Learning Multiple Fault Diagnosis

Yousri El Fattah
fattah@ics.uci.edu
Paul O'Rorke
ororke@ics.uci.edu

Technical Report 91–06

January 30, 1991

To appear in *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications*, February 24-28, 1991, Miami Beach, Florida.

This research was supported in part by National Science Foundation Grant Number IRI-8813048, McDonnell Douglas Corporation, Douglas Aircraft Company, and the University of California Microelectronics Innovation and Computer Research Opportunities Program.

Learning Multiple Fault Diagnosis

Yousri El Fattah (fattah@ics.uci.edu) Paul O'Rorke (fattah@ics.uci.edu)

Department of Information and Computer Science
University of California, Irvine, CA 92717

Abstract

This paper describes two methods for integrating model-based diagnosis (MBD) and explanation-based learning. The first method (EBL) uses a generate-test-debug paradigm, generating diagnostic hypotheses using learned associational rules that summarize model-based diagnostic experiences. This strategy is a form of "learning while doing" model-based troubleshooting and could be called "online learning." The second diagnosis and learning method described here (EBL-STATIC) involves "learning in advance." Learning begins in a training phase prior to performance or testing. Empirical results of computational experiments comparing the learning methods with MBD on two devices (the polybox and the binary full adder) are reported. For the same diagnostic performance, EBL-STATIC is several orders of magnitude faster than MBD while EBL can cause performance slow-down.

AI Topic: Machine Learning, Model-based diagnosis.

Domain Area: Digital systems diagnosis.

Language/Tool: PROLOG on Sun3/60 workstations.

Status: Implemented and tested.

Effort: 0.5 person year.

Impact: Diagnosis speedup learning.

1 INTRODUCTION

Two major approaches to diagnosis have been proposed. First generation expert systems used rules to encode associations between symptoms and diagnoses. Imperfections such as incorrectness or incompleteness in the rule-base encoding the diagnostic knowledge of these systems made them "fragile" or "brittle" [7]. Model-based reasoning was proposed as a way of overcoming these problems [2].

Researchers in diagnosis now believe that causal models and associational knowledge are complementary and they should be combined in systems that use each technique when it is most appropriate [11, 6]. In the work reported here, reasoning based on device models and first principles provides robust diagnostic performance. Associational rules provide efficient recognition of conflict sets and diagnoses directly from symptoms.

Learning methods are used to convert model-based diagnostic experience into associational rules. The learning methods, derived from research on explanation-based learning (EBL) [4, 12], are intended to improve the performance of the diagnostic system.

There is some controversy surrounding the question about whether EBL can improve the performance of model-based diagnostic systems. Davis [1] has argued that attempts to apply EBL to model-based diagnosis are misguided. He claims EBL

> *is simply not applicable in the current case.* Model-based systems do not deliberate in an unguided fashion and they do not deliberate before every action. The reasoning processes are sharply focused procedures typically realizable as computationally trivial graph traversal; hence they do not significantly affect speed (Except for finding multiple faults, which is inherently exponential, so nothing helps.)

Keller [10] has argued that diagnostic systems are not immune to improvement by EBL. We view this controversy as an empirical question and attempt to shed some light on it using computational experiments.

Previous works have considered the EBL to speed up model based diagnosis. See Resnick [8] and Zercher [13]. Those works operate under the *single-fault assumption.*

We consider a model based diagnosis (MBD) system for diagnosing *multiple faults.* Our goal is to determine the effectiveness of ways of employing EBL to speed up the MBD system's performance without sacrificing precision or completeness. We are interested in finding all minimal candidates (multiple as well as single faults) for a collection of observations. A candidate is a set of correctness assumptions that account for all occurring conflicts (contradictions between predictions and observations),i.e., the conjunctive retraction (suspension) of those assumptions would remove the contradictions. The question is: Can EBL speed up diagnosis while generating the same collection of minimal candidates as a non-learning MBD system would?

We explain in the next section three systems for diagnosing multiple faults. The first, called MBD, performs multiple fault diagnosis with no learning. The second,

called EBL, learns as it solves actual diagnostic problems and constructs macros for later use. The third, called EBL-STATIC, performs static learning before starting any actual diagnostic problem solving. We study the performance of the three systems on two simple circuits.

2 METHODS

We describe three systems for multiple fault diagnosis. The first system (MBD) performs no learning, but is the basis on which the other two systems (EBL) and (EBL-STATIC) are built. The latter two systems incorporate explanation-based learning in different forms.

2.1 MBD

The MBD system uses the theory given by Reiter [9] and emulates the GDE system of de Kleer and Williams [3]. The general system architecture is shown in figure 1. Our candidate generation procedure, *get-all-diagnoses*, is based on Reiter's HS-Tree algorithm for computing hitting sets [9].

2.2 EBL

In this system, EBL is used to construct macro-rules for various diagnostic tasks while solving actual problems using model-based machinery. Two types of macro rules are learnt by the system: conflict set recognition rules *c-rules* and diagnosis recognition rules *d-rules*. The overall EBL system operates according to the procedure EBL, depicted in figure 2. Diagnosis verification via constraint suspension as required in the algorithm is aimed at preserving correctness and completeness. The checking is needed because rules learned from previously diagnosed cases may suggest an overly-general diagnosis on the current problem. An example to demonstrate this point is this. Consider the polybox example (section 3.1) where EBL is given a problem where two conflicts are found, namely [m2,m1,a1] and [m3,m1,a1,a2]. EBL learns two c-rules corresponding to those conflicts. Next, EBL is given a problem where three conflicts are present; the previous two as well as the additional conflict: [m3,m2,a2]. Since c-rules exist, EBL will apply them, and since they succeed EBL will conclude that the set of all diagnoses is the same as the previous case, viz. [[a1], [a2,m2], [m1], [m3,m2]]. The candidates [a1], [m1] are overgeneral. Running the constraint suspension checking on either candidate will produce the additional conflict [m3,m2,a2]. The debugging module will then specialize the candidate [a1] to [a2,a1], [m2,a1], [m3,a1] and the candidate [m1] to [[a2,m1], [m2,m1], [m3,m1]].

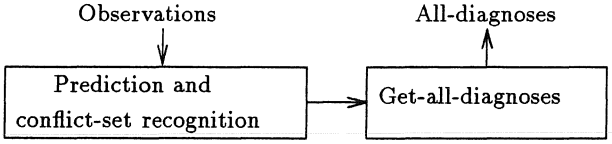

Figure 1: MBD Diagnosis System

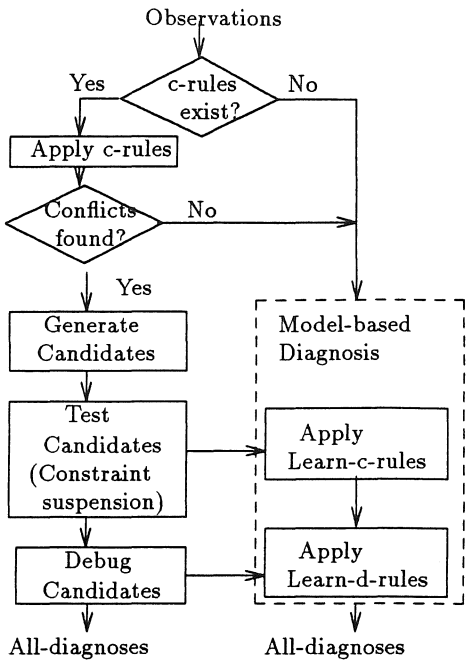

Figure 2: EBL Diagnosis System

2.3 EBL-STATIC

In this system, we need only to specify what the observable parameters are. We then assign symbolic values to those parameters and propagate the constraints in symbolic form. Once a complete set of c-rules is statically learnt, the system during actual diagnosis will only use those rules to determine all potential conflicts. Based on the conflicts found, the diagnoses are determined via a d-rule, if one succeeds. Otherwise, the procedure *learn-d-*

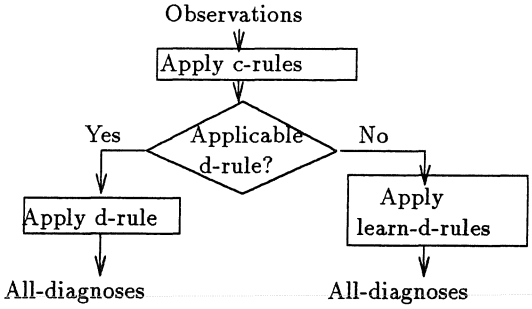

Figure 3: EBL-STATIC Diagnosis System

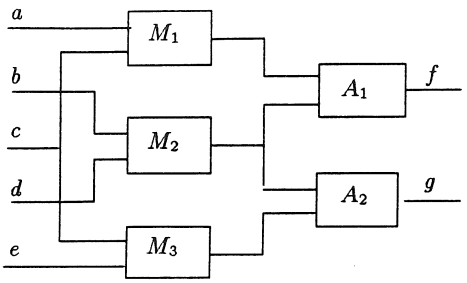

Figure 4: The Polybox Circuit

```
conflict_set([m2,m1,a1],1) :-
    obs(a,_a), obs(c,_c), obs(b,_b), obs(d,_d),
    obs(f,_f), diff(_f,_a*_c+_b*_d).
conflict_set([m3,m2,a2],2) :-
    obs(b,_b), obs(d,_d), obs(c,_c), obs(e,_e),
    obs(g,_g), diff(_g,_b*_d+_c*_e).
conflict_set([m3,m1,a1,a2],3) :-
    obs(a,_a), obs(f,_f), obs(c,_c), obs(e,_e),
    obs(g,_g), diff(_g,_f-_a*_c+_c*_e).
```

Figure 5: Complete set of c-rules for the polybox circuit

rule is applied.

Compared to the EBL system, there is no constraint suspension checking and learning is divided into static and dynamic parts. Learning c-rules is static, while d-rules are learnt dynamically.

The EBL-STATIC system architecture is shown in figure 3.

3 Examples

3.1 Polybox

Consider the polybox circuit depicted in figure 4. For the set of observable parameters: a, b, c, d, e, f, g, the EBL-STATIC system will learn statically the complete set of c-rules shown in figure 5. The first rule simply says that if the output of the first adder is different from the value of a times the value of c plus the value of b times the value of d then $a1$, $m1$, and $m2$ are suspect. The cpu time for EBL-STATIC-learn-c-rules is 1.44 seconds. In a set of 100 problems, the EBL system was able to learn

```
d_table([1,2],[[a2,a1],[a2,m1],[m2],[m3,a1],[m3,m1]]).
d_table([2,3],[[a1,m2],[a2],[m1,m2],[m3]]).
d_table([1,2,3],[[a2,a1],[a2,m1],[a2,m2],[m2,a1],
    [m2,m1],[m3,a1],[m3,m1],[m3,m2]]).
d_table([1,3],[[a1],[a2,m2],[m1],[m3,m2]]).
```

Figure 6: Complete set of d-rules for the polybox circuit

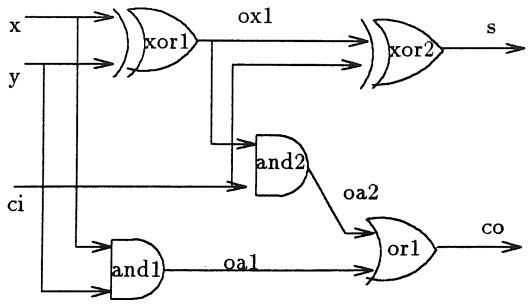

Figure 7: Binary Full Adder

```
conflict_set([xor1,xor2],1) :-
    obs(x,_x), obs(y,_y), obs(ci,_ci), xor(_x,_y,A),
    xor(A,_ci,B), obs(s,_s), diff(_s,B).
conflict_set([and1,xor1,and2,or1],2) :-
    obs(ci,_ci), obs(x,_x), obs(y,_y), and(_x,_y,A),
    xor(_x,_y,B), and(B,_ci,C), or(C,A,D),
    obs(co,_co), diff(_co,D).
conflict_set([and1,xor2,and2,or1],3) :-
    obs(s,_s), obs(ci,_ci), obs(x,_x), obs(y,_y),
    and(_x,_y,A), xor(B,_ci,_s), and(B,_ci,C),
    or(C,A,D), obs(co,_co), diff(_co,D).
conflict_set([and1,and2,or1],4) :-
    obs(ci,0), obs(x,_x), obs(y,_y), and(_x,_y,A),
    obs(co,_co), diff(_co,A).
conflict_set([xor1,and2,or1],5) :-
    obs(x,_x), obs(y,_y), obs(ci,1),
    diff(_x,_y), obs(co,_co), diff(_co,1).
conflict_set([xor2,and2,or1],6) :-
    obs(s,0), obs(ci,1), obs(co,_co), diff(_co,1).
conflict_set([and1,or1],7) :-
    obs(x,1), obs(y,1), obs(co,_co), diff(_co,1).
```

Figure 8: Complete set of c-rules for the full adder circuit

all three c-rules. For the same set of 100 problems, both EBL and EBL-STATIC learn dynamically the same set of d-rules. The d-rules are represented as table entries, indexed by the collection of conflict set labels. The d-table is shown in figure 6. The first rule in figure 6 says that if the conflict sets are those labeled 1 and 2, namely [m2,m1,a1] and [m3,m2,a2] then the set of all diagnoses is [[a2,a1],[a2,m1],[m2],[m3,a1],[m3,m1]].

3.2 Binary Full Adder

Consider the 1-bit binary full adder in figure 7.

For the collection of observable parameters: x, y, ci, s, co, the EBL-STATIC system learns 7 c-rules in 1 second. The c-rules are shown in figure 8. For the same set of 100 problems, both EBL and EBL-STATIC learn dynamically the same set of d-rules. The d-rules are represented as table entries, indexed by the collection of conflict set labels. The d-table is shown in figure 9.

```
d_table([1],[[xor1],[xor2]]).
d_table([2,3],[[and1],[and2],[or1],[xor2,xor1]]).
d_table([6,5],[[and2],[or1],[xor1,xor2]]).
d_table([1,3],[[and1,xor1],[and2,xor1],[or1,xor1],[xor2]]).
d_table([6,1],[[xor1,and2],[xor1,or1],[xor2]]).
d_table([7,1,6],[[xor1,and1],[xor1,or1],[xor2,and1],[xor2,or1]]).
d_table([5,1],[[xor1],[xor2,and2],[xor2,or1]]).
d_table([1,2],[[and1,xor2],[and2,xor2],[or1,xor2],[xor1]]).
d_table([4],[[and1],[and2],[or1]]).
d_table([4,1],[[xor1,and1],[xor1,and2],[xor1,or1],
    [xor2,and1],[xor2,and2],[xor2,or1]]).
d_table([7],[[and1],[or1]]).
```

Figure 9: Complete set of d-rules for the full adder circuit

4 Empirical Evaluation

We have carried out an empirical study to compare the performance of the three diagnosis systems: MBD, EBL, and EBL-STATIC. We studied the performance on the polybox and the binary full adder circuits.

A batch of 100 problems was generated. Diagnosis problems were created by randomly inducing single to triple faults in the circuit and then simulating the circuit behavior. Faults were assumed independent.

We fed the same problem batch to each of the three diagnosis systems. All three systems produced exactly the same output (all diagnoses) for the same problem, but took different cpu times. We plotted the cumulative cpu time for each diagnosis system for each circuit example. See figures 10, 11.

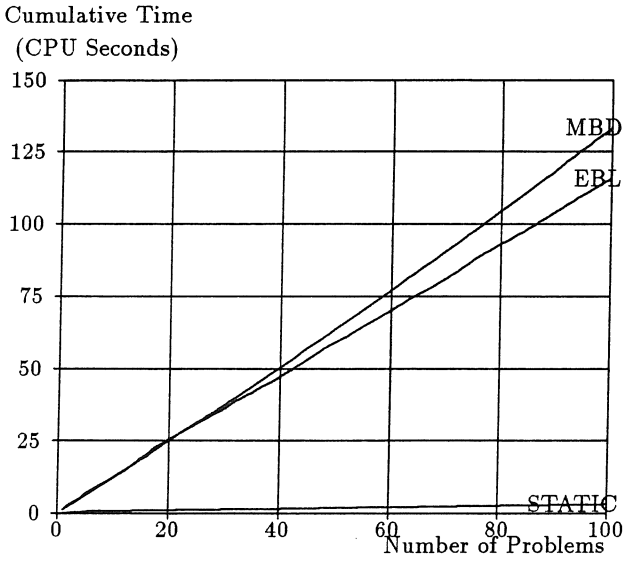

Figure 10: Polybox Empirical Results

In EBL, the net effect of speed-up from the c- and d-rule learning on one hand, and the constraint-suspension checking slow-down on the other hand, may depend on the size and the nature of the circuit. For the polybox circuit EBL contributed a net marginal speed-up, but for the full adder caused slow-down. This is because constraint propagation died quickly in the constraint suspension phase for the polybox circuit, but not so quickly for the full adder.

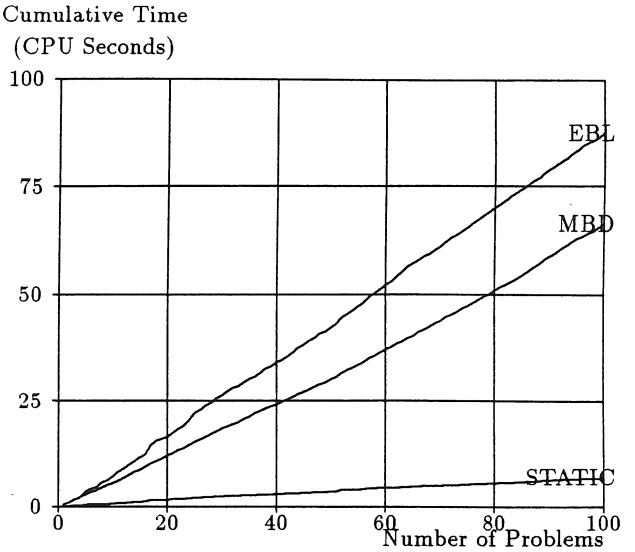

Figure 11: Binary full adder Empirical Results

For EBL-STATIC, conflict sets were generated on the basis of the pre-determined c-rules. The circuit model was no longer needed. The complete set of d-rules was learnt while solving a few problems. This meant that most of the time (96% for the polybox, and 89% for the full adder), all diagnoses were generated solely on the basis of associational rules. The time to match the rules conditions was extremely small in comparison with the time required for constraint propagation/suspension plus the hit set construction, as in the MBD or EBL cases. The effect of speed-up is very significant, as is evident from the performance plots of figures 10 and 11. On the average, EBL-STATIC was able to solve 10 to 30 problems in the same time required by MBD or EBL to solve one problem.

For the full adder, the speed-up effect of EBL-STATIC is less pronounced than in the case of the polybox. This can be explained by the increase in the number of c-rules, and the possibility of non-minimal conflict sets. The presence of non-minimal conflict sets slows down the system due to the computation wasted in matching their c-rules conditions, and then eliminating those sets later on before applying the hit set algorithm. For example, if c-rule number 5 in figure 8 succeeds then so will c-rule number 2. This means that there will be cases where the conflict sets:[and1,xor1,and2,or1], and [xor1,and2,or1] will be found. In such cases, the system will then have to eliminate the non-minimal set: [and1,xor1,and2,or1]; since [xor1,and2,or1] is a subset of that set. This means that the time required to match the conditions of the c-rule

number 2 was actually wasted.

5 CONCLUSION

This paper describes two methods for integrating model-based diagnosis and explanation-based learning, illustrating them in terms of the well-known "polybox" and full adder examples. The first method uses a generate-test-debug paradigm, generating diagnostic hypotheses using learned associational rules that summarize model-based diagnostic experiences. The rules associate violations of abstract constraints on observed inputs and outputs directly with multiple fault diagnoses. Unfortunately, the diagnoses suggested by the rules may be incorrect. If too few examples have been observed, the system may not have encountered relevant constraint violations and the rules may suggest diagnoses that are too general, missing components that may actually be faulted. Constraint suspension is used to test whether the proposed diagnoses actually explain the fault. If not, additional constraint violations occur leading immediately to further learning. This strategy is a form of "learning while doing" model-based troubleshooting and could be called "on-line learning."

The second diagnosis and learning method described here involves "learning in advance" since some learning occurs in a training phase prior to performance or testing. Constraint suspension testing can be avoided if all possible constraint violations are known in advance. In the training phase, an analysis of the model is done and abstract constraints between inputs and outputs are identified. Associations between combinations of constraint violations and diagnoses are made given examples of actual faults. The performance system avoids having to fall back on the model. This approach is similar to recent work on precomputing the kinds of information traditionally learned from examples in explanation-based learning. Our EBL-STATIC diagnosis system was inspired by Etzioni's work on STATIC, a "learning in advance" system for the planner PRODIGY. Etzioni [5] showed that analyses of domain descriptions can be used to achieve performance gains equivalent to EBL in the absence of examples. Our work indicates that, when it is feasible, explanation-based caching in advance is to be preferred over standard EBL in learning to diagnose multiple faults.

Acknowledgement. *This research was supported in part by National Science Foundation Grant Number IRI-8813048, McDonnell Douglas Corporation, Douglas Aircraft Company, and the University of California Microelectronics Innovation and Computer Research Opportunities Program.*

References

[1] R. Davis. Form and content in model based reasoning. In *Proceedings, AAAI-87 Workshop on Model Based Reasoning*, pages 11–27, 1987.

[2] R. Davis and W. Hamscher. Model-based reasoning: Troubleshooting. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, chapter 8, pages 297–346. Morgan Kaufmann, 1988.

[3] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[4] G. DeJong. An introduction to explanation-based learning. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, chapter 2, pages 45–81. Morgan Kaufmann, 1988.

[5] O. Etzioni. Why prodigy/ebl works. In *Proceedings, AAAI-90*, pages 916–922, 1990.

[6] W. Horn. *Causal AI Models— Steps Toward Applications.* Hemisphere Publishing Co., New York, 1990.

[7] D. Partridge. The scope and limitations of first generation expert systems. *Future Generation Computer Systems*, 3(1):1–10, 1987.

[8] P.Resnick. Generalizing on multiple grounds: Performance learning in model-based troubleshooting. Technical Report AI-TR 1052, MIT Artificial Intelligence Laboratory, February 1989.

[9] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[10] R.M.Keller. In defence of compilation. Technical Report FIA-90-06-25-1, NASA Ames Research Center, May 1990.

[11] R.G. Simmons. Combining associational and causal reasoning to solve interpretation and planning problems. Technical Report AI-TR 1048, MIT Artificial Intelligence Laboratory, September 1988.

[12] S.T. Kedar-Cabelli T.M. Mitchell, R.M. Keller. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[13] K. Zercher. Model-based learning of rules for error diagnosis. In W. Hoeppner, editor, *Proceedings GWAI-88*, pages 196–205. Springer Verlag, Berlin, 1988.