Trading off the costs of inference vs. probing in diagnosis

Johan de Kleer, Olivier Raiman
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto CA 94304 USA
Email: dekleer, raiman@parc.xerox.com

Abstract

This paper proposes a new algorithm which when provided the relative costs of computation vs. probing minimizes the total cost of diagnosis. During the diagnosis process the decision of whether to probe or to compute is dependent on the expected costs and benefits of each alternative. It is unlikely that we will be able to find general analytic and simple-to-compute models for the costs and benefits. Therefore, we base our algorithm on simple empirically derived models of costs and benefits. With these models, our algorithm operates by continuously choosing the optimum action to make next. This algorithm will not blow up on the rare pathological cases and will always (on average) find diagnoses at equal to or better cost than a conventional GDE/Sherlock. When the cost of probing is high, then our algorithm behaves exactly the same as GDE/Sherlock. When the cost of computation is high, the algorithm performs the diagnosis at far lower cost than GDE/Sherlock.

1 Introduction

A key goal of the diagnostic task is the identification of which component malfunctions are causing the undesirable symptoms. Recently there has been a considerable amount of exploration of the task of restoring the device to correct functioning at minimal cost [II; 15; 17]. Much of this research is aimed at minimizing the total cost of probing, repairing, and downtime. For example, the decision of whether to repair or to probe is dependent on the relative costs of probing vs. repairing. There has also been a considerable amount of effort into designing better diagnostic algorithms [7]. Although these algorithms have made model-based diagnosis far more practical, their design goals have been to produce faster algorithms without concern about the tradeoff of inference time vs. other costs. In this paper we explore the tradeoff between inference and probing costs and propose a new algorithm which when provided the relative costs of computation vs. probing minimizes the total cost of diagnosis.

There are three basic reasons why it is important to take into account computation time. First, no matter how good the algorithm, there are devices and inputs that require an inordinate amount of computation. When faced with these rare, but nevertheless very real, cases many algorithms fail to provide useful diagnoses within reasonable time. Second, the relative costs of computation and probing vary tremendously in applications. In one application, almost every signal can be measured immediately. In another application, measuring a signal may require sending a technician a mile down a mine shaft. On the other hand, some applications have enormous computational resources available, while in others the amount of computational resources available is minimal. Finally, even if computation could be regarded as "free," downtime to the customer often is not. The cost of the computation is often not the actual CPU charges, but the cost during which the customer's system is nonoperational.

2 The GDE/Sherlock Framework

We adopt the model-based diagnosis framework of GDE/Sherlock [4; 5]. A typical GDE [10] implementation consists of a constraint propagator (constraints are typically used to model the components) and an assumption-based truth maintenance system (the ATMS is used to record assumptions). GDE operates as follows:

1. Using the constraint propagator and the ATMS, identify all (preferably minimal) conflicts (a conflict is a set of components, at least one of which is faulted). The conflicts are represented as ATMS nogoods.

2. From the conflicts, construct all diagnoses.

3. Given all the diagnoses, determine the optimal probe.

The sequence of computation and probing can be illustrated by Fig. 1. Only by determining all the diagnoses first can GDE produce the optimal sequence of probes.

Sherlock [7] is a far more efficient diagnostic algorithm, upon which we base our new algorithm. The first two steps of the GDE algorithm are inherently exponential. We can remove much of those exponentials by observing that most diagnoses (the low probability ones) have an insignificant effect on probe selection and most conflicts are irrelevant for determining those diagnoses. By focusing the ATMS (an HTMS [9]) we avoid ATMS label explosion. Sherlock operates roughly as follows (for more details see [7]):

1. Perform a best-first search (using prior probabilities) to find a candidate diagnosis which does not subsume any of the currently known conflicts.

2. Use the constraint propagator and the HTMS to test whether the candidate diagnosis is consistent with the observations. If it is not, then find one conflict which subsumes it and add it to the current known conflict set, and return to step 1.

3. Save the new diagnosis, compute its relative posterior probability via Bayes Rule.

4. If the cutoff criteria is not me, then go to step 1.

5. If we have one diagnosis, we are done.

6. Perform the best probe, based on the diagnoses so far. Go to step 4.

In the situation (which we explore in this paper) where most diagnoses have similar probability, the probe-compute profile of Sherlock is no different than that, of GDE (Fig. 1). We propose that computation and probing be intermixed (illustrated in Fig. 2). Intermixing computation and probing will clearly increase the total amount of probing—probes will be selected with less available information about the space of possible diagnoses. What may not be as obvious is that this intermixing will decrease overall computation time. When the probabilities of diagnoses vary significantly, Sherlock will also intermix computation and probing because if all current diagnoses are eliminated, it will return to step 1. But this is ad hoc — our new algorithm will make the decision of whether to compute or probe on a principled basis.

In Sherlock, every candidate diagnosis has to be checked and eliminated (in step 1 or step 2). The fact that computation and probing are intermixed does not reduce the total number of candidates that have to be checked. Computation time is improved because candidates can be eliminated far more quickly if more information is known about the faulty device. A probe may yield a new conflict which can easily eliminate a large

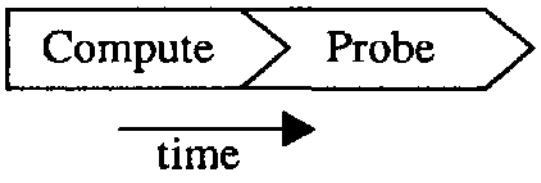

Figure 1: A complete computation phase followed by a probing phase.

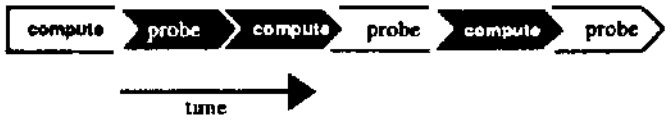

Figure 2: Intermixing computation and probing.

number of the candidates in step 1. If a particular candidate is not subsumed by a conflict, it still needs to be checked to see whether it gives rise to new conflicts in step 2. Every new probe made of the device, in essence, splits the device into two or more pieces. Thus the test of a new candidate is much less expensive. So the basic advantage of intermixing computation and probing is that the same computation task will become cheaper once more is known about the device.

Steps 1 through 3 above form an anytime algorithm—it can be stopped at any time and provide a current list of valid diagnoses in decreasing probability order and the accuracy of this diagnosis set increases monotonically. Steps 2 and 6 consume all the cost, so we will modify the algorithm to do a cost-benefit analysis at those points to decide whether to probe or to compute. The resulting algorithm will be described later.

In order to simplify the analysis we make a number of assumptions about the nature of the diagnosis task:

- Costs of all probes are constant and equal. This can be relaxed somewhat by introducing multi-step lookahead; but we do not explore this issue in this paper. If probe costs vary, then the decisions made by the algorithm may not be reasonable.

- The cardinality of the diagnoses is presumed to be constant. In other words, we do not expect the car-

dinality of the diagnoses being considered to change during the diagnosis session. (For example, we cannot handle the case where all singlefaults are eliminated, and then doublefaults are considered. Our proposed algorithm will suggest suboptimal probes in this case.)

3 Benefit of Knowing Another Candidate

GDE/Sherlock's algorithm searches the diagnostic space in a fixed order. It yields the diagnoses in decreasing probability. At any point the search can be suspended and restarted. The basic question we need to answer is the relative value of finding a next candidate. That value will be the number of fewer measurements the algorithm will have to make given that we know that candidate. This is not easy to estimate. GDE/Sherlock cannot easily provide the number of diagnoses that explain the symptoms before the cutoff criteria will be met[1], nor what their probabilities. Hence, we have no information about the number of diagnoses that might exist. To truly make the optimum next probe requires knowing all the diagnoses. Perhaps there is some analytic formulation of the number of candidates, but we have not been able to construct one. Instead we employ an heuristic formulation based on extensive experimentation with the GDE/Sherlock algorithm.

We have extensively analyzed a set of logic devices described in [2]. Those analyses provide considerable indirect information about computational cost. There is essentially very little benefit obtained from multiple step lookahead to determine the next probe [6]. In [8] we showed that it isn't even necessary to use GDE/Sherlock to obtain a reasonable algorithm. We presented a crude diagnostic algorithm based on random generate-and-test which works reasonably well using a small number of diagnoses to decide the next probe. That paper includes a number of graphs of how the number of measurements needed to diagnose drops with the number of diagnoses used to make the decision of which measurement to make next. GDE/Sherlock is, of course, a much computationally efficient algorithm (but much more complex). As the graphs only show the number of diagnoses, they are independent of the candidate generation algorithm and therefore apply to GDE[2] as well. The horizontal axis shows the number of diagnoses used to make the probe decision. The vertical axis shows the diagnostic cost (the average number of probes required to find every single-fault in the device—each with a number of sensitive vectors).

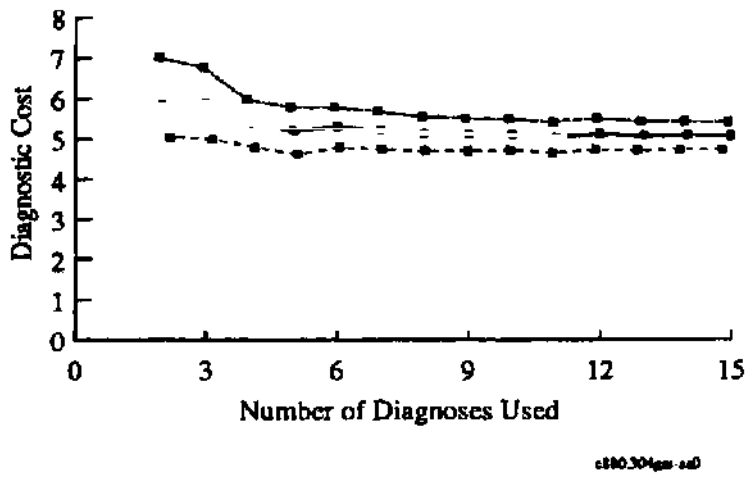

Figure 3: Scores of C880.

Fig. 3 shows the graphs for device c880. This device has 383 gates and 880 test points. As there are a large number of diagnoses, and only some of them are considered, different subsets yield different results. Samples were gathered by randomly choosing different subsets of the diagnoses. The middle graph on the figure shows the mean, the upper and lower ones show one standard deviation away. (These upper and lower curves are not used in this paper.) We have generated these curves for all of the devices in the test suite. These curves asymptotically approach $log_2 n$ where n is the number of components in the device. This is roughly what one would expect in the situation where components have two fault modes and we only consider single faults. If there are n components, there are $2n$ initial diagnoses. On average, half of those are eliminated by the initial symptom leaving n possible diagnoses. Using binary tests to discriminate among n alternatives takes $log2n$ probes on average. More generally, if components have / fault modes and the cardinality of the diagnoses are /, we expect the diagnostic cost in these graphs to approach /og2 ^ - The shape of the curve is harder to estimate. The case where $k < 3$ (k is the number of diagnoses considered) is difficult to analyze. For $k > 2$, the result can be modeled fairly consistently with (c is diagnostic cost) a decreasing exponential:

$$c = log_2 \frac{(fn)^l}{2}[1 + 2^{-k}]$$

Therefore the number of measurements saved by finding one more diagnosis (beyond k) is:

$$2^{-k-1} log_2 \frac{(fn)^l}{2}$$

There is one caveat worth mentioning in using this formula: it makes the presupposition that the algorithmic strategy does not change for the duration of the session. In actual fact, the algorithm may increase or decrease k

For all the examples discussed here failure mode probabilities are similar and small. The cutoff criteria used is met when the upper bound of posterior probability of the next candidate is significantly less than that of the best candidate. See summary for more discussion.

[2]They were, in fact, generated by GDE so if there is anything suspect about this, its not that they do not describe GDE correctly.

after subsequent probes. More likely it will decrease it because the value of higher *k's* drops as the number of diagnoses that remain drop. On the principle that the best prediction of a future decision is what we decide in the present, we use this formula.

4 Cost of Generating Another Candidate

The cost of generating another candidate is dependent on the kind of device, the size of the device, the inputs, etc. Again we are not able to come up with an analytic model of the computational cost of finding the next diagnosis. However, by observing the GDE/Sherlock algorithm in operation one sees some valuable properties. Let's first look at the case where we are not making probes. GDE/Sherlock generates diagnoses at a roughly constant rate. Intuitively, one can see how this arises from the best-first search. During the best-first search most diagnoses are eliminated by existing conflicts. The best-first search avoids these diagnoses at nearly negligible cost. The significant cost is incurred when a candidate is discovered which is not subsumed by any known conflict, but we cannot be sure is free of conflicts. In this case we have to re-invoke the ATMS. If no inconsistency is found, then we have a new diagnosis. If an inconsistency is found, then a conflict(s) is found and used to accelerate the best-first search. Rarely does a device have a large number of conflicts worth recording before finding another diagnosis.

Therefore, our algorithm predicts the computational time to produce the next candidate as equal to the average time to find the candidates it has so far. In order to account for the rare case where the cost of generating candidates varies greatly, it includes the time spent since the last diagnosis was found in the calculation. Therefore, if it spends a lot of computational effort to find the next diagnosis, then its prediction of when the next diagnosis will be found gets further and further out and if the time to find this diagnosis becomes too large the search is abandoned.

The introduction of probing lias little effect on the rate of candidate diagnosis production. This can be qualitatively understood as follows. The addition of a probe makes the test of any candidate about twice as fast. This is partially due to the fact that a probe may have introduced new general conflicts therefore making it easy for the best-first search to skip these candidates directly. In addition, when a candidate diagnosis must be tested against the observations (at step 2), the TMS operations are faster because the probe fixed a device quantity thereby splitting the device and thereby shortening ATMS labels. As the probes tend to be quite good [8] even when *k* is small, the probe reduces the number of diagnoses by approximately1/2.It reduces the number of diagnoses the algorithm has discovered by1/2, as well as the yet undiscovered ones. Hence, discovering new

Candidate	Probe	$\Delta t(s)$
1	1	.12
2	1	.12
3	1	.10
4	1	.13
5	2	.13
6	2	.17
7	3	.15

Table 1: Typical example of circuit c880 showing time to produce candidate does not vary significantly during diagnosis.

diagnoses has become twice as difficult. The two effects tend to cancel yielding an unchanged rate of generation of new diagnoses. Table 1 illustrates the kind of costs we see in diagnosing devices.

Although the rate of diagnosis production after a probe does not change, the number of diagnoses that remain to be found after a reasonable probe is halved. The probe has, in effect, eliminated these diagnoses for free.

5 A diagnostic algorithm which takes cost into account

In order to utilize our model our algorithm will always find at least *(k* = 3) candidates. This is both because our model of the benefit of a next diagnosis only applies if *k* > 2 and it allows for the computation a meaningful average cost of diagnosis discovery. The algorithm is provided two exchange rates: e_p which is the cost incurred per probe; e_9 which is the cost incurred per second of inference. The key step in the algorithm is to determine whether it is worthwhile to make another measurement. The implementation maintains a variable *t* (initially *t* = 0) which represents the time taken to find the current diagnoses. *M* will be the estimate of the number of probes necessary to isolate the diagnosis *(l o g* $_2$ *-(fn)/2*initially).

1. If they can be found before Sherlock's stopping criteria is met, find 3 diagnoses which explain all the current symptoms. Update *t* with the time to find these diagnoses.
2. If there is only one diagnosis, stop.
3. If the stopping criteria has been met, go to Step 7.
4. If *M* < *log2k,* set M to *log2k.* Compute the benefit *(B)* of finding another candidate:

$$B = M2^{-[k+1]}e_p.$$

5. If tei/k *B* (i.e., the cost of finding the next candidate outweighs the benefit), then go to Step 7.

e_s	e_p	GDE/Sherlock cost	New cost	$\overline{p}$
1	1000	5554	5554	5.6
1000	1	4705	3606	13.1

Table 2: Diagnostic Costs for C880 with different cost tradeoffs.

e_s	e_p	GDE/Sherlock cost	New cost	$\overline{p}$
1	1000	2052	2052	2.05
1000	1	2904	966	2.3

Table 3: Diagnostic Costs for C432 with different cost tradeoffs.

6. Allow the computation to run for Bk/ e*t* seconds or until another diagnosis is found. Update *t.* If a diagnosis is found, go to Step 3.

7. Perform the probe (using one-step lookahead and then entropy) which best discriminates among the current *k* diagnoses. Divide *M* by 2.

8. Remove the diagnoses eliminated by the probe result. Update *k* and *t* accordingly. Go to Step 1.

6 Experimental Results

We have run our algorithm on all the devices in our test suite. Table 2 lists the costs to diagnose a significant sample of c880's faults. The first line in the table makes computation nearly free and probing expensive. This is the presumption of GDE's architecture and the resulting costs for GDE and the new algorithm are the same. In the second entry, cost of computation has become extremely expensive. GDE which does not take cost of computation into account performs much worse, while the new algorithm performs significantly better. The last column of the table, *p,* is the average number of probes the new algorithm makes. Here we see the number of probes changes significantly depending on the cost tradeoff—exactly what one would expect. Table 3 shows the results for the smaller device C432 (432 test points, 160 components). The cost benefit of the new algorithm is clear for this device also.

7 Perspective from Decision Theory

Our algorithm can be viewed as a very simple, myopic, decision theoretic strategy. For clarity, the following analysis presumes binary valued variables. Let /(p, *s)* be our best estimate the cost to diagnose the device having made p probes and *s* candidates remaining. Our algorithm repeatedly makes the decision of whether to probe or to compute. The decision of whether to compute or to probe is determined with one-step lookahead. The decision to probe has cost e_p + *f(p* + 1 , s/2). The decision to compute is more complex to compute. We need to obtain the best estimate of cost if we continue to compute. By tracking the costs to compute past candidates we maintain the mean and standard deviation of times to compute candidates. Let *p(t)* be the area under a normal curve from present to t. Suppose we compute for time *t,* the expected cost is:

$$e_s \Delta t + p(t) f(p, s+1) + (1 - p(t)) f(p, s)$$

We choose *t* such this is minimized.

For simplicity let us only compute changes of /(p, *s).* Therefore, the cost of computation is the maximum of:

$$e_s \Delta t + p(t)[f(p, s) - f(p, s+1)].$$

Our algorithm estimates *f(p, s)* — *f(p,* s + 1) directly *(B* in our algorithm). To accurately determine the cost of probing we need to compute /(p,*s)* — /(p + !,§)• We estimate this to be negligible as two effects tend to cancel. Every good probe reduces the number of possible candidates by 1/2. Likewise every good probe reduces the number of known candidates by 1/2 (i.e., the s/2). Our preliminary experiments demonstrate it to be of little utility to compute the standard deviations and means of candidate computation costs. We will continue to use our simpler approximation for decision making. It is likely that if we encounter devices which have wide variance in candidate computation times that our approximation will become significantly suboptimal.

8 Summary

We have presented a simple algorithm which explicitly trades off computational cost against probing cost. By intermixing computation with probing, the new algorithm lowers the total cost of finding a diagnosis over the less flexible approach that is usually employed.

The overall result is only a small beginning of the research needed to understand the tradeofTs within diagnosis in order to develop minimal cost algorithms. There are a number of immediate simplifications made in this paper that need to be more closely analyzed. Some of them are:

1. Our algorithm always needs at least 3 diagnoses to function. However, if probing is completely free there is no necessity to ever construct any diagnoses at all as the faulty component can be determined directly. Our algorithm's architecture does not accommodate this possibility.

2. Sometimes all faults of the initial cardinality are eliminated. Our algorithm needs to be extended to take this possibility into account.

3. When the prior probabilities of component failure modes vary significantly, the estimate of the size of the diagnosis space *(M)* in the algorithm is too high, producing too high a benefit of future probes.

4. The algorithm should be extended to accommodate varying probe costs. For example, it would expend a significant amount of computation, if the next probe to perform was very expensive.

5. The model of the benefit of finding a new candidate was derived from the case where failure probabilities are nearly equal and with the standard ultimate GDE/Sherlock stopping criteria (probability cliff) for finding new candidates. The stopping criteria used clearly influences the benefit of finding a next candidate.

6. We observe that the algorithm usually lowers *k* during a diagnostic session. Which is of little surprise—as the number of diagnoses goes down, the advantage of finding more candidates becomes less. However, our benefit model was based on a constant *k.* This probably introduces a small overestimation in the algorithms calculation of the benefit of a next candidate.

7. We assume $f(p, s) - f(p+1, s/2)$ is negligible. This needs more careful analysis and experimentation.

Acknowledgments

We thank Daniel Bobrow, David Heckerrnan and Eric Horvitz for their helpful comments and insights on this paper.

References

[I] Bakker, R.R., and Bolurseau, M., Pragmatic reasoning in model-based diagnosis, in: *Proceedings ECAI-92,* Vienna (1992).

[2] Brglez, F. and Fujiwara, H., A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN, distributed on a tape to participants of the Special Session on ATPG and Fault Simulation, Int. Symposium on Circuits and Systems, June 1985; partially characterized in F. Brglez, P. Pownall, and R. Hum, Accelerated ATPG and fault grading via testability analysis, *Proc. IEEE Int. Symposium on Circuits and Systems,* (June, 1985) 695-698.

[3] Boddy, M. and Dean, T., *Solving time-dependent* planning problems, in: Proceedings *IJCAI-89,* Detroit, MI (1989) 979-984.

[4] de Kleer, J. and Williams, B.C., Diagnosing multiple faults, *Artificial Intelligence* 32 (1987) 97-130. Also in *Readings* in *NonMonolonic Reasoning,* edited by Matthew L. Ginsberg, (Morgan Kaufmann, 1987), 372-388. Also in [12].

[5] de Kleer, J. and Williams, B.C., Diagnosis with behavioral modes, in: *Proceedings IJCAI-89,* Detroit, MI (1989) 104-109. Also in [12].

[6] de Kleer, J., Raiman, O. and Shirley, M.H., One step lookahead is pretty good, in: [12] 138-142.

[7] de Kleer, J., Focusing on probable diagnoses, in: *Proceedings AAAI-91,* Anaheim, CA (1991) 842-848. Also in [12].

[S] de Kleer, J. and Raiman, O., How to diagnose well with very little information, in: *Proceedings of the Fourth International Workshop On Principles of Diagnosis,* Aberystwyth, Wales (1993) 160-165.

[9] de Kleer, J., A hybrid truth maintenance system, forthcoming.

[10] Forbus, K., and de Kleer, J., *Building Problem Solvers,* MIT Press, Cambridge, MA (1994).

[II] Friedrich, G., and Nejdl, W., Choosing observations and actions in model-based diagnosis-repair systems, in: *Proceedings KR-92,* Cambridge, MA (1992) 489-498.

[12] Hamscher, W., de Kleer, J., and Console, L. (eds), *Readings in Model-Based Diagnosis,* Morgan-Kaufmann, 1992.

[13] Ilorvitz, E.J., Breese, J., and Henrion, M., *Decision Theory in Expert Systems and Artificial Intelligence,* Technical Report KSL-88-13, Stanford University Knowledge Systems Laboratory, February, 1988.

[14] Horvitz, E.J., Reasoning under varying and uncertain resources constraints, in: *Proceedings AAAI-88,* Saint Paul, Minn. (1988) 111-116.

[15] Nejdl, W., and Bachmayer J., Diagnosis and repair iteration planning versus n-step lookahead planning, in: *Proceedings of the Fourth International Workshop On Principles of Diagnosis,* Aberystwyth, Wales (1993) 121-135.

[16] Russell, S. and Wefald, E., On optimal game-tree search using rational meta-reasoning, in: *Proceedings IJCAI-89,* Detroit, MI (1989) 334-340.

[17] Sun, Y., and Weld, D., A framework for model-*based repair, in: Proceedings AAAI-93, Washing*ton,D.C. (1993} 182-187.