, March 1990.

[MKKC86] Thomas M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Example-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[NT89] Kumpati S. Narendra and Mandayam A. L. Thathachar. *Learning automata: an introduction.* Prentice Hall, Englewood Cliffs, N.J., 1989.

[OM90] Dirk Ourston and Raymond J. Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of AAAI-90*, pages 815–20, 1990.

[Smi89] David E. Smith. Controlling backward inference. *Artificial Intelligence*, 39(2):145–208, June 1989.

[Sub89] Devika Subramanian. *A Theory of Justified Reformulations.* PhD thesis, Stanford University, March 1989.