Genetic program based data mining to reverse engineer digital logic

James F. Smith III*, ThanhVu H. Nguyen

Naval Research Laboratory, Code 5741, Washington, D.C., 20375-5000

ABSTRACT

A data mining based procedure for automated reverse engineering and defect discovery has been developed. The data mining algorithm for reverse engineering uses a genetic program (GP) as a data mining function. A genetic program is an algorithm based on the theory of evolution that automatically evolves populations of computer programs or mathematical expressions, eventually selecting one that is optimal in the sense it maximizes a measure of effectiveness, referred to as a fitness function. The system to be reverse engineered is typically a sensor. Design documents for the sensor are not available and conditions prevent the sensor from being taken apart. The sensor is used to create a database of input signals and output measurements. Rules about the likely design properties of the sensor are collected from experts. The rules are used to create a fitness function for the genetic program. Genetic program based data mining is then conducted. This procedure incorporates not only the experts' rules into the fitness function, but also the information in the database. The information extracted through this process is the internal design specifications of the sensor. Uncertainty related to the input-output database and the expert based rule set can significantly alter the reverse engineering results. Significant experimental and theoretical results related to GP based data mining for reverse engineering will be provided. Methods of quantifying uncertainty and its effects will be presented. Finally methods for reducing the uncertainty will be examined.

Keywords: data mining, knowledge discovery, genetic programs, genetic algorithms, reverse engineering, defect discovery

1. INTRODUCTION

An engineer must design a signal that will yield a particular type of output from a sensor device (SD). The engineer does not have design specifications for the sensor system and the machine may not be disassembled or invasively examined. The engineer might attempt to find the correct signal through trial and error, but this would be very time consuming and access to experimental resources is very expensive. To deal with this problem a genetic program (GP) based data mining (DM) procedure has been invented[1].

A genetic program is an algorithm based on the theory of evolution that automatically evolves populations of computer programs or mathematical expressions, eventually selecting one that is optimal in the sense it maximizes a measure of effectiveness, referred to as a fitness function[2-5]. The system to be reverse engineered is typically a sensor. The sensor is used to create a database of input signals and output measurements. Rules about the likely design properties of the sensor are collected from experts. The rules are used to create a fitness function for the genetic program. Genetic program based data mining is then conducted[3-6]. This procedure incorporates not only experts' rules into the fitness function, but also the information in the database. The information extracted through this process is the internal design specifications of the sensor. The design properties extracted through this process can be used to design a signal that will produce a desired output[1].

GPs require a terminal set and function set as inputs. The terminals are the actual variables of the problem. These can include a variable like "x" used as a symbol in building a polynomial and also real constants. The function set consists of a list of functions that can operate on the variables. When a GP was used as a DM function in the past to automatically create fuzzy decision trees, the terminals consisted of fuzzy root concepts and the functions consisted of fuzzy logical connectives and fuzzy modifiers[3-5].

* Correspondence: Email: jfsmith@drsews.nrl.navy.mil

Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2006,
edited by Belur V. Dasarathy, Proc. of SPIE Vol. 6241, 624103, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.664952

When the GP is used as a data mining function, a database of input and output information is required. In the case of fuzzy decision trees the database represented a collection of scenarios about which the fuzzy decision tree to be evolved would make decisions. The database also had entries created by experts representing decisions about the scenarios. The optimal fuzzy decision tree would be the one that could most closely reproduce the experts' decisions about the scenarios. When the GP is used as a data mining function for evolving digital logic (DL), the database contains inputs to the DL as well as measured outputs. The experts' opinions are manifested in the selection of the input and associated output to be included in the database. For the DL case an additional form of input consisting of "rules" about DL construction are included.

Section 2 discusses data mining and the use of a genetic program as a data mining function. Section 3 examines one of the digital logic designs to be reverse engineered using genetic program based data mining. Section 4 explains the genetic program's terminal set, function set, and fitness function. Section 4 also gives detailed formulations of the rule fitness, fitness score, input-output fitness, and overall fitness. Closed form results for a class of digital logic maps that provide a global maximum for the rule fitness are given. From these closed form results a short input-output (IO) data base is derived in close-form to facilitate uncertainty analysis. Section 5 provides experimental results with detailed descriptions of the evolutionary properties. The class of solutions found in section 4 is shown to be intrinsically related to the GP's evolutionary process. Finally, section 6 provides conclusions.

2. EVOLUTIONARY ALGORITHM BASED DATA MINING

Data mining is the efficient extraction of valuable non-obvious information embedded in a large quantity of data[6]. Data mining consists of three steps: the construction of a database that represents truth; the calling of the data mining function to extract the valuable information, e.g., a clustering algorithm, neural net, genetic algorithm, genetic program, etc; and finally determining the value of the information extracted in the second step, this generally involves visualization.

When used for reverse engineering, the GP, typically data mines a database to determine a graph-theoretic structure, e.g., a system's DL diagram or an algorithm's flow chart or decision tree[3-5]. The GP mines the information from a database consisting of input and output values, e.g., a set of inputs to a sensor and its measured outputs. GP based data mining will be applied to the construction of the DL described in section 3.

To use the genetic program it is necessary to construct terminal and function sets relevant to the problem. Before the specific terminal and function sets for the reverse engineering problems are described, a more detailed description of one of the digital logic examples to be considered will be given in section 3.

3. THE DIGITAL LOGIC TO BE REVERSE ENGINEERED

One DL design to be reverse engineered is depicted in Figure 1. This DL is not known to the GP. The GP only has access to a database of input signals to the DL and measured output, as well as, a database of rules provided by experts for building the DL.

In Figure 1, the DL consists of three input channels each with a sensor attached. The sensors receive signals from sources one, two and three. Only measurements from sources one, the central source is of interest. Due to the geometry of the sources and properties of the sensors only sensor two can receive emissions from source one that are significant. Unfortunately, sensor two's measurement may be corrupted by emissions from sources two and three. The digital logic is constructed so that if there were significant corruption of sensor two's measurements, then the final OR-gate returns unity, so the measurements can be ignored.

In Figure 1 there are a number of DL elements depicted that are used repeatedly. DL components and signals will ultimately become elements of the GP's terminal and function sets. The sensors in Figure 1 will receive an analog signal and convert it to a digital form, i.e., they will map real-valued input to the set of integers. A sampling window of size N is used, i.e., the signal is sampled every Δt seconds for a total of N samples in that window. The sample is indicated by the vector $\vec{s}_j$ in (1) with sampling beginning at time t_o. The j-subscript implies the signal originated in the j^{th} source, where $j=1,2,3,$

$$\vec{s}_j = [s_j(t_o), s_j(t_o + \Delta t), \quad \ldots \quad , s_j(t_o + (N-1)\cdot \Delta t)]\,. \tag{1}$$

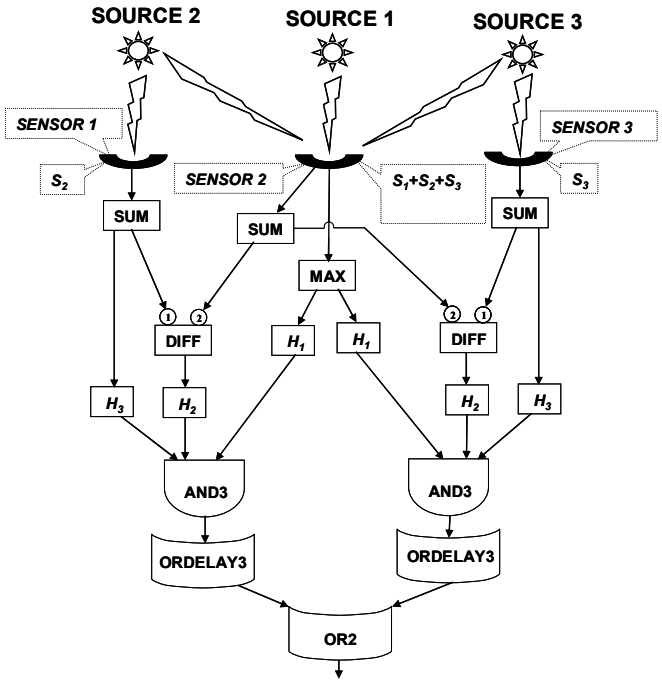

Figure 1: A system of three sensors designed to measure signals from source one while minimizing the corrupting influences of signals from sources two and three.

The DL function, *SUM*, given explicitly in (2), represents the logarithmic sum of the absolute value of the time components of the digitized input that has been received for a single window of length *N*,

$$SUM(\vec{s}_j) = \log\left[\sum_{k=1}^{N} \left| s_j(t_o + (k-1)\cdot \Delta t) \right|\right]. \tag{2}$$

The elements labeled H_i, for $i=1,2,3$, are Heaviside step functions as given in (3). If the input is greater than or equal to a threshold, τ_i, for $i=1,2,3$; then a value of unity is transmitted, otherwise a zero is transmitted,

$$H_i(s) = \begin{cases} 1, & if \quad s \geq \tau_i \\ 0, & if \quad s < \tau_i \end{cases}. \tag{3}$$

The DL function, *MAX*, given in (4), returns the common logarithm of the maximum absolute value of the time components of the input signal for a single window of length *N*. The element labeled *DIFF*, takes the difference between input to its first and second arguments as indicated in (5).

$$MAX(\vec{s}_j) = \log\left[\bigvee_{k=1}^{N} \left| s_j(t_o + (k-1)\cdot \Delta t) \right|\right] \tag{4}$$

$$DIFF(I_1, I_2) = I_1 - I_2 \tag{5}$$

The DL function, *OR3DELAY,* takes only Boolean inputs, i.e., it expects zero or one as an input. It waits until it has three consecutive inputs from three consecutive time windows, hence the "3" in its name. Once it receives three consecutive inputs, it yields as an output the maximum of its inputs. Also not depicted, but used in the GP's function set are *AND3DELAY*, which takes three inputs of zero or one corresponding to three consecutive time windows and yields as output the minimum of its inputs. Finally, the symbols labeled *AND3, OR3, AND2,* and *OR2* are the conventional logical connectives *AND* and *OR*, with the numerical designation indicating the number of inputs expected, e.g., *AND3* expects three Boolean inputs.

The signals are additive, at any given time sensor two may record a superposition of the three sources' transmissions, which is represented by $s_1(t)+ s_2(t) + s_3(t)$. If the three sensors' signals are of sufficient magnitude then this is characteristic of corruption and the final *OR* in Figure 1 returns unity.

4. TERMINAL SET, FUNCTION SET, FITNESS FUNCTIONS, CLOSED FORM RESULTS FOR A CLASS OF OPTIMAL DL MAPS AND INPUT-OUTPUT DATABASE

This section describes the terminal set, function set, fitness functions, closed form results for a class of DL designs and an input-output database. The description is given in terms of DL elements and properties, but the genetic program based reverse engineering technique is very general and can be applied to any system that can be described in a graph theoretic language, e.g., decision processes described in terms of decision trees[3-5].

4.1 Terminal and function sets

The terminal set consists of the following elements:

$$T=\{SUM\_SIG123, MAX\_SIG123, SUM\_SIG2, MAX\_SIG2, SUM\_SIG3, MAX\_SIG3\}, \quad (6)$$

where

$$SUM\_SIG123 = SUM(\vec{s}_1 + \vec{s}_2 + \vec{s}_3), \quad (7)$$

$$MAX\_SIG123 = MAX(\vec{s}_1 + \vec{s}_2 + \vec{s}_3), \quad (8)$$

$$SUM\_SIG2 = SUM(\vec{s}_2), \quad (9)$$

$$MAX\_SIG2 = MAX(\vec{s}_2), \quad (10)$$

$$SUM\_SIG3 = SUM(\vec{s}_3), \quad (11)$$

$$MAX\_SIG3 = MAX(\vec{s}_3). \quad (12)$$

All sensor measurements begin at time, t_o.

The function set consists of the following elements:

$$F=\{AND3, OR3, AND2, OR2, AND3DELAY, OR3DELAY, H_1, H_2, H_3, DIFF\}. \quad (13)$$

The function *AND3DELAY* is not used in Figure 1. By including it, the GP's ability to discriminate against extraneous functions is emphasized.

One of the DL designs the GP is to evolve is given in Figure 1. The GP's ability to do this will be determined largely by

the fitness function and the underlying databases to be discussed. The chromosome to be evolved by the GP is given in Figure 1 and represented in prefix notation in (14).

$$\begin{array}{c} OR2\ OR3DELAY\ AND3\ H_3\ SUM\_SIG3\ H_2\ DIFF\ SUM\_SIG3\ SUM\_SIG123\ H_1\ MAX\_SIG123 \\ OR3DELAY\ AND3\ H_3\ SUM\_SIG2\ H_2\ DIFF\ SUM\_SIG2\ SUM\_SIG123\ H_1\ MAX\_SIG123 \end{array} \tag{14}$$

4.2 Overall fitness, rule fitness, input-output fitness and database structure

As with all GPs there must be a fitness function for evaluation of the evolving population of chromosomes. The fitness function, referred to as the overall fitness (OF) denoted as f_{OF} is actually the sum of two other fitness functions. These functions are the rule fitness (RF) and the input-output fitness (IOF) denoted as f_{RF} and f_{IOF}, respectively. The rule fitness is given in (15) where the indicator function, I_i is unity if the i^{th} rule in Table 1 is satisfied and zero otherwise, and v_i is the value of the i^{th} rule as given in Table 1,

$$f_{RF} = \sum_{i=1}^{12} I_i \cdot v_i \,. \tag{15}$$

Let DL_j denote the j^{th} element of the evolving population of chromosomes within the GP for $j = 1,2,\ldots,m_{ps}$ where m_{ps} is the population size, i.e., the number of chromosomes. Let each DL_j consists of an OR2 or AND2 that connects two subgraphs, denoted as DL_j_left and DL_j_right. Let $l\left(DL_j\_\varsigma\right)$ be the length, i.e., the number of nodes in DL_j_ς, for $\varsigma \in \{left, right\}$. According to Table 1, if $l\left(DL_j\_\varsigma\right)$ is greater than or equal to 20 then the parsimony pressure, $\alpha_p \cdot l\left(DL_j\_\varsigma\right)$ is subtracted from the rule fitness followed by division by 100, ultimately yielding the rule score, denoted as g_{RS}. This subtraction is done if either $l\left(DL_j\_left\right)$ or $l\left(DL_j\_right\right)$ exceeds 20. The quantity α_p is referred to as the parsimony coefficient[1]. The rule score can be expressed compactly as

$$\begin{aligned} g_{RS}\left(DL_j\right) = & \frac{c}{100}\left\{f_{RF}\left(DL_j\_left\right) - \chi\left[l\left(DL_j\_left\right) - 20\right]\cdot\alpha_p \cdot l\left(DL_j\_left\right)\right\} \\ + & \frac{c}{100}\left\{f_{RF}\left(DL_j\_right\right) - \chi\left[l\left(DL_j\_right\right) - 20\right]\cdot\alpha_p \cdot l\left(DL_j\_right\right)\right\} \end{aligned}. \tag{16}$$

The parameter *c* in (16) is the product of the indicator functions for rules 14-21 in Table 1 for the closed form results in this section and *c* is unity for the computational results in section 5.

If the rule score exceeds the rule threshold denoted as, κ_{RT} then and only then is the input-output fitness evaluated. By forcing the rule score to exceed a threshold before the input-output fitness is evaluated a great deal of computational complexity is avoided.

Let DL^T denote the true digital logic diagram that underlies the SD used to construct the input-output database. For the examples considered in this paper let there be three signals. The input-output database is assumed to have the following structure

$$M_{DB} = \begin{bmatrix} \vec{S}_1^1 & \vec{S}_2^1 & \vec{S}_3^1 & B^1 \\ \vec{S}_1^2 & \vec{S}_2^2 & \vec{S}_3^2 & B^2 \\ \vdots & \vdots & \vdots & \vdots \\ \vec{S}_1^m & \vec{S}_2^m & \vec{S}_3^m & B^m \end{bmatrix}. \tag{17}$$

RULE SUMMARY		
R1: If either *OR3DELAY* or *AND3DELAY* are present during rule fitness evaluation add $v_1 = 5$.	R8: *no_diff2H*= number of times a *DIFF* feeds into a Heaviside step function. *no_diff* = number of *DIFF* operators in the DL. During rule fitness evaluation add $v_8 = 5 \cdot \left(\frac{no\_diff2H}{no\_diff} \right)$.	R15 Each subgraph takes the same form, i.e., it is a function of *z(SIG(i), SIG123)* where *i=2* for one subgraph and *i=3* for the other. If this rule is satisfied then $I_{15} = 1$, otherwise $I_{15} = 0$.
R2: If R1 is satisfied and *OR3DELAY* or *AND3DELAY* are present during rule fitness evaluation then add $v_2 = 6$.	R9: *no_diffw2sig*= no. times two consecutive signal feed into a *DIFF*. During rule fitness evaluation add $v_9 = 5 \cdot \left(\frac{no\_diffw2sig}{no\_diff} \right)$.	R16 Join the *z(SIG2,SIG123)*and *z(SIG3,SIG123)* subgraphs using *OR2* or *AND2*. If this rule is satisfied then $I_{16} = 1$, otherwise $I_{16} = 0$
R3: If *AND3* or *OR3* are present during fitness evaluation add $v_3 = 5$.	R10: *no_diffw2diffsig*= no. times a *DIFF* takes two different signals. During rule fitness evaluation add $v_{10} = 5 \cdot \left(\frac{no\_diffw2diffsig}{no\_diff} \right)$.	R17: The threshold τ_2 is used after the *DIFF*. If this rule is satisfied then $I_{17} = 1$, otherwise $I_{17} = 0$
R4: If none of the three comparators denoted as $H_i, i = 1,2,3$ follow *AND3* or *OR3* during fitness evaluation add $v_4 = 3$.	R11: *no_maxorsum* = number of time *DIFF* takes two *MAX_SIG* input or two *SUM_SIG* inputs. During rule fitness evaluation add $v_{11} = 5 \cdot \left(\frac{no\_maxorsum}{no\_diff} \right)$.	R18: The structure $H_m diff A_i^\sigma A_j^\sigma$ takes the form $H_2 diff A_i^\sigma A_{\{1,2,3\}}^\sigma$ where $i = \{2\}$ *or* $\{3\}$. If this rule is satisfied then $I_{18} = 1$, otherwise $I_{18} = 0$
R5: If there is only one delay present, i.e., in the form of *OR3DELAY* or *AND3DELAY* during rule fitness evaluation add $v_5 = 5$.	R12: During rule fitness evaluation add $v_{12} = 5 / no\_diff$.	R19: All thresholds must be used. If this rule is satisfied then $I_{19} = 1$, otherwise $I_{19} = 0$
R6: Let *have_not_alone_signal* = the number of times that *SUM_SIG_i* or *MAX_SIG_i*, *i={1,2,3}* or *2* or *3* is fed into $H_i, i = 1,2,3$ or DIFF. Let *num_signals*= the number of times *SUM_SIG_i* or *MAX_SIG_i* for *i={1,2,3}* or *2* or *3* appears in the chromosome.. During rule fitness evaluation add $v_6 =$ $5 \cdot \left(\frac{have\_not\_alone\_signal}{num\_signals} \right)$	R13: Penalizes the rule fitness by subtracting parsimony pressure if the chromosome length is equal to or greater than 20 and divide the difference by 100.	R20: Each subgraph takes the form $z = DC_3 H_m diff A_i^\sigma A_j^\sigma H_q A_i^\alpha H_r A_j^\beta$ as a result of the rules above. Assume that $\beta \neq \sigma$ and $\alpha = \sigma$. If this rule is satisfied then $I_{20} = 1$, otherwise $I_{20} = 0$
R7: If none of the $H_i, i = 1,2,3$ are adjacent to each to each other then during fitness evaluation add $v_7 = 5$	R14 One subgraph of the DL uses *SIG2* and *SIG123* as input; the other side uses *SIG3* and *SIG123* as input. If this rule is satisfied then $I_{14} = 1$, otherwise $I_{14} = 0$	R21: Each subgraph should be as short as is consistent with the above rules. If this rule is satisfied then $I_{21} = 1$, otherwise $I_{21} = 0$.

Table 1: Rule set for closed form results and computational GP experiments.

Where $\vec{S}_j^k$ is the three time window input from the j^{th} source for the k^{th} input; $B^k \in \{0,1\}$ is the k^{th} output from DL^T for $k=1,2,\ldots, m$, i.e.,

$$B^k = DL^T\left(\vec{S}_1^k, \vec{S}_2^k, \vec{S}_3^k\right); \quad for \quad k = 1,2,\ldots,m\,. \tag{18}$$

The input-output fitness for the j^{th} chromosome, DL_j, is defined as

$$f_{IOF}(j, M_{DB}) = 1\Big/\left[1 + \sum_{k=1}^{m}\left|DL_j\left(\vec{S}_1^k, \vec{S}_2^k, \vec{S}_3^k\right) - B^k\right|^2\right]. \tag{19}$$

The overall fitness, f_{OF}, for the j^{th} chromosome can be written as

$$f_{OF}(j, M_{DB}) = g_{RS}(DL_j) + \chi\left(g_{RS}(DL_j) - \kappa_{RT}\right)\cdot f_{IOF}(j, M_{DB})\,. \tag{20}$$

It is important to recall that in actual implementation, the input-output fitness is only evaluated if the rule fitness is greater than or equal to the rule threshold. Selectively evaluating the input-output fitness greatly reduces the computational complexity and hence the run-time of the GP.

4.3 Closed form results for class of DL maps and database that maximize the various fitness functions

It is possible through inspection of (16) to write down a set denoted as, Z_{MLDL}, of minimum length DL maps that result in a global maximum for the rule score. Analysis of Z_{MLDL} makes it possible to also write down a closed form expression for the image set, denoted as R_{MLDL}, under the DLs making up Z_{MLDL}. The elements of R_{MLDL} can be written as explicit functions of the parameters that characterize distinct elements of Z_{MLDL}. From the explicit form of R_{MLDL} it is possible to invert out a database matrix with closed form expressions for its elements. The closed form expressions are given in terms of sensor system parameters. Table 2 gives a small subset of a database created in this fashion. A database given explicitly in terms of the sensor system parameters can be used in uncertainty analysis and GP convergence tests. It is readily shown using the rule set and the closed form database that emerges from this approach that loss of rules or rows of the IO database can result in not one but many DLs that maximize the overall fitness.

Preliminary to the development described above it is useful to define the following subsets of the terminal and function sets. Let

$$D \in DELAY = \{AND3DELAY, OR3DELAY\}; \tag{21}$$

$$C_3 \in CONN3 = \{AND3, OR3\}; \tag{22}$$

$$C_2 \in CONN2 = \{AND2, OR2\}; \tag{23}$$

$$COMP = \{H_1, H_2, H_3\}; \tag{24}$$

$$A_a^\sigma, A_b^\sigma, A_i^\alpha, A_j^\beta \in MAXSUM = \{MAX\_SIG1, MAX\_SIG2, MAX\_SIG3, MAX\_SIG123,\ldots \\ SUM\_SIG1, SUM\_SIG2, SUM\_SIG3, SUM\_SIG123\}. \tag{25}$$

Input-Output Table for Three Signal Example									
Component	γ_1^q	γ_2^q	γ_3^q	$\gamma_1^{q'}$	$\gamma_2^{q'}$	$\gamma_3^{q'}$	Window2	Window3	Output
OR2	10^{τ_3}	*1*	$-2\cdot 10^{\tau_3}$	*0*	*0*	*0*	$\vec{O}(N)$	$\vec{O}(N)$	*1*
OR3DELAY	10^{τ_3}	-10^{τ_3}	-10^{τ_3}	*0*	*0*	*0*	$\vec{O}(N)$	$\vec{O}(N)$	*1*
AND3	*1*	*1*	10^{τ_3}	*0*	*0*	*0*	$\vec{O}(N)$	$\vec{O}(N)$	*0*
AND3	10^{τ_3}	*1*	*1*	*0*	*0*	*0*	$\vec{O}(N)$	$\vec{O}(N)$	*0*
τ_3 along Auxillary Antenna	10^{τ_1}	10^{τ_1}	-10^{τ_3}	*0*	*0*	*0*	$\vec{O}(N)$	$\vec{O}(N)$	*1*

Table 2: Subset of the Input-Output data base derived from the rules in Table 1.

The parameters σ,α and β carry the value "MAX" or "SUM" and a,b,i and j take the values $\{1\},\{2\},\{3\}$, and $\{1,2,3\}$. As examples of this notation, $A_{\{3\}}^{MAX}$ corresponds to *MAX_SIG3,* whereas $A_{\{1,2,3\}}^{SUM}$ corresponds to *SUM_SIG123.*

It follows from Rules 1-13 of Table 1 that subclass of minimum length DL maps that maximize the rule score take the following form

$$z(D,C_3,a,b,i,j,\sigma,\alpha,\beta,q,r) = DC_3 H_l DIFFA_a^{\sigma} A_b^{\sigma} H_q A_i^{\alpha} H_r A_j^{\beta}, \tag{26}$$

where

$$a \neq b \quad a,b = \{1\},\{2\},\{3\},\{1,2,3\}; \quad l,q,r = 1,2,3\,. \tag{27}$$

Rules 1-13 only govern half the DL design, i.e., one subgraph of the tree. It follows from Rule 14 of Table 1 that the other subgraph of the DL map will also take a form given by (26). It follows from Rules 15-21 that the DL map that includes both subgraphs takes the form

$$Z(D,C_3,\sigma,\sigma,\beta,q,r) = C_2 z(D,C_3,\{2\},\{1,2,3\},\{2\},\{1,2,3\},\sigma,\sigma,\beta,q,r) z(D,C_3,\{3\},\{1,2,3\},\{3\},\{1,2,3\},\sigma,\sigma,\beta,q,r), \tag{28}$$

where

$$\sigma \neq \beta\,. \tag{29}$$

Also from Rule 19 of Table 1 all thresholds must be used, so since threshold two has been used already, once one additional threshold is assigned the other is known.

Equation (28) gives a typical element of Z_{MLDL}. The use of all 21 rules in Table 1 reduces Z_{MLDL} to a set with 32 elements. It would hardly be worth using a GP for such a small number of alternative solutions. The 21 rules of Table 1 allow simple closed form results for the elements of Z_{MLDL}. Likewise, the simplicity of Z_{MLDL} arising from the 21 rules allows a database to be inverted in closed form. Under the database the overall fitness will have a single global maximum at the desired DL map, i.e., DL^T. The database subset taken from a much larger database given in Table 2 is also a closed form result, exhibiting the database matrix's explicit dependence on system parameter, i.e., thresholds. Table 2 is a subset of a database constructed using only rules 1-16 from Table 1.

The first and one of the most important steps in data mining is the construction of the database. The database is generally constructed by or at least "cleansed" by a domain expert so that it reflects truth. For this example it is valuable

to construct an ideal database. This database can actually be derived mathematically and rendered in closed form as explicit functions of the system parameters and DL graph labels that must be determined. A database of this kind is very effective in studying the effect of uncertainty on the ultimate DL evolved. For the analysis in this paper uncertainty refers to the loss of a rule or rules from Table 1 or the loss of at least one row from the database matrix given in Table 2.

Before giving the database subset it is essential to define some notation. Let

$$\vec{\Gamma}_i^q = \left[\gamma_i^q, \gamma_i^{q'}, \vec{O}(N-2) \right]; \quad for \quad i = 1,2,3; \quad q = 1,2,\ldots,m \; ; \tag{30}$$

where γ_i^q and $\gamma_i^{q'}$ are the first and second pulse values for the signal denoted by $\vec{\Gamma}_i^q$. The quantity $\vec{O}(n)$ is defined to be a row vector consisting of *n* zeros. The quantity $\vec{\Gamma}_i^q$ is the i^{th} source signal for the q^{th} measurement to be stored in the database, i.e., the q^{th} row in the database matrix (17) for one time window.

Table 2 provides a subset of a database derived by solving an analog of (28) for rules 1-16, by inspection or solving systems of inequalities arising from the analog of (28). Each row of Table 2 is determined to encourage the GP to place a particular label on the graph, the label of interest being given in the first column of that row. As indicated in the first row, the second through seventh columns give the values of γ_i^q and $\gamma_i^{q'}$ for signals *i=1,2, and 3*. The entry $\vec{O}(N)$ implies there are *N* zeros in the window. There is no need to have separate entries for each source signal for windows two and three as all three signals exhibit the same behavior, i.e., they have zero entries for the final two windows. The final column gives the simulated measured output of the system of interest.

Also, under ideal conditions there is great freedom in specifying input. When making measurements using a real SD the freedom to specify any desired input would not exists. The SD would only accept certain inputs, the output would likely be corrupted by noise and might be characterized by lost data. Given these difficulties a real database must be many orders of magnitude larger to attempt to deal with corruption and the fact that the observer could not select the most desirable input-output entries.

The rule set and database in Tables 1 and 2, respectively, have value for understanding the nature of uncertainty in expertise and data. If certain rules are removed from Table 1 then the number of potential DL maps grows rapidly. For example, if Rule 19 is eliminated from Table 1 then between the remaining rules and the database there is not enough information to uniquely specify the threshold labeling. So DL^T is no longer uniquely determined. The resulting DL maps will have underlying graphs that in graph-theoretic terms are isomorphic to the graph of DL^T. An isomorphism exists between two graphs if there exists a map that establishes a one-to-one correspondence between the vertices of the two graphs and the map preserves edges. If the goal of the observer was to determine a signal that would produce a certain response from the SD then obtaining DL maps that differ only in threshold labeling is frequently enough. If the observer knows the maximum possible threshold for the system, but does not know which thresholds label particular nodes, they can in many instances create a signal that gives the desired response by simply increasing the energy in the signal to correspond to the potential of the maximum threshold labeling a particular node. So in this case rule and database uncertainty is not an extreme liability. The input signal's functional form does not change; the true cost of uncertainty is a minor increase in the energy contained in the signal.

5. DATA MINING RESULTS

In this section two different DL schemes data mined by the GP are considered. The two examples presented here are representative of the many experiments that have been conducted to show the effectiveness of the GP based data mining procedure presented in this paper. The first is the DL represented in (14) and also in Figure 1. This DL will assume the value of DL^T for the discussion below. Using various databases too large to reproduce here and different random number generator seeds, the GP was able to reverse engineer (14) in no more than 76 GP generations. The different number of generations and amounts of CPU time required reflects the effect of different input-output databases and also

the random number generator seeds. One database may constrain the evolutionary process more than another resulting in fitness values that over time push the population more rapidly toward DL^T. Also, since the initial population is generated randomly; and crossover, mutation, architecture altering steps[1] (AAS) and symmetrical replication[1] (SR) have random aspects, a change in the seed of the random number generator can also impact run-time.

To get a feel for the evolutionary process it is useful to examine some intermediate generations that lead to DL^T. For the case in which (14) is reverse engineered in 76 generations the elite chromosomes found for different generations are provided in Table 3 with the 76th generation reproducing the correct chromosome given in (14).

Generation	Best Chromosome found in the Population for the Indicated Generation
1	*OR2 H_2 DIFF SUM_SIG123 MAX_SIG123 OR2 DIFF MAX_SIG2 H_1 MAX_SIG2 SUM_SIG3*
25	*OR2 AND3DELAY OR3 H_3 MAX_SIG2 H_2 SUM_SIG123 H_2 DIFF MAX_SIG2 MAX_SIG123 AND3DELAY OR3 H_2 SUM_SIG2 SUM_SIG3 H_2 DIFF MAX_SIG2 MAX_SIG123*
40	*OR2 OR3DELAY AND3 H_1 MAX_SIG2 H_1 MAX_SIG123 H_2 DIFF SUM_SIG2 SUM_SIG123 AND3DELAY AND3 H_3 SUM_SIG3 H_1 SUM_SIG2 H_2 DIFF SUM_SIG2 SUM_SIG3*
50	*OR2 OR3DELAY AND3 H_1 MAX_SIG2 H_1 MAX_SIG123 H_2 DIFF SUM_SIG2 SUM_SIG123 OR3DELAY AND3 H_1 MAX_SIG3 H_1 MAX_SIG123 H_2 DIFF SUM_SIG3 SUM_SIG123*
76	*OR2 OR3DELAY AND3 H_1 MAX_SIG123 H_3 SUM_SIG2 H_2 DIFF SUM_SIG2 SUM_SIG123 OR3DELAY AND3 H_1 MAX_SIG123 H_3 SUM_SIG3 H_2 DIFF SUM_SIG3 SUM_SIG123*

Table 3: Evolution of the DL depicted in Figure 1

The chromosomes entered into Table 3 reflect some of the characteristics observed during the evolutionary process. From the first generation forward the GP is able to find best candidates that have an *OR2* at the end of the chromosome. This property arises from Rule 16 in Table 1. The presence of two *DIFF* operators in the first generation is also promising. The best chromosome for the first generation is much too short when compared to the desired result.

New innovations are found in generation 25 in that both arguments of both *DIFF*s use *MAX* functions as well as the *SIG123* structure. Even though it is expected that both arguments will ultimately use *SUM* functions, the use of a common function for both arguments may show evolution in the proper direction. Both *DIFF* operators are preceded by H_2 which is what is found in (14). Even with these innovations the best chromosome of generation 25 is far from the correct result.

The best chromosome of the 40th generation has subgraphs consistent with the closed form result of (26). All generations after the 26th have elite chromosomes that assume this form. The computational results use a smaller set of rules, i.e., rules 1-16 than those used in deriving (27-29) so the rules alone do not require that the solutions converge to this form. The GP's effort from generation 27 through 76 involves finding a solution with the proper node labels. Various rows in the IO database contribute to proper labeling, e.g., if the fifth row in the database subset in Table 2 is deleted then it is likely the final GP solution would not have proper threshold labeling. An improper threshold value is undesirable from the standpoint of trying to reproduce the exact digital logic. If the goal is to produce an input signal that produces unity as an output then even with the threshold value wrong, as long as the input signal has sufficient energy to take into account uncertainty, then the desired output is obtained. In conclusion, the ultimate cost of information uncertainty in this case is a small amount of additional energy.

The best chromosome of the 50th generation is far closer to (14). The *MAX* functions in the arguments of the *DIFF* have been replaced by *SUM* functions. The arguments of the *DIFF* operators are the ones for the final result and the output of both *DIFF*s is passed into H_2 as found in (14). In this chromosome replacing *H_1 MAX_SIG2* with *H_3 SUM_SIG2* and *H_1 MAX_SIG3* with *H_3 SUM_SIG3* would yield the correct result. Finally, the desired result is found in generation 76.

For a second example consider the DL given below in (31) as DL^T, i.e., truth,

OR2 OR3DELAY AND3 H_1 SUM_SIG123 H_3 MAX_SIG2 H_2 DIFF MAX_SIG2 MAX_SIG123 OR3DELAY AND3 H_1 SUM_SIG123 H_3 MAX_SIG3 H_2 DIFF MAX_SIG3 MAX_SIG123. (31)

The GPs evolutionary process for inverting (31) is summarized in Table 4.

This example is similar to (14), in fact if in (14) the *MAX* operations are replaced by the *SUM* operation and *SUM* replaced by *MAX* then (31) is obtained. Given that (14) and (31) only differ in labeling of the underlying graph it is anticipated that the GP based evolutionary processes that yield (14) and (31) would be similar. This anticipation is born out, but there are differences in the evolutionary processes. One significant difference is that the chromosome in (31) is evolved in a smaller number of generations than the one found in (14). There is nothing that is obvious about the rule set or input-output data based used for both chromosomes that would favor one over the other. Experimentation seems to indicate the difference in the number of generations required is related to the seed of the random number generator.

Generation	Best Chromosome found in the Population for the Indicated Generation
1	*OR2 SUM_SIG2 AND3DELAY DIFF SUM_SIG2 SUM_SIG123*
8	*OR2 AND3DELAY DIFF SUM_SIG3 SUM_SIG123 AND3DELAY H_2 DIFF MAX_SIG2 MAX_SIG123*
16	*OR2 AND3DELAY OR3 H_2 SUM_SIG2 SUM_SIG3 H_2 DIFF MAX_SIG2 MAX_SIG123 OR3DELAY H_2 OR3 H_2 SUM_SIG2 SUM_SIG3 H_2 DIFF MAX_SIG2 MAX_SIG123*
30	*OR2 OR3DELAY AND3 H_2 MAX_SIG123 H_3 MAX_SIG2 H_2 DIFF MAX_SIG2 MAX_SIG123 AND3DELAY AND3 H_3 SUM_SIG2 H_1 SUM_SIG123 H_2 DIFF SUM_SIG2 SUM_SIG3*
46	*OR2 OR3DELAY AND3 H_1 SUM_SIG123 H_3 MAX_SIG2 H_2 DIFF MAX_SIG2 MAX_SIG123 OR3DELAY AND3 H_1 SUM_SIG123 H_3 MAX_SIG3 H_2 DIFF MAX_SIG3 MAX_SIG123*

Table 4: Evolution of the DL given in (31).

Just as with the example in Table 3 the best chromosome of the first generation has an *OR2* at the end, but is otherwise too short and far removed from the correct answer. By the eighth generation the *"H_2 DIFF MAX_SIG2 MAX_SIG123"* structure has emerged. The best chromosome of the 16th generation preserves the best features of previous generations and also makes use of an *OR3DELAY*, but it still has many defects. For all generations after the 26th generation the elite chromosome has two subgraphs that take the form found in closed form in (26). The elite chromosome of the 30th generation has many correct labels and incorrect ones. It illustrates how evolution can fluctuate from generation to generation producing individuals of higher fitness, but departing significantly from the true DL in form. Finally in generation 46 the GP converges having produced the correct DL design.

6. SUMMARY AND CONCLUSIONS

Genetic program (GP) based data mining has proven effective for reverse engineering the complex digital logic underlying sensor devices (SDs) when the original design specifications for these devices are unavailable and invasive study of the systems is impossible.

The database that was subjected to data mining consisted of known input to the digital logic (DL), the associated measured output and a set of rules provided by experts relating to their assumptions about the digital logic. It is found that having a set of expert rules in the database is essential; the measured output of the digital logic is rarely sufficient to uniquely reverse engineer the design.

Explicit formulations of fitness functions and rules are given. Closed-form results for a class of optimal solutions to the rule fitness are shown to be effective in understanding required input-output data base properties. The closed-form results are also extremely useful in understanding the effect of rule or input-output database uncertainty on the final solution evolved by the GP.

Experimental observation and theoretical analysis of the effects of uncertainty show that even when there is a significant reduction in the quality of rule or input-output measurement information: the DL map evolved by the GP will still carry enough information for the design of signals with specific properties. The creation of these signals is considered of greater importance than having the exact DL design for the SD. So the results of the uncertainty analysis related to the closed form expressions are deemed of great importance.

Significant experimental results and theoretical analysis are provided to support the above conclusions. Reverse engineered DL examples are provided along with details of their evolutionary history and the implications of uncertainty.

ACKNOWLEDGEMENTS

This work was sponsored by the Office of Naval Research. The authors would also like to acknowledge Dr. Jeffrey Heyer for useful conversations about and support for an unrelated application of genetic algorithms.

REFERENCES

1. James F. Smith, III and ThanhVu H. Nguyen; "Data mining based automated reverse engineering and defect discovery", *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005*, B. Dasarathy, Vol. 5812, pp. 232-242, SPIE Proceedings, Orlando, 2005.
2. J.R., Koza, F.H. Bennett III, D. Andre, and M.A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*. Chapter 2, Morgan Kaufmann Publishers, San Francisco, 1999.
3. James F. Smith, III, "Fuzzy logic resource manager: real-time adaptation and self-organization", *Signal Processing, Sensor Fusion, and Target Recognition XIII*, I. Kadar, Vol. 5429, pp. 77-88, SPIE Proceedings, Orlando, 2004.
4. James F. Smith, III, "Fuzzy logic resource manager: decision tree topology, combined admissible regions and the self-morphing property", *Signal Processing, Sensor Fusion, and Target Recognition XII*, I. Kadar, pp. 104-114, SPIE Proceedings, Orlando, 2003.
5. James F. Smith, III, "Fuzzy Logic Resource Manager: Evolving Fuzzy Decision Tree Structure that Adapts in Real-Time," *Proceedings of the International Society of Information Fusion 2003*, X. Wang, pp. 838-845, International Society of Information Fusion Press, Cairns, Australia, 2003,
6. J.P. Bigus, *Data Mining with Neural Nets*, Chapter 1, McGraw-Hill, New York, 1996.