ELSEVIER

Information Sciences 120 (1999) 223–237

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A genetic algorithm approach to measurement prescription in fault diagnosis ☆

Benjamin Han, Shie-Jue Lee *

Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan, ROC

Received 3 April 1998; received in revised form 11 June 1999; accepted 11 September 1999

Abstract

To fully discriminate among all possible diagnoses in a fault diagnosis task, one needs to take measurements from the system being diagnosed. The primary effects of taking one measurement in diagnosis based on first principles were presented in A. Reiter [Artificial Intelligence (32) (1987) 57–95] and a more detailed, formal account was given in A. Hou [Artificial Intelligence (65) (1994) 281–328]. However, the order in which measurements are to be taken is an issue. We propose a genetic algorithm to determine a good measurement order for a diagnosis task. The method applies operators such as selection, crossover, and mutation to evolve an initial population of measurement sequences. The quality of a measurement sequence is evaluated based on the cost taken for the measurement sequence to find the final diagnosis. Experiments on testing circuits have shown that the quality of measurement sequences is greatly improved after evolution.

Keywords: Model-based diagnosis; First principles; Measurements; Measurement ordering; Genetic operators

1. Introduction

In a nontrivial fault diagnosis task using model-based approaches, such as Reiter's theory of diagnosis from first principles [12,16], one is always

☆ Partially supported by National Science Council under grant NSC-83-0408-E-110-004.

* Corresponding author.
E-mail address: leesj@ee.nsysu.edu.tw (S.-J. Lee)

0020-0255/99/$ - see front matter
PII: S0020-0255(99)00071-7

confronted with the problem of discriminating among competing diagnoses obtained. To narrow down the candidate diagnoses, measurements have to be taken from the system being diagnosed. A measurement is a system observation which probes the output of a single component within the system. Assuming that measurements will not cause any unwanted side effect, the result of a measurement is used to compute new diagnoses which confirm the observation and the measured result. Subsequent measurements can be taken in order to settle down to only one possible diagnosis.

Deriving new diagnoses based on the result of a measurement has been formalized in the work of Reiter [16] and Hou [14]. However, the measurement ordering problem, i.e., determining the best order in which measurements are to be taken, has received very little investigation so far. The one-step lookahead method proposed in [7,8] is one try which selects the best next measurement by minimizing the total number of measurements required. The method demands a simulation of probing each of the possible measurements with each of the possible outcomes. Minimal conflict sets under each measurement with each possible outcome are computed, and the entropy of the whole system being diagnosed under different measurements is calculated. The measurement with which the entropy of the system is minimal is then selected as the next measurement. This method is expensive in computation power as the number of possible measurements increases.

The measurement ordering problem is well suited to a genetic algorithms approach since it is an NP-hard problem and does not, in practice, require a perfect solution. Genetic algorithms (GAs) have been developed and studied [10,13] and have been applied to many fields with success [1,10,15]. GAs operate on a set of strings instead of only one, so they can be more robust. GAs use stochastic operators instead of deterministic ones, so they can be more efficient. Furthermore, the genetic operators used can be easily implemented. We propose a genetic algorithm to help find the best order of measurements to be taken in a diagnosis task for digital circuits. The quality of a measurement sequence is evaluated based on the cost taken for the measurement sequence to find the final diagnosis. Our method applies genetic operators such as selection, crossover, and mutation to evolve an initial population of measurement sequences. Experiments with testing circuits have shown that our method is effective; the quality of measurement sequences is greatly improved after evolution for each testing circuit.

The rest of the paper is organized as follows. We first briefly review the theory of diagnosis from first principles in Section 2. Then in Section 3 we describe in detail the genetic algorithm used for the measurement ordering problem. Section 4 presents the results of experiments using the proposed method to diagnose some testing circuits. Finally, Section 5 concludes our work.

2. Diagnosis from first principles

Suppose we are given an observation of a system which conflicts with the way the system is meant to behave, the fault diagnosis task is to pinpoint the possible diagnoses, i.e., the possible sets of faulty components, that cause the misbehavior of the system. Reiter [16] has built and formalized the major theorems for diagnosis from first principles upon the work of de Kleer [6] and Genesereth [9].

A *system* is a pair (SD, COMP) where SD is the system description and COMP is a finite set of constants denoting the collection of components of the system. The system description is a set of first-order logic sentences [3] describing:

1. The functionality of a component within the system; e.g., let A be an AND gate with two inputs, the sentence describing A is

$$\neg \mathrm{AB}(A) \rightarrow \mathrm{out}\ (A) = \mathrm{AND}(\mathrm{in1}(A)\mathrm{in2}(A)),$$

   where the predicate AB stands for *abnormal*, which is true if and only if gate A is malfunctioning. The sentence states that if A is normal then the output of A is the conjunction of A's two inputs, $\mathrm{in1}(A)$ and $\mathrm{in2}(A)$.
2. The connections between the components of the system; e.g., $\mathrm{out}(A) = \mathrm{in1}(B)$ means that the output of component A is connected with the first input of component B.

Real world diagnostic settings involve observations. Observations allow us to determine whether something is wrong and hence whether a diagnosis is called for. An *observation* of a system is a finite set of first-order sentences describing the values observed at input or output pins of gates in a system under investigation. We shall write (SD, COMP, OBS) for a system (SD, COMP) with observation OBS.

Definition 1. Δ is a *diagnosis* for (SD, COMP, OBS) if and only if $\Delta \subseteq \mathrm{COMP}$ is a minimal set such that

$$\mathrm{SD} \cup \mathrm{OBS} \cup \{\neg \mathrm{AB}(c) \mid c \in (\mathrm{COMP}\text{-}\Delta)\}$$

is consistent.

Note that we need to do consistency checking in order to confirm that a set is a diagnosis or not. The consistency checking can be done with a propositional logic theorem prover [2]. For an observation OBS of a system, there may be many possible diagnoses to account for the system's misbehavior. To refine further the set of diagnoses, one needs to take measurements from the system. A *measurement* is just an additional observation which probes the output of a single system component. Let Π be the result of a measurement. Each diagnosis either predicts Π or $\neg\Pi$.

Proposition 1. *A diagnosis Δ for (SD, COMP, OBS) predicts Π if and only if*

$$\mathrm{SD} \cup \mathrm{OBS} \cup \{\neg \mathrm{AB}(c) \mid c \in (\mathrm{COMP}\text{-}\Delta)\} \models \Pi$$

where $\models$ stands for logical consequence.

Note that with Π, the observation is expanded to $\mathrm{OBS} \cup \Pi$. The original set of diagnoses can then be refined as stated in the following theorem.

Theorem 1. *Suppose we take a measurement Π for (SD, COMP, OBS).*

1. *Every diagnosis for (SD, COMP, OBS) which predicts Π is also a diagnosis for (SD, COMP, OBS $\cup$ Π);*
2. *No diagnosis for (SD, COMP, OBS) which predicts $\neg\Pi$ is a diagnosis for (SD, COMP, OBS $\cup$ Π);*
3. *Any diagnosis for (SD, COMP, OBS $\cup$ Π) which is not a diagnosis for (SD, COMP, OBS) is a strict superset of some diagnosis for (SD, COMP, OBS) which predicts $\neg\Pi$.*

Apparently, deriving the set of new diagnoses from the set of old diagnoses and Π involves a lot of consistency checks. If we come to one possible diagnosis for the system being diagnosed, then we stop. However, many measurements may need to be taken before the number of possible diagnoses is reduced to one. Also, we may have many candidate measurements each time. The problem arises: what is the best order of measurements to be taken to minimize the total cost required in the whole diagnosis process? In fact, the derivation of possible diagnoses is closely related to the order of measurements taken.

Consider the circuit in Fig. 1 which contains 8 gates, 6 AND gates and 2 OR gates. The values on the left are the inputs and the value on the right is the output of the circuit, and these values constitute the original observation, OBS.

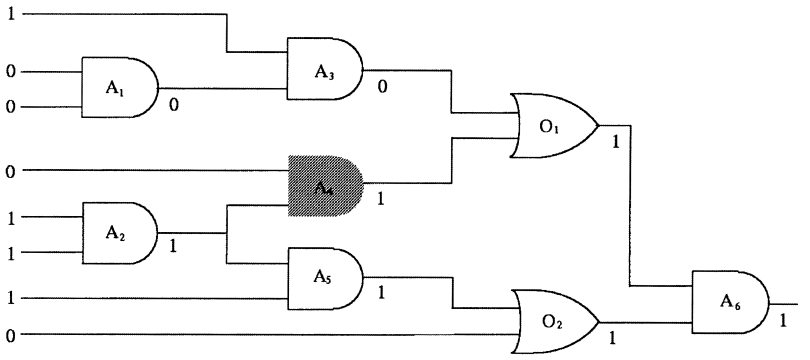

Fig. 1. An example circuit.

In the figure, seven more values are shown which indicate the results of seven possible measurements. For simplicity, suppose gate A_4 is out of order. From OBS, we deduce the set, D_0, of possible diagnoses to be

$$D_0 = \{\{A_1\}, \{A_3\}, \{A_4\}, \{A_6\}, \{O_1\}\}.$$

Suppose we have two measurement sequences:

$$S_1 = \{\text{out}(A_4), \text{out}(A_3), \text{out}(O_1), \text{out}(O_2), \text{out}(A_2), \text{out}(A_5), \text{out}(A_1)\},$$
$$S_2 = \{\text{out}(O_2), \text{out}(O_1), \text{out}(A_3), \text{out}(A_2), \text{out}(A_4), \text{out}(A_1), \text{out}(A_5)\}.$$

If we use S_1, we take the first measurement and get $\text{out}(A_4) = 1$ then we deduce $D_1 = \{\{A_4\}\}$ and we stop. Now let us work on the case with S_2. We take the first measurement of S_2 and get $\text{out}(O_2) = 1$. This measurement does not bring anything useful and we have

$$D_1 = \{\{A_1\}, \{A_3\}, \{A_4\}, \{A_6\}, \{O_1\}\},$$

which is identical to D_0. Take the second measurement of S_2 and get $\text{out}(O_1) = 1$. We deduce

$$D_2 = \{\{A_1\}, \{A_3\}, \{A_4\}, \{O_1\}\}.$$

Take the third measurement of S_2 and get $\text{out}(A_3) = 0$. We deduce

$$D_3 = \{\{A_4\}, \{O_1\}\}.$$

The fourth measurement of S_2 contributes nothing and we have $D_4 = D_3$. Take the fifth measurement of S_2 and get $\text{out}(A_4) = 1$. We deduce

$$D_5 = \{\{A_4\}\}.$$

Note that with S_1, we only need to take one measurement, but with S_2 we have to take five measurements, before deriving the final diagnosis. Obviously, the order in which measurements are to be taken affects the efficiency of a diagnosis system.

3. Measurement ordering by genetic algorithms

GAs were invented based on the inspirations from natural selection and evolution [10,13]. GAs manipulate a population of binary strings, named chromosomes. Each chromosome represents an encoded solution to the problem to be solved. During the evolution process, a chromosome is evaluated and the fitness value associated with the chromosome is computed. A fitness value is a positive number which reflects the quality of the corresponding chromosome, i.e., how good this particular solution is. The higher a fitness value, the better the corresponding chromosome (encoded solution). The chromosomes are selected from the population according to the fitness

distribution so that the one with higher fitness value is selected more frequently than that with lower fitness value. Moreover, two descendant chromosomes are generated by crossing over a pair of selected chromosomes, and new chromosomes are mutated according to a certain probability.

In a *generational GA* the evaluation process goes on until a whole new population is generated, and the old population is then replaced by the new one. In a *steady-state GA* the replacement occurs immediately after the evaluation of each single chromosome. In either case a stopping criteria is set, e.g., when the average fitness value goes beyond some threshold, and when the criteria are met the whole process stops.

3.1. Our method

For convenience, we call our method "a genetic algorithm for ordering measurements", abbreviated as GAOM. For a given circuit, a measurement sequence is called a *measurement request list (MRL)* which is an ordered set of all possible measurements that can be taken for the circuit. For example, S_1 and S_2 of Section 2 are two MRLs for Fig. 1. Apparently, there are $7! = 5040$ different MRLs for Fig. 1. Each measurement in an MRL is taken in order, and the result is used to update the set of possible diagnoses. This process goes on until only one diagnosis is left. It is obvious that every MRL is guaranteed to find the diagnosis. The question is: how far do we need to go through the measurements contained in an MRL to find the final diagnosis? Of course, the less the cost to find the diagnosis, the better the efficiency of the corresponding MRL.

A good MRL should keep the cost low for obtaining the final diagnosis. As described in Section 2, deriving new diagnoses involves a lot of consistency checks. Intuitively, a good MRL should minimize the number of measurements, n_M, to be taken and the number of consistency checks, n_P, involved in the process of obtaining possible diagnoses. We use the product of these two numbers as the performance index (PI) of an MRL as follows:

$$\mathrm{PI} = n_M \cdot n_P.$$

One MRL is said to be better than another MRL if the PI value of the former is smaller than that of the latter. Our goal here is to obtain the best MRL using GAOM.

Classic GAs use bit strings as the representation of candidate solutions. In GAOM a chromosome is represented simply by an MRL. Essentially, the goal of GAOM is to optimize the ordering of measurements to be taken. Bit-string representation is not natural for such order-based GAs since illegal strings may be produced after applying genetic operators [4]. Using MRLs as chromosomes facilitates the operation of the genetic operators used in GAOM.

GAOM is a steady-state genetic algorithm with a worst-replacement scheme and can be outlined as follows.

1. *Initialization*: Randomly obtain a population of MRLs.
2. *Initial evaluation*: Compute the PI value of each MRL within the population by performing diagnosis using the measurements contained in the corresponding MRL. Then calculate the fitness value of the MRL based on its PI value.
3. Test if one of the stopping criteria (time, fitness, etc.) holds. If yes, stop the procedure. For example, one would like to stop the procedure when the average PI value goes below some threshold or the final diagnosis is found.
4. *Genetic operator selection*: Select a genetic operator.
5. *Selection of MRLs*: Assume that the operator selected requires n MRLs as operands. Select n MRLs from the population.
6. *Applying the genetic operator*: Generate descendant chromosomes by applying the genetic operator selected in step 4 on the MRLs selected in step 5.
7. *Evaluation*: Select one of the new MRLs generated in step 6. Compute the PI value of this MRL by performing diagnosis using the measurements contained in this MRL. Then calculate the fitness value of the MRL based on its PI value.
8. *Updating the population*: If the new MRL evaluated in step 7 already exists in the population, keep this MRL in the population and update the values associated with it. Otherwise replace the worst MRL, i.e., the MRL with the largest PI value in the population, with the newly generated one (worst-replacement scheme).
9. *Updating the fitness values of the genetic operators*: Update the fitness values of all related genetic operators according to the performance of the new MRL.
10. Repeat steps 7–9 until all of the descendant MRLs generated in step 6 are evaluated and inserted into the population.
11. Repeat steps 3–10.

The following sub-sections give details on MRL evaluation, the genetic operators involved in GAOM, and dynamic update of the fitness values of the genetic operators.

3.2. MRL evaluation

An MRL is evaluated by performing diagnosis using it. The PI value is collected from the diagnosis process, and is used to compute the raw fitness value f_r of the MRL using the following formula:

$$f_r = \frac{P_{max}}{P} \cdot f_{r,min},$$

where P is the PI value of the MRL, $P_{\max}$ the maximal PI value of all the MRLs in the population, and $f_{\mathrm{r,min}}$ is a constant which is the reserved minimal raw fitness value of all the MRLs in the population. This conversion essentially inverts the direction of optimization from minimizing to maximizing, and at the same time preserves the ratio between the PI values of different MRLs.

After the raw fitness value of an MRL is determined, the fitness value f of the MRL is computed by scaling the raw fitness value using the following linear equation [10]:

$$f = a \cdot f_{\mathrm{r}} + b,$$

where the two coefficients a and b are obtained by solving the following equations:

$$\overline{f_{\mathrm{r}}} = a \cdot \overline{f_{\mathrm{r}}} + b,$$
$$c \cdot \overline{f_{\mathrm{r}}} = a \cdot f_{\mathrm{r,max}} + b,$$

where c is a constant typically in the range from 1.2 to 2, $f_{\mathrm{r,max}}$ the raw fitness value of the best MRL within the population, and $\overline{f_{\mathrm{r}}}$ is the average raw fitness of the population. Intuitively, computing a and b this way ensures that an average MRL contributes one expected offspring, while the most successful MRL contributes c expected offspring in the selection process. At the beginning of the evolution process, significant diversity might appear in the population, and the fitness scaling ensures that superior MRLs will not take over the whole population too early. On the other hand, when the population converges in late stages of the process, the fitness scaling guarantees that superior MRLs still have better chance to be selected, so that the evolution process will not become a random walk among those slightly inferior MRLs.

However, there are occasions when the whole population converges to some average raw fitness value and there still exist some bad MRLs whose raw fitness values are far below the average. In this case the fitness scaling will yield negative fitness values, which are unacceptable for the working of GAOM. To solve the difficulty, we recalculate a and b by solving the following equations:

$$\overline{f_{\mathrm{r}}} = a \cdot \overline{f_{\mathrm{r}}} + b,$$
$$f_{\min} = a \cdot f_{\mathrm{r,min}} + b,$$

where $f_{\min}$ is a given constant representing the reserved minimal fitness value.

3.3. Genetic operators

GAOM uses five operators: selection, crossover, mutation, fine-tune, and re-evaluation.

Selection. This operator applies the Roulette wheel selection method which is essentially a weighted random selection approach. Let the sum of the fitness

values of all chromosomes within the population be F. Then the probability that a chromosome with fitness value f is selected is f/F.

Crossover. Let S_i and S_j be two MRLs. Compute the average number of consistency checks associated with S_i and that with S_j, respectively. For each measurement, M, in S_i, if the number of consistency checks associated with M exceeds the average, collect M into an ordered set T_i, in which M_x is in front of M_y if and only if M_x is in front of M_y in S_i. Similarly, collect all such measurements in S_j into another ordered set T_j. Then the two descendants, S_i' and S_j', of S_i and S_j are

$$S_i' = T_i + (T_j - T_i) + O_j(S_i - (T_i + (T_j - T_i))),$$
$$S_j' = T_j + (T_i - T_j) + O_i(S_j - (T_j + (T_i - T_j))),$$

where for two ordered sets A and B, $A + B$ returns an ordered set appending B to the end of A, $A - B$ returns an ordered set containing the elements only belonging to A and in the order specified in A, and O_x reorders its argument according to the order specified in S_x.

Intuitively, if a measurement result agrees with all the old diagnoses, the measurement is useless since it does not bring any information and the set of new diagnoses is totally the same as the set of old diagnoses. In this case, the number of consistency checks is equal to the number of old diagnoses. On the other hand, if a measurement result disagrees with old diagnoses, the old diagnoses will be rejected and new diagnoses will be generated. The measurement is useful since it brings information and contributes to the refinement of diagnoses. In this case, consistency checking may be executed many times in the derivation of new diagnoses. Therefore, a useful measurement result which helps updating already-known diagnoses requires many calls of consistency checking. We use the average number of consistency checks associated with an MRL as the threshold of preference to the measurements in the MRL. We would like to probe such measurements which are potentially useful early in the diagnosis process.

Although the crossover operator moves those measurements in one MRL with the number of consistency checks exceeding the average to the front of the MRL, the relative order of these promising measurements is preserved. The remaining measurements in the MRL are rearranged according to their relative order specified in the other MRL. In this way, good orderings in different MRLs can be exchanged for better performance.

Consider two MRLs, S_1 and S_2:

$$S_1 = \{M_3, M_1, M_5, M_2, M_4, M_6, M_8, M_7\},$$
$$S_2 = \{M_7, M_3, M_8, M_5, M_4, M_2, M_1, M_6\}$$

each containing eight measurements. The number of consistency checks associated with each measurement is listed in Table 1. The average number of

Table 1
Two MRLs for crossover

S_1		S_2	
Measurement	Number of consistency checks	Measurement	Number of consistency checks
M_3	30	M_7	30
M_1	20	M_3	50
M_5	60	M_8	60
M_2	70	M_5	30
M_4	40	M_4	20
M_6	80	M_2	60
M_8	90	M_1	50
M_7	60	M_6	30

consistency checks for S_1 is 56.25, and for S_2 is 43.75. Thus we have $T_1 = \{M_5, M_2, M_6, M_8, M_7\}$ and $T_2 = \{M_3, M_8, M_2, M_1\}$. The two descendants, S_1' and S_2', after applying the crossover operator are:

$$S_1' = \{M_5, M_2, M_6, M_8, M_7, M_3, M_1, M_4\},$$
$$S_2' = \{M_3, M_8, M_2, M_1, M_5, M_6, M_7, M_4\}.$$

Mutation. This operator acts on a single MRL. Let S_i be an MRL. Randomly choose two measurements, M_1 and M_2, within S_i and invert the ordering of all measurements in between (inclusive) M_1 and M_2. The result is the descendant of S_i. This operator is essentially the inversion operator first introduced by Holland [13].

Fine-tune. This operator acts on a single MRL. Let S_i be an MRL. Compute the average number of consistency checks associated with S_i and collect all the measurements with the number of consistency checks exceeding the average into an ordered set T_i as depicted in the description of the crossover operator. Randomly reorder the elements of T_i, and denote the result as T_i'. Then the descendant S_i' is

$$S_i' = T_i' + \{S_i - T_i'\}.$$

The intuition of this operator is to move the promising measurements to the front of the MRL, and rearrange their relative order randomly in the hope that the new ordering of these promising measurements will improve the PI value of the MRL.

Generally speaking, the fine-tune operator achieves a different effect than the crossover operator. A crossover operator first lumps the promising measurements (those with the above-average number of consistency checking) into the front of an MRL, then appends it with the measurements taken from the other

MRL. The fine-tune operator only lumps the promising easurements to the front, without "borrowing" measurements from another MRL.

Intuitively the fine-tune operator improves an MRL by reorder itself, i.e., no information is drawn from another MRL, so that the promising measurements are considered first in the hope that fewer measurements are required to complete the process. The crossover operator, however, actually rebuilds two new MRLs by swapping the useful schema in them. The only case where the effect of the crossover operator is the same as that of the fine-tune operator is that the two MRLs involved in a crossover are exactly the same ones. Without the fine-tune operator it is far less likely that the promising measurements are lumped in the front while the other measurements maintain the same ordering in an MRL.

Re-evaluation. This operator simply returns the MRL it received but updates the values associated with the MRL. For real diagnosis tasks, even with the same circuit the fault patterns will not be the same all the time. If an MRL is associated with a high fitness value under some fault pattern, it is possible that it will receive a rather low fitness value under another fault pattern. Without the re-evaluation operator, this inaccurate association will never be updated, hence the association cannot reflect the real fault tendency adaptively anymore.

3.4. Adapting fitness values of genetic operators

It has been shown that the control parameters such as the size of the population, mutation and crossover rates, and the number of crossover points affect a GA's performance [18]. Although some of them can be preset according to the results of empirical studies [5,11], the various operator rates need to be adapted constantly in the course of the evolution process in order to reflect the change in the population and in the environment. In GAOM, we use credit propagation for updating the fitness values of all four operators based on Davis' work [4]. In short, each of the operators receives a reward if it yields a superior descendant than the replaced MRL. This reward is then distributed exponentially to the operator producing the parent of the MRL, the operator producing the parent of the parent of the MRL, etc. The exact number of generations considered backward, *parent length*, and the constant for multiplying the original reward, *credit constant*, are the parameters of this method. Note that the updating method used here is slightly different from Davis' approach in that we give reward to a particular operator based on the difference between the fitness value of the new MRL and that of the replaced MRL, instead of assigning reward to an MRL based on the difference between the fitness value of the new MRL and that of the best MRL in the population.

4. Experimental results

To test GAOM, we incorporate it into a diagnosis system for diagnosing faults in digital circuits that we developed based on first principles. Experiments were done on five testing circuits. These five circuits are described in Table 2. The second column gives the number of gates (components) contained in each circuit. The number of possible measurements and possible MRLs (search space) are listed in columns 3 and 4, espectively. Column 5 describes the function and the IC number of each circuit.

Note that in these experiments, all fault patterns were randomly generated according to a given error rate distribution of the components in each circuit. Also, the parent length for credit propagation is set to 5, the reserved minimal raw fitness value and the reserved minimal fitness value are both set to 10.0, the constant c used for scaling raw fitness values is 1.5, and the number of faulty components in all simulated fault patterns used is set to 2.

In each experiment, we first randomly generated an initial population of MRLs for each circuit. After the initialization phase, the initial population of MRLs is evolved using GAOM operators. We randomly generated a fault pattern for each MRL during the course of the evolution process. The spirit of these tests is to simulate the situation in which a diagnosis system deals with real-world problems, i.e., the fault patterns of the same circuit keep changing all the time. Table 3 gives the values of the parameters used in GAOM for each testing circuit.

We used GAOM to evolve an initial population of MRLs 10 times for each testing circuit. The results obtained in the 10 times are averaged and the averaged results: the initial PI value, the PI value after evolution, and the improvement rate for each circuit, are shown in Table 4. Note that the improvement rate is defined as (column 2−column 3)/(column 3) in the table. Clearly, the PI values, which are what GAOM tries to minimize, are all reduced considerably for all testing cases. The last column of Table 4 shows the per-

Table 2
Five testing circuits

Testing circuit	Number of gates	Number of possible M's	Number of possible MRLs	Description
c01	8	7	5040	Arbitrary circuit (Fig. 1)
c02	18	8	40320	BCD to decimal decoder (7442)
c03	19	14	87178291200	Lookahead carry generator (74182)
c04	16	8	40320	3–8 demultiplexer (74138)
c05	16	14	87178291200	Dual 4–1 data selectors (74153)

Table 3
The values of the parameters used for each testing circuit

Testing circuit	Population size	Number of evaluations	Initial operator fitness				Credit constant
			Crossover	Mutation	Fine-tune	Re-evaluation	
c01	20	100	0.55	0.05	0.25	0.15	0.001
c02	40	200	0.55	0.05	0.25	0.15	0.00001
c03	40	400	0.55	0.05	0.25	0.15	0.00001
c04	40	200	0.55	0.05	0.25	0.15	0.00001
c05	40	400	0.55	0.05	0.25	0.15	0.00001

Table 4
Improvement on PI values after evolution

Testing circuit	Initial PI	PI after evolution	Improvement (%)	Percentage of search space circuit
c01	534.10	216.67	59.434	1.984
c02	145797.90	24529.58	83.176	0.496
c03	606380.90	51719.73	91.471	4.588×10^{-7}
c04	46166.80	7602.59	83.532	0.496
c05	122290.90	12948.33	89.412	4.588×10^{-7}

centage of the search space (i.e., the number of possible MRLs) explored for each circuit. This equals the number of evaluations divided by the size of the search space, e.g., 100/5040 = 1.984% for c01. Fig. 2 shows how the PI value evolves as the number of evaluations increases for circuits c03 and c05. Similar curves are obtained for the other circuits.

5. Concluding remarks

We have proposed a measurement ordering strategy which can be incorporated into a diagnosis system for determining the best order of measurements to be taken in a diagnosis process. Specifically, we have proposed GAOM, a genetic algorithm for ordering measurements, and successfully integrated it into our diagnosis system developed for digital circuits based on first principles [14,16]. Experimental results on five testing circuits were presented and they showed that GAOM is effective in reducing the cost of a diagnosis task significantly.

The GA approach has a fundamental advantage that the whole population of promising MRLs keeps adapting itself with the real fault tendency exhibited in the target system. It is true because every MRL in the population is actually applied in a real diagnosis process, and its performance is used for its survival

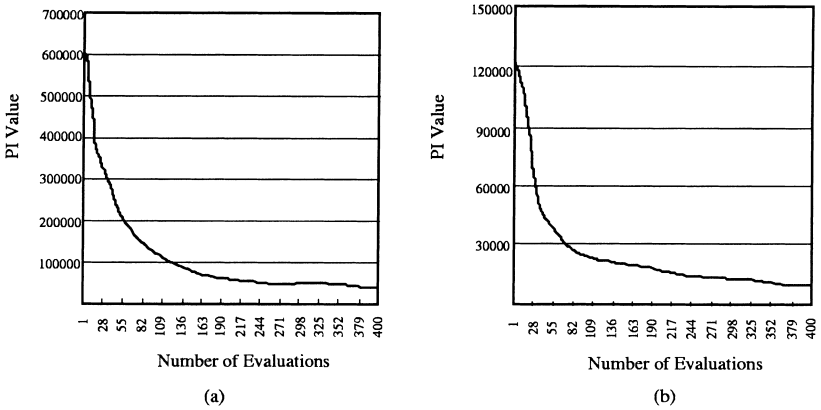

Fig. 2. PI value vs. number of evaluations for (a) c03 and (b) c05.

in the successive generations. In the first glance it seems to be expensive to obtain optimal orderings by actually applying the inferior MRLs in diagnosis processes. However, in a real-world setup the whole adaptation process occurs on-line, namely no off-line training is required. We can always start from a randomly generated population of MRLs and in time the system will adapt itself with the real fault tendency exhibited in the target system. Moreover, no presumption is made with the performance index we want to optimize. One can always redefine the PI to incorporate other cost factors and use the same architecture, to meet the requirements of any specific diagnosis task [17]. In any case, the average PI value of the whole population of MRLs can be lowered down by training the diagnosis system with fault patterns randomly generated or obtained from past experiences before any diagnosis process is engaged.

GAOM has another advantage that the computation overhead involved for evolving promising MRLs is negligible. The adaptation process takes place on-line, which means that the system could adapt itself with the fault tendency exhibited in the target system in real-time – no off-line training is necessary. Of course the initial population can be generated in a more intelligent way, other than the "random" one used in the experiments, so that the convergence could take place sooner. By contrast, the simulation of the probing for every possible measurement requires considerable amount of consistency checks in de Kleer's approach (each simulation actually is a partial derivation of diagnoses based on the conjectured probe outcome) – in the end the overhead may well surpass the effort the approach tries to save, especially when there are many possible measurements (and to make things worse there are possibly more than two possible outcomes for each possible probe in a target system other than the digital circuits studied here).

References

[1] L.B. Booker, D.E. Goldberg, J.H. Holland, Classifier systems and genetic algorithms, Artificial Intelligence 40 (1989) 235–282.
[2] M. Buro and H.K. Büning, Report on a SAT Competition, athematik/Informatik, Universität Paderborn, 1992.
[3] C. Chang, R. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.
[4] L. Davis et al., Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[5] K.A. De Jong, Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Thesis, Department of Computer and Communication Sciences, University of Michigan at Ann Arbor, 1975.
[6] J. de Kleer, Local Methods for Localizing Faults in Electronic Circuits, vol. 394, MIT AI Memo, Cambridge, MA, 1976.
[7] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artificial Intelligence 32 (1987) 97–130.
[8] J. de Kleer et al., One-step lookahead is pretty good, in: Hamsher, Console, de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, Los Altos, CA, 1992, pp. 138–142.
[9] M.R. Genesereth, The use of design descriptions in automated diagnosis, Artificial Intelligence 24 (1984) 411–436.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[11] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics 16 (1) (1986) 122–128.
[12] R. Greiner, B.A. Smith, R.W. Wilkerson, A correction to the algorithm in Reiter's theory of diagnosis, Artificial Intelligence 41 (1989/1990) 79–88.
[13] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[14] A. Hou, A theory of measurement in miagnosis from first principles, Artificial Intelligence 65 (1994) 281–328.
[15] S.K. Pal, P.P. Wang, Genetic algorithms for pattern recognition, CRC Press, Boca Raton, FL, 1996.
[16] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57–95.
[17] W.B. Rouse, Human problem solving performance in a fault diagnosis task, IEEE Transactions on Systems, Man, and Cybernetics 8 (4) (1978) 258–271.
[18] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, R. Das, A study of control parameters affecting online performance of genetic algorithms for function optimization, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1989, pp. 51–60.