Genetic Programming with Guaranteed Quality

Stefan Droste
Lehrstuhl Informatik II
Universität Dortmund
44221 Dortmund, Germany
droste@ls2.informatik.uni-dortmund.de

ABSTRACT

When using genetic programming (GP) or other techniques that try to approximate unknown functions, the principle of *Occam's razor* is often applied: find the simplest function that explains the given data, as it is assumed to be the best approximation for the unknown function. Using a well-known result from learning theory, it is shown in this paper, how Occam's razor can help GP in finding functions, so that the number of functions that differ from the unknown function by more than a certain degree can be bounded theoretically. Experiments show how these bounds can be used to get guaranteed quality assurances for practical applications, even though they are much too conservative.

1 Introduction

Genetic programming (GP) is a paradigm for generating computer programs by using the basic principles of natural evolution: crossover, mutation, and selection (see "Koza (1992)" for a detailed introduction to GP). Often GP is used to find a function, that approximates an unknown function as good as possible, where the only information about the unknown function is a set of training examples, i.e. inputs with the correct output values of the unknown function. The main problem here is not to find a function that shows the prespecified behaviour on the given inputs, but resembles the unknown function on the other inputs, i.e. one wants to find functions with good generalization properties.

To solve this problem, often the principle of *Occam's razor* is applied, i.e. one tries to find the simplest function, that outputs the correct values for all inputs in the training set. It is assumed that simpler functions generalize better than the average function showing the desired input-output behaviour on the training examples. In GP, for example, one tries to find simple functions by using a parsimony factor, so that more complex functions get lower fitness than simpler functions, if both have the same fitness on the training examples. The principle of Occam's razor, although seeming intuitively reasonable, has never been proven to lead to functions that generalize better than functions resulting from other strategies. Furthermore, it is not clear, how to measure the simplicity of a function: the most often used measure, the size of the function (i.e. the size of the program representing the function) is dependent on the form of representation and can be misleading, if the program contains redundant code, that increases its size without its complexity. Nevertheless, many experiments in GP show that trying to find small functions leads to better generalizing functions than ignoring function size (see e.g. "Hooper and Flann (1996)", "Kinnear (1993)", "Rosca (1996)", and "Zhang and Mühlenbein (1995)").

In this paper the well-known Occam's razor theorem (see "Blumer et al. (1987)") from the field of learning theory is used to give an explanation on the influence the size of a function can have on its generalization properties. Roughly speaking, it states, that one should restrict the search space as much as possible while trying to find a good approximation of the unknown function, as this will reduce the probability of finding functions that differ by more than an error bound from the unknown function. If the search space is small enough in comparison to the number of used training examples and the error bound, Occam's razor theorem gives upper bounds on the probability, that many of the found functions differ by more than the error bound from the unknown function.

Because trying to find the smallest function that explains the training data results in restricting the search space to small functions, the principle of Occam's razor can therefore lead to provable good results (although Occam's razor theorem gives reason for the complementary strategy of trying to find the largest function that explains the data, too, as this strategy results in restricting the search space to large functions).

The next sections are organized as follows: section two formally defines the problem of finding a function, that approximates an unknown function given only by a number of training examples. Section three restates Occam's razor theorem and tries to explain some of its consequences. Experiments showing that the theoretical quality bounds are much too conservative and giving even stronger evidence, that the size of a function is related to its generalization quality, are described in the next section: first of all, the used GP system (see "Droste (1997)" for a more detailed description) is described, then it is shown how to upper-bound the number of functions of a certain maximal size (which is here used to apply Occam's razor theorem), and then the experimental results are presented. The last section contains the conclusion.

2 Definitions

In this section we want to formally define the problem of finding generalizing functions and show that all systems that generate functions being consistent with the given data have the same generalization quality, if we assume that all possible functions have the same probability of being the unknown function.

Let f be the unknown function we want to approximate as good as possible, where f is from the set F of all functions between A and B (A and B are arbitrary finite sets). As we do not want to model a system that is only used to approximate a single function but a whole class of functions, we assume that f is chosen randomly according to a probability distribution $P_F : \mathcal{P}(F) \to [0,1]$ (where $\mathcal{P}(F)$ is the set of all subsets of F).

The only information we get about the unknown function f is a set of inputs $X_i \in A$ with the correct output $f(X_i) \in B$, i.e. a set $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$, where m is the number of examples. We assume that the inputs X_i are randomly and independently chosen according to a probability distribution $P_A : \mathcal{P}(A) \to [0,1]$. Hence, it is possible that two inputs X_i and X_j ($i \neq j$) of the training examples are equal, even though the second input gives no additional information. We also use this probability distribution P_A to measure the error $Err(h,f) \in [0,1]$ of a hypothesis $h \in F$ with regard to the unknown function f:

$$Err(h,f) := P_A(\{X \in A \mid h(X) \neq f(X)\}).$$

So our goal is to find for an unknown function f, which is accessible to us only by a set of examples $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$, a hypothesis $h : A \to B$ minimizing $Err(h,f)$. If this task is done by a (GP) system S, which outputs a hypothesis h, when its input is $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$, with probability $P_S((X_1, f(X_1)), \ldots, (X_m, f(X_m)), h)$, our goal is to find a system S that minimizes the expected error $Err(S)$:

$$Err(S) := \sum_{h\in F} \sum_{f\in F} \sum_{(X_1,\ldots,X_m)\in A^m} P_F(f) \cdot P_A(X_1) \cdot \ldots \cdot$$

$$P_A(X_m) \cdot P_S((X_1, f(X_1)), \ldots, (X_m, f(X_m)), h) \cdot Err(f,h).$$

(Here we assume that all inputs X_i are chosen independently from each other and from the unknown function f.)

With respect to a given set $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$ of examples, we call a hypothesis $h \in F$ *consistent*, if it has the desired input-output-behaviour on all training examples, i.e. the set $C(X_1, \ldots, X_m, f)$ of consistent functions is defined by

$$C(X_1, \ldots, X_m, f) :=$$

$$\{h : A \to B \mid \forall i \in \{1, \ldots, m\} : h(X_i) = f(X_i)\}.$$

It seems reasonable only to use *consistent systems*, i.e. systems that output only hypotheses, which are consistent with the given examples, i.e. hypotheses from $C(X_1, \ldots, X_m, f)$, as there can be no benefit in ignoring the (in general small amount of) data we have about the otherwise unknown function. Any not consistent system can be simply improved by making its output-hypothesis consistent. So any system can be made consistent without quality loss.

If we assume that all functions in F have the same probability of being the unknown function, i.e. $\forall F' \subseteq F : P_F(F') = |F'|/|B|^{|A|}$, then all consistent systems S have the same expected error

$$\sum_{(X_1,\ldots,X_m)\in A^m} P_A(X_1) \cdot \ldots \cdot P_A(X_m) \cdot$$

$$\frac{(|B|-1) \cdot P_A(A \setminus \{X_1, \ldots, X_m\})}{|B|}.$$

This is valid, because for all inputs in $A \setminus \{X_1, \ldots, X_m\}$ the probability of guessing the correct output is $(|B|-1)/|B|$, as all $|B|$ possible outputs have the same probability of being the correct output and there is no information about the correct output.

So there is no system for finding a hypothesis that gives better results than any other system; all consistent systems have the same expected error, if we assume, that all functions have the same probability of being the unknown function (and all non-consistent systems have even greater expected error).

But what can we do, if all consistent systems have the same quality? First of all, in practical applications not all arbitrary functions shall be learned, but only "important" functions. This means that P_F has not the same value for all functions $f \in F$, but is greater for "important" functions and smaller for "less important" functions. But as long as we cannot define what those "important" functions are, we can only make empirically based statements.

Another possibility is to have a way of predicting the error of the found hypotheses, i.e. the system has no above-average quality over all unknown functions (as this is impossible), but there are indicators for the quality of the hypotheses. As the examples are chosen randomly and the system can work randomly, too, all we can hope for are indicators for a statement of a probabilistic kind. In the next section we show how we can get an indicator for a statement of the following kind: from the R found hypotheses the number of hypotheses h with $Err(h,f) < \varepsilon$ is with probability at least $1-\delta$ at least a number depending on R, δ, and ε.

3 Occam's razor theorem

Let us assume that for every set $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$ of training examples the (GP) system S outputs a hypothesis $h \in F$ that is consistent with this set, i.e. that the system is consistent. If the inputs $X_1, \ldots, X_m \in A$ are chosen randomly, the found hypotheses are likely to be different for different choices. But if the system would output the same hypothesis h for many different and large sets of examples, one could assume that h is a good approximation of the unknown function f, as it is consistent with f on many inputs. Analogously, if the system would output hypotheses of a prespecified class $H \subset F$ for many training examples, one could assume that the average approximation quality of the found functions in H is relatively high, if H contains relatively few functions.

Occam's razor theorem (see "Blumer et al. (1987)") contains an exact formulation of this common-sense-argumentation (for sake of completeness, the short proof of the theorem is added):

Theorem 1 (Occam's razor theorem) *Let H be a subset of F and f an arbitrary element of F. The probability of independently choosing m examples $X_1, \ldots, X_m \in A$ according to P_A, so that there is a function $h \in H$, which is consistent with $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$ and has error $Err(h,f) > \varepsilon$, is less than $|H| \cdot (1-\varepsilon)^m$.*

Proof: If $h \in H$ is a function with $Err(h,f) > \varepsilon$, the probability of independently choosing m examples $X_1, \ldots, X_m$ of f with $h \in C(X_1, \ldots, X_m, f)$ is less than $(1-\varepsilon)^m$. If we assume, that all functions $h \in H$ have error greater than ε, the probability, that any function in H is an element of $C(X_1, \ldots, X_m, f)$, is less than $|H| \cdot (1-\varepsilon)^m$. As this assumption overestimates the probability, that there is a consistent function in H with error more than ε, the theorem is proven. $\square$

How can we use this theorem to get a probabilistic lower bound on the number of functions with error at most ε in the set of functions our (GP) system has found? Let us assume that our system was repeated R-times (where the training examples were chosen independently according to P_A for every run) and that k-times a consistent hypothesis of a prespecified subset H of functions between A and B was found. As the probability of independently choosing m examples, so that there is a consistent hypothesis in H with error greater than ε, is less than $|H| \cdot (1-\varepsilon)^m$, the probability that such a hypothesis was found at least l-times can be bounded above by

$$\binom{R}{l} \cdot \left(|H| \cdot (1-\varepsilon)^m\right)^l . \tag{1}$$

Another bound can be obtained using the Chernoff bound (see "Hagerup and Rüb (1989)" for a proof of the Chernoff bound). The Chernoff bound states that for n independent random variables $Z_i \in \{0,1\}$ with $E(Z_i) = p_i$, $p = (\sum_{i=1}^n p_i)/n$, and $\delta > 0$, the probability that the sum of the Z_i is at least $(1+\delta) \cdot n \cdot p$ can be bounded above in the following way:

$$P(\sum_{i=1}^{n} Z_i \geq (1+\delta) \cdot n \cdot p) \leq \left(\frac{\exp(\delta)}{(1+\delta)^{1+\delta}}\right)^{np} .$$

If Z_i ($i \in \{1, \ldots, R\}$) is one, if and only if in the i-th run a consistent hypothesis h of H with $Err(h,f) > \varepsilon$ was found, the sum $\sum_{i=1}^R Z_i$ is the random number of consistent hypotheses in H with error greater than ε, that were found during the R runs. As a consistent hypothesis of H with error greater than ε can only be found, if a consistent hypothesis in H with error greater than ε exists, the expected value of every Z_i is less than $|H| \cdot (1-\varepsilon)^m$. Hence, the probability, that at least $(1+\delta) \cdot R \cdot |H| \cdot (1-\varepsilon)^m$ such hypotheses are found, is bounded above by

$$\left(\frac{\exp(\delta)}{(1+\delta)^{1+\delta}}\right)^{R \cdot |H| \cdot (1-\varepsilon)^m} =: p(\delta, \varepsilon, R, |H|, m).$$

So for $l > R \cdot |H| \cdot (1-\varepsilon)^m$ the probability, that at least l hypotheses of H with error greater than ε are found, is less than

$$p\left(\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m} - 1, \varepsilon, R, |H|, m\right), \tag{2}$$

because:

$$\begin{aligned} l &= (1+\delta) \cdot R \cdot |H| \cdot (1-\varepsilon)^m \\ &\Longleftrightarrow \\ \delta &= \frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m} - 1. \end{aligned}$$

As neither bound (1) nor bound (2) is always (i.e. for all R,l,$|H|$,ε, and m) smaller than the other, the minimum of these two bounds is smaller than any one of these bounds alone. Hence, we obtain:

Corollary 1 *Let H be a subset of F, f an arbitrary element of F, and S a (GP) system that randomly outputs an element $h \in C(X_1, \ldots, X_m, f)$, when the training examples $\{(X_1, f(x_1)), \ldots, (X_m, f(X_m))\}$ are its input. If S is run R-times with independently chosen examples according to P_A, the probability, that at least l-times the output h is an element of H with $Err(h, f) > \varepsilon$, is less than*

$$\min\left(\binom{R}{l} \cdot (|H| \cdot (1-\varepsilon)^m)^l, \left(\frac{\exp(\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m} - 1)}{\left(\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m}\right)^{\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m}}}\right)^{R \cdot |H| \cdot (1-\varepsilon)^m}\right).$$

So if the system S during its R runs outputs a hypothesis of H exactly k-times, the probability that at least $k-l$ of these hypotheses have an error at most ε is at least

$$1 - \min\left(\binom{R}{l} \cdot (|H| \cdot (1-\varepsilon)^m)^l, \left(\frac{\exp(\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m} - 1)}{\left(\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m}\right)^{\frac{l}{R \cdot |H| \cdot (1-\varepsilon)^m}}}\right)^{R \cdot |H| \cdot (1-\varepsilon)^m}\right).$$

In order to make this lower bound as high as possible, we must restrict our search space as much as possible, so that $|H|$ becomes as small as possible. One possibility to do this, is Occam's razor: if the system tries to find consistent solutions, which are as small as possible, it automatically restricts itself to small solutions; as the number of functions of a given maximal size is restricted, the probability of having found hypotheses with small error can be lower bounded. Of course, as Occam's razor theorem gives only a rough bound, not every set H will lead to a lower bound greater than zero for all reasonable values of ε, i.e. to get a usable quality guarantee the set H has to be relatively small.

Look, for instance, at two GP systems that both use the number of hits on the training examples as a fitness measure, where only the second one additionally uses a parsimony factor. If we stop a system only if it has found a consistent solution, then the average size of the functions found by the first system is likely to be higher than the average size of those of the second system. Therefore, Corollary 1 gives us a better lower bound for the probability of finding good solutions for the second GP system than for the first one.

Although this theorem is used in "Blumer et al. (1987)" as an argument for the principle of Occam's razor, it can be used as a counter-argument, too: restrict the search for consistent functions to functions as large as possible. This will decrease the size of the solution-space, too, so Corollary 1 can be used to make predictions about the quality of the hypotheses of this system, too. But certainly it is more efficient to restrict the search space to functions as small as possible.

So the lessons to be learned from Occam's razor theorem are: restricting the search space can lead to hypotheses with probabilistic error bounds, independent of the way the search space is restricted. But one has to be careful not to restrict the search space too rigorously, as otherwise the system may not be able to find consistent hypotheses in the search space, because the search space probably does not include any consistent functions. In the next section it is shown that the GP system presented in "Droste (1997)" can find hypotheses from a search space small enough to guarantee a usable error bound of the found hypotheses using Corollary 1.

4 Application of Occam's razor theorem

In this section it is shown that Occam's razor theorem (and Corollary 1), even though being only a rough estimation, can be used in practical applications to get probabilistic bounds for the quality of the found hypotheses. The GP system we use was originally presented in "Droste (1997)", hence the next subsection will only give a rough outline. Then the experimental results are presented and compared to the probabilistic boundaries of Corollary 1.

4.1 GP with OBDDs

The used GP system outputs for a given set of m training examples $\{(X_1, f(X_1)), \ldots, (X_m, f(X_m))\}$ of an unknown Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ a consistent hypothesis $h : \{0,1\}^n \rightarrow \{0,1\}$ (i.e. $A = \{0,1\}^n$ and $B = \{0,1\}$ in the context of section 2). As all inputs of the examples are chosen with equal probability from $\{0,1\}^n$ (i.e. $P_A(A') = |A'|/2^n$ for all $A' \subseteq \{0,1\}^n$), the error $Err(h, f)$ is simply the number of inputs, where the values of h and f disagree, divided by 2^n. Because we restrict our attention to Boolean functions, the individual functions are not represented by S-expressions, but by a very efficient data structure for the representation of Boolean functions called Ordered Binary Decision Diagram (OBDD) (see "Bryant (1986)"). The used GP system tries to output an OBDD, that is consistent with the training examples and as small as possible.

An OBDD is defined as follows: given a permutation π of the n Boolean input variables $x_1, \ldots, x_n$, an OBDD is a directed acyclic graph with exactly one source and two sinks, where:

- every non-sink node is labeled by one of the Boolean input variables $x_1, \ldots, x_n$

- every non-sink node has exactly two outgoing edges, labeled 0 and 1 (leading to the *0-successor* and to the *1-successor*, resp.)
- one of the sinks is labeled by 0 and the other by 1
- the two sinks have no outgoing edges
- if there is an edge from a node labeled x_i to a node labeled x_j, then $\pi(x_i)$ has to be smaller than $\pi(x_j)$.

An OBDD is evaluated for an input $(a_1, \ldots, a_n) \in \{0,1\}^n$ by starting at the source and following the a_i-edge, if the actual node is labeled by x_i, until a sink is reached. The label of the sink is the value of the function. So the OBDD O in Figure 1 represents the Boolean function $f_O(x_1, x_2, x_3) = x_1\overline{x}_3 \vee \overline{x}_1 x_2 \overline{x}_3$ (for $\pi = (x_1, x_2, x_3)$).

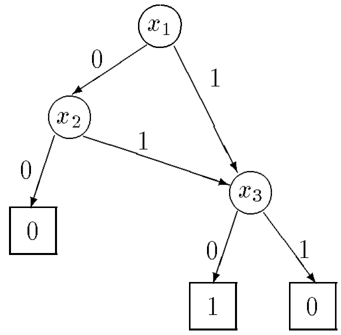

Figure 1: An example OBDD O

As every variable can appear at most once on a path, every path must have length at most n; hence, an OBDD can be evaluated in time $O(n)$, where n is the number of variables, while for an S-expression one needs time linear in the size of the S-expression for evaluation.

Furthermore, only *reduced* OBDDs are used in the GP system, i.e. OBDDs which have no nodes with identical 0- and 1-successor and do not contain any isomorphic subgraphs. It is well-known, that, given a Boolean function f, the reduced OBDD representing f is uniquely determined, and that the number of nodes in a reduced OBDD directly corresponds to the number of subfunctions of the represented function (see "Bryant (1986)"). Therefore, the number of nodes of a reduced OBDD (where the two sinks are not counted, as both appear in every OBDD not representing a constant function) is a well defined measure for the complexity of the represented function, which is not the case for general S-expressions, which can contain redundant code.

The GP system furthermore only uses reduced OBDDs, which are consistent with the training examples. This is done by replacing new children, which are not consistent with the training examples, by their parents and generating only consistent, but otherwise random individuals in the initial generation. Mutation (10% probability) and crossover (90% probability) are used in slight variation of the standard operators (see "Koza (1992)") to guarantee that the resulting structures are still valid reduced OBDDs. As consistence with the training examples is now guaranteed, only the size of the reduced OBDD influences the fitness, i.e. a smaller reduced OBDD has higher fitness. All parameter settings (if not stated otherwise in subsection 4.3) follow the much more detailed description of the GP system in "Droste (1997)". All in all, this GP system tries to find a hypothesis that is consistent with the training examples and has an OBDD-representation with as few nodes as possible (a well-known and NP-hard problem, see "Sauerhoff and Wegener (1994)").

This GP system has certain advantages for our goal of applying Occam's razor theorem: first of all, as it uses only consistent OBDDs, its output is always a consistent function, which is necessary to apply Occam's razor theorem (although both the system and the theorem can easily be extended to handle hypotheses, which are consistent on all but m' training examples). Furthermore, it was shown by experiments in "Droste (1997)" that this system can generate relatively small OBDDs (sometimes even smaller than the OBDD of the unknown function, which is possible, as the training examples only contain a small amount of information about the unknown function). This probably means that the system finds enough functions of a subspace $H \subset \{f : \{0,1\}^n \rightarrow \{0,1\}\}$ small enough to get reasonable error bounds by using Occam's razor theorem.

4.2 Counting the number of OBDDs

Because the set of functions with reduced OBDDs of size at most k and with exactly n variables will be used in the next subsection as the set H of Corollary 1, we need to compute an upper bound on the number of such functions: therefore, we will generalize the concept of reduced OBDDs to reduced OBDDs with s sinks (instead of two sinks in the original concept). Let $T(n,k,s)$ be an upper bound on the number of reduced OBDDs on n variables (w.l.o.g. we assume that the variable ordering π is $(x_1, \ldots, x_n)$) with exactly k non-sink nodes and at most s sinks. An OBDD of such kind can be split in three parts:

- the source with label x_i ($i \in \{1, \ldots, n\}$)
- the subOBDD O_0 starting at the 0-successor of the source, i.e. the structure of all nodes we reach, when we make a depth-first-search starting at the 0-successor
- the structure O_1, which starts at the 1-successor of the source, but where all nodes already contained in the subOBDD O_0 are excluded.

If the number of nodes in O_0 is $k_0 \in \{0, \ldots, k-1\}$, there are at most $T(n-i, k_0, s)$ possible subOBDDs O_0, because all nodes must have labels x_j with $j > i$. Then O_1 must contain $k - k_0 - 1$ nodes, and all its nodes must have a label x_j with $j > i$, too, so the number of variables is at most $n - i$. But as all nodes of O_0 are excluded from O_1, all nodes of O_0 can additionally be sinks of O_1, i.e. the number of sinks of O_1 is bounded above by $s + k_0$. Therefore, the number of OBDDs with n variables, k nodes, and s sinks can be bounded above by $T(n, k, s)$ defined by the following recursive formula:

$$\sum_{i=1}^{n} \sum_{k_0=0}^{k-1} T(n-i, k_0, s) \cdot T(n-i, k-k_0-1, s+k_0).$$

The recursion starts from the following cases (tested in this order):

1. If $k = 0$ then $T(n, k, s) = s$, as an OBDD without nodes must be a sink.
2. If $n = 0$ then $T(n, k, s) = 0$, as an OBDD with at least one node but without variables does not exist.
3. If $k = 1$ then $T(n, k, s) = n \cdot s \cdot (s-1)$, as an OBDD with one node is determined by its label and its two sink-successors.
4. If $k = 2$ then $T(n, k, s) = \sum_{i=1}^{n-1} 2 \cdot s \cdot (n-i) \cdot s \cdot (s-1)$, as an OBDD with two nodes is determined by the label x_i of its source, the sink-successor of the source and the other non-sink successor of the source.

Using this recursive formula, the number of OBDDs with n variables, two sinks, and at most k nodes can be upper bounded by:

$$S(n, k) := \sum_{i=0}^{k} T(n, i, 2).$$

This upper bound is used together with Corollary 1 in the next subsection to make probabilistic error guarantees for the hypotheses found by the GP system.

4.3 Experimental Results

In this subsection experimental results are presented which show how Occam's razor theorem can be applied to make probabilistic quality guarantees for the results of the experiments. The guaranteed quality of some experiments is very low, as for some test functions the found OBDDs were much too big. But in the light of the results of section 2 and as only a rough upper bound on the number of OBDDs is used, this is no surprise.

For every single run of the GP system $m = 512$ training examples $X \in \{0,1\}^n$ were chosen independently and with equal probability and given to the GP system together with the correct output $f(X)$ of the unknown function f. Then the GP system was run for $G = 2500$ generations with 50 individuals per generation, and the function represented by the smallest OBDD (i.e. the *Best-of-Run* OBDD) found during those G generations was the output-hypothesis h. This was repeated 100-times, so that 100 hypotheses were found.

To use Occam's razor theorem, one has to predetermine a set H of hypotheses, which, on the one hand, is relatively small, but, on the other hand, contains enough functions, so that a great number of found consistent hypotheses comes from H. The described first 100 runs were just used to find this set H: if a is the average size of the 100 Best-of-Run (BoR) OBDDs, the set H for the second run (consisting of 100 single runs) was chosen as the set of functions with OBDDs of size at most $\lfloor a \rfloor$. Then the GP system was run 100-times in the same way again, and the number of BoR-hypotheses found in H was counted. As one can assume that the distribution of the sizes of the found OBDDs is more or less the same for different runs (consisting of 100 single runs each), the number of found hypotheses of H should be high in the second run, too.

Using the results of the second 100 runs (i.e. the number of found hypotheses in H), Corollary 1 was used to get probabilistic error bounds for these hypotheses. Then these results were compared with the real error of the found hypotheses, which could be computed as we used well-known test functions as the unknown (to the GP system, not to us) functions. The tests consist of five different pairs of functions and variable orderings (which have great influence on the sizes of the OBDDs of the unknown function and the found hypotheses, too):

- MUX_{20}^{+}: The 20-multiplexer MUX_{20} defined by

  $$MUX_{20}(a_0, \ldots, a_3, d_0, \ldots, d_{15}) := d_{8a_0+4a_1+2a_2+a_3},$$

  (i.e. it outputs the value of the data bit adressed by the adress bits) with the variable ordering $a_0, \ldots, a_3, d_0, \ldots, d_{15}$. Using this variable ordering the reduced OBDD of MUX_{20} contains 31 nodes (no ordering leads to an OBDD for MUX_{20} with less nodes).

- MUX_{20}^{-}: The 20-multiplexer MUX_{20} with the variable ordering $d_0, \ldots, d_{15}, a_0, \ldots, a_3$. Using this variable ordering the reduced OBDD of MUX_{20} contains more than 65535 nodes (no ordering leads to a reduced OBDD for MUX_{20} with more nodes).

- ADD_{20}^{+}: The 20-adder ADD_{20} defined by

  $$ADD_{20}(x_0, y_0, \ldots, x_9, y_9) := s_0,$$

  where s_0 is the most significant bit of the binary representation of the sum of the numbers $\sum_{i=0}^{9} 2^{9-i} \cdot x_i$ and $\sum_{i=0}^{9} 2^{9-i} \cdot y_i$. The variable ordering $x_0, y_0, \ldots, x_9, y_9$ allows an OBDD to have a low error rate, even if only the first variables are tested.

- ADD_{20}^-: The 20-adder ADD_{20} with the variable ordering $x_9, y_9, \ldots, x_0, y_0$, making it impossible to achieve a low error rate, if only the first variables are tested.
- PAR_{20}: The 20-parity function PAR_{20} defined by

$$PAR_{20}(x_0, \ldots, x_{19}) := x_0 \oplus \ldots \oplus x_{19}$$

with the variable ordering $x_0, \ldots, x_{19}$. This function has a reduced OBDD with only 39 nodes, which contains every variable on every path from the source to one of the sinks (independent of the variable ordering).

As for every test function the number n of variables is 20, the $m = 512$ randomly chosen training examples give only a very small amount of information on the behaviour of the function on all $2^{20} = 1048576$ possible inputs.

The average size and the average error (in percent of all 2^{20} inputs) of the BoR-OBDDs over the first 100 runs are shown in Table 1.

Function	Avg. OBDD Size	Avg. Error
MUX_{20}^+	61.0	9.8 %
MUX_{20}^-	253.5	46.4 %
ADD_{20}^+	30.4	4.5 %
ADD_{20}^-	165.0	33.3 %
PAR_{20}	276.3	50.0 %

Table 1: **Results of the first 100 runs**

These average sizes of the BoR-OBDDs were used for the second run (consisting of 100 single runs) to predetermine the set H, e.g. for ADD_{20}^+ the set H is the set of all OBDDs with at most 30 non-sink nodes. The average error is added here only to see the correspondence between the average size and the average error during these experiments: the smaller the OBDDs, the better they generalize, i.e. the smaller is their error (on average). The parity function plays a special role here, as an OBDD has to contain at least one path, where all variables were tested, to achieve more than 50% hits. Because all BoR-OBDDs contained no path, where all variables were tested, all BoR-OBDDs had exactly 524288 (i.e. 50%) hits.

So in the second run, consisting of $R = 100$ single runs, it was counted how often the BoR-OBDD had size at most $\lfloor a \rfloor$, where a is the average size of the BoR-OBDDs of the first 100 runs. Then Corollary 1 was used together with the upper bound $S(n, \lfloor a \rfloor)$ on the number of OBDDs to compute the smallest ε, so that at least half of these OBDDs with at most $\lfloor a \rfloor$ nodes have an error at most ε with 99% probability. The results are shown in Table 2.

So for ADD_{20}^+ in the second 100 runs $k = 41$ BoR OBDDs with at most 30 nodes were found. With probability 99% at least 21 (i.e. at least one half of all 41)

Function	$\lfloor a \rfloor$	#BoR-OBDDs $\leq \lfloor a \rfloor$	Error ε
MUX_{20}^+	61	61	50.6 %
MUX_{20}^-	253	49	97.2 %
ADD_{20}^+	30	41	26.9 %
ADD_{20}^-	165	50	88.9 %
PAR_{20}	276	55	98.1 %

Table 2: **Error guarantees for the second 100 runs**

of them have error at most 26.9%. For the other functions the probabilistic error bounds are much worse, as $\lfloor a \rfloor$ is much higher, i.e. the found consistent OBDDs are on average much bigger. So for the other function types the error of at least half of the found OBDDs with at most $\lfloor a \rfloor$ nodes can only be probabilistically guaranteed to be lower than a bound greater than 50.25%. But every upper bound greater than 50.25% has no practical value, as randomly choosing a function results with high probability in better functions:

If we assume that a function is chosen randomly by independently choosing the function values for all inputs with equal probability, the expected value of the number of incorrectly guessed outputs of such a function is

$$\frac{1048576}{2} = 524288.$$

Using the Chernoff bound one can show that with probability at least 99%, the number of incorrectly guessed outputs is at most 0.5% higher than the expected value, i.e. at most

$$1.005 \cdot 524288 = 0.5025 \cdot 1048576.$$

Hence, an error rate of at most 50.25% can even be guaranteed with 99% probability by randomly guessing the outputs. This does not mean, that the results of the GP system for all other functions than ADD_{20}^+ are worse than randomly chosen functions, but that the probabilistic error guarantee only results in good bounds, when the found OBDDs are relatively small.

Nevertheless, Occam's razor theorem and Corollary 1 make it possible to give probabilistic error guarantees for the results of the GP system. The GP system used here is not even optimized, as for the function types MUX_{20}^-, ADD_{20}^-, and PAR_{20} the sizes of the found OBDDs after $G = 2500$ generations were far from converging, i.e. running the system for more generations would have resulted in smaller BoR-OBDDs, which means better probabilistic error bounds. But this paper is not intended to show that the used GP system gives better results than any other system, but it shall show how Occam's razor theorem can be used to get probabilistic error bounds for a GP system.

4.4 Majority Function

When using Occam's razor theorem (or Corollary 1, resp.), one gets a probabilistic lower bound on the number of hypotheses found with error at most ε. Now one can ask for the practical value of this result, as it is not known, which of the found hypotheses have an error at most ε. For the second 100 runs of ADD_{20}^{+}, for instance, we know that with 99% probability at least 21 of the found 41 hypotheses with OBDD-size at most 30 have an error of at most 26.9%, but it is not known, which these hypotheses are.

To get a hypothesis f^* with small error, the majority of the found hypotheses of H was used (an idea, which was used more sophistically by "Zhang and Joung (1997)" in GP before to use the information contained in the whole population): so $f^*(X)$ is defined as the value the majority of the found hypotheses of H has for input X; if there is no majority, $f^*(X)$ is randomly chosen as 0 or 1. This strategy resulted in rather good results for the experiments made. Table 3 shows the average error of the hypotheses of H found in the second 100 runs and the error of the majority hypothesis f^*.

Function	Ave. Error	$Err(f^*, h)$
MUX_{20}^{+}	5.5 %	0.0 %
MUX_{20}^{-}	46.5 %	40.7 %
ADD_{20}^{+}	4.5 %	3.1 %
ADD_{20}^{-}	32.7 %	17.6 %
PAR_{20}^{+}	50.0 %	50.0 %

Table 3: **Average error in comparison to majority**

So for the 20-multiplexer MUX_{20}^{+} with the optimal variable ordering the majority function f^* of the found hypotheses was the 20-multiplexer, i.e. using only at most 51200 training examples (512 examples for every of the 100 runs) from 1048576 possible inputs, the unknown function was exactly approximated. For all other function types the majority of the found hypotheses had a much smaller error than the average of the found hypotheses of H; the only exception is the 20-parity function, where even the majority had 50% error.

So the experiments show how Occam's razor theorem can be applied to get probabilistic error bounds for the hypotheses found by the GP system. But as Occam's razor theorem gives only a rough upper bound and the number of OBDDs is roughly upper bounded, too, the error guarantees are much too conservative.

5 Conclusion

This paper shows how Occam's razor theorem, a well-known result from the field of learning theory, can be used to get probabilistic error bounds for the results of a GP system, that tries to approximate an unknown function only given by a set of training examples. Occam's razor theorem gives a theoretical foundation for the strategy of restricting the search space as much as possible, as this strategy will decrease the probability of finding functions, that differ much from the unknown function. As trying to find the smallest solution, that explains the given training data, results in restricting the search space to small functions, Occam's razor theorem gives reason for the principle of Occam's razor, which is often successfully used in GP.

A number of experiments show, how Occam's razor theorem can be applied and that the theoretical bounds are much too conservative, as the resulting functions have much better generalization properties than theoretically guaranteed. But the intention of the paper is not to show that the used GP system outputs better hypotheses than any other system, but to show how to get theoretically founded probabilistic error bounds (exemplified by the used GP system).

Acknowledgments

Hereby I thank Thomas Jansen, Rainer Menke, and Ingo Wegener for their help while preparing this paper. This research was supported by the Deutsche Forschungsgemeinschaft as part of the Collaborative Research Center "Computational Intelligence" (531).

Bibliography

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M.K. 1987. Occam's Razor. *Information Processing Letters.* Number 24. Pages 377-380.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers.* Volume 35. Pages 677-691.

Droste, S. 1997. Efficient Genetic Programming for Finding Good Generalizing Functions. *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Pages 82-87.

Hagerup, T. and Rüb, C. 1989. A Guided Tour of Chernoff Bounds. *Information Processing Letters.* Number 33. Pages 305-308.

Hooper, D. C. and Flann, N. S. 1996. Improving the Accuracy and Robustness of Genetic Programming through Expression Simplification. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Page 428.

Kinnear, K. E. 1993. Generality and Difficulty in Genetic Programming: Evolving a Sort. *Proceedings of the Fifth International Conference on Genetic Algorithms.* Pages 279-286.

Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

Rosca, J. P. 1996. Generality versus Size in Genetic Programming. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Pages 381-387

Sauerhoff, M. and Wegener, I. 1996. On the Complexity of Minimizing the OBDD Size for Incompletely Specified Functions. *IEEE Transactions on CAD.* Volume 15. Number 11. Pages 1435-1437.

Zhang, B.-T. and Joung, J.-G. 1997. Enhancing Robustness of Genetic Programming at the Species Level. *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Pages 336-342.

Zhang, B.-T. and Mühlenbein, H. 1995. Balancing Accuracy and Parsimony in Genetic Programming. *Evolutionary Computation.* Volume 3. Number 1. Pages 17-38.