# Investigating Cause for Failures in Discrimination Tree from Multiple Views

**Radhika Selvamani B.** **Deepak Khemani**

A.I. & D.B. Lab,
Department of Computer Science & Engineering,
I.I.T.Madras, India
bradhika@peacock.iitm.ernet.in, khemani@iitm.ac.in

## Abstract

Case Based Expert systems have been used to get expert solutions in domains with reoccurring similar problems by comparing the new problem with the past problems. Discrimination networks have been widely used for quick retrieval from a set of cases. But experts are able to exploit their experience, in ways other than best case retrieval. This paper discusses *cause induction in discrimination Tree* (CID-Tree). The motivation is to locate a set of problem cases in a discrimination tree organized on independent parameters, and thus identify possible causes for the problems. The domain of interest is manufacturing, and the goal is to try and identify process parameters that could be causes for certain kinds of defect. Often domain knowledge is distributed among different experts of different processes. This paper also supports the use of multiple views pertaining to discrimination trees structured on parameters from different shop floors. The intuition is that if defect cases cluster around certain parameters, then a cause for those defects could be identified.

# 1 Introduction

CBR has been widely used in many domains [Watson, 1997] like classification, interpretation, diagnosis, planning etc., mainly focusing in the representation and retrieval of past cases, which provide the required solutions, and augmenting the new cases into the case base. CIDTree is a new approach of using CBR in data mining.

The motivation of this paper is to investigate defects in manufacturing. A defect in a product could be caused by many reasons. When there are many parameters involved, even experts find it difficult to pinpoint the cause. The problem focuses on providing an aid to the experts in inferring the cause for the problem. The following example introduces the notion of a flawed node.

## 1.1 A Toy Example

Let us say one wants to select a toy for a birthday gift for a 6-8 year boy. The sales person or a recommender system suggests a GI Joe. But you feel you do not want a toy involving violence of any kind. The following toy organization as in figure 1 would help in this situation. Node B could identify with “Battle Related” toys. C could identify with “Construction Kits”. D could identify with “Dolls and House Keeping” toys. Given that you do not want B, you could choose C or D. The discrimination between the desired and the undesired toys would lie with the parents of B, C, and D that identifies with toys for 6-8 year group. We call node B *flawed*.

In the above example we identified the flawed node via a plausible recommender system dialogue. The user provided the key information. In the manufacturing scenario that we investigate, a flawed node is determined via an induction process for a set of defect cases. The flawed node is identified as a possible cause for the defects (figure 2).

## 1.2 Cognitive Perspective

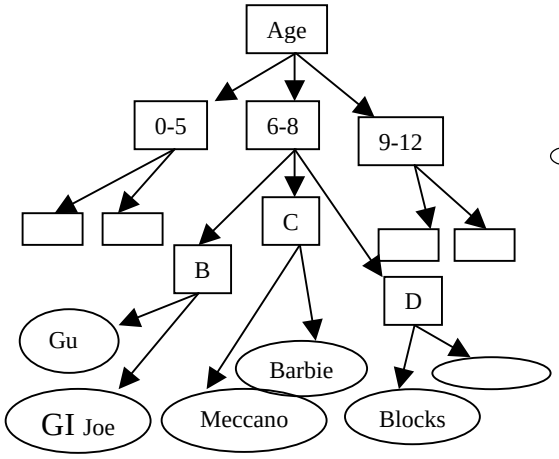


Figure 1. Toys Taxonomy

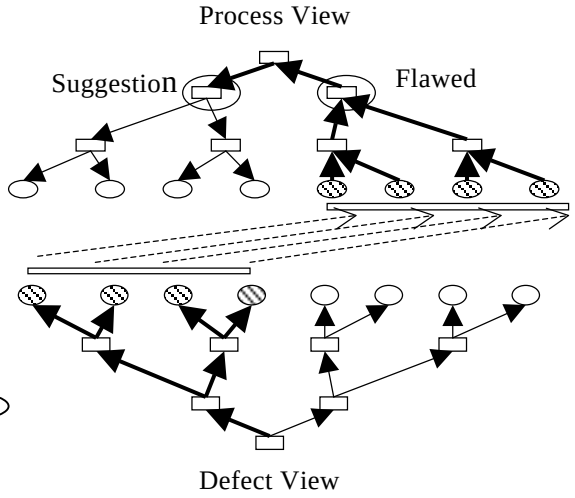


Figure 2. Overlapping Clusters of the Process View with those of the Defect View Identifying the Flawed Node

The cognitive model states that knowledge resides as chunks in memory. Knowledge residing in the memory is not abstract to start with, but can be eventually generalized from specific instances [Schank, 1999]. This nature of the memory organization, has been well implemented in hierarchical case organization techniques like shared feature networks and discrimination feature networks [Kolodner, 1993]. Specific instances are clustered according to their shared features. They are also discriminated based on their specific features that are not shared. While clustering is the primary focus in shared feature networks discrimination is primary in discrimination nets. Each hierarchy represents the level of generality derived over the cases. Organizing the cases based on their anomalies, retrieving similar cases and augmenting new cases through the levels of generality are the basic functionalities provided by the networks. These notions focused mainly on past case retrieval. Our goal is to find a parameter, whose value could discriminate between defects and normal artifacts. Hence we choose the discrimination tree for memory organization in this methodology. Then we address the problem of extracting causes for defects in a manufacturing scenario. This is done by discriminating separately on defects and operational parameters, and then looking for correlations in the two networks. CIDTree is a discrimination tree whose functionality is cause induction for the defects among the classified instances.

### 1.3 Data Mining

Class comparison technique (also known as discrimination) provides description comparing two or more collections of data [Han, 2001]. The target class is the one of interest, which is to be compared with other normal classes. The normal classes together form the contrasting class. A data deviation rule summarizes the behavior of a set of data, which deviates (substantially or to a certain degree) from the general trend or behavior of the majority of data. This paper deals with data deviation. It tries to find the most specific discrimination (deviation) between the target and the contrasting class satisfying the following constraints. There should be substantial match between the target and the contrasting classes. The matching classes should hold a reasonable number of instances.

### 1.4 Distributed Problem Solving Approach

A more powerful, extensible strategy for overcoming the inherent bounds of intelligence present in any finite AI (or natural) system is to put the system in a society of systems, so that it can draw on a diverse collection of expertise and capabilities in the same way that people overcome limitations of individuals by banding together into societies that are designed to accomplish what individuals cannot [Durfee, 1991]. Distributed problem solving approach has been used in CIDTree, by segregating the features into different domain dependent views.

The case organization and view formation are discussed in section 2 and 3. The CIDTree methodology has been explained in section 3, followed by the sections 4 through 7 describing the implementation in a manufacturing domain. Related work in data mining is compared in section 8 and the conclusions and future focus follows.

## 2 Case Organization in Views

In most domains like manufacturing, where there is a large flow of data, sets of attributes get values in different departments of the manufacturing process. There are three types of attributes prevailing in most domains namely, the design specification attributes, process attributes and the outcome attributes. Each of these sets of attributes forms a view. The process attributes may form more than one view depending on the different process stages pertaining to a case (figure 3). The domain experts have specific knowledge about their respective departments, but may lack details about other departments. So when a problem is encountered in such a domain, the experts usually tend to analyze the problem from their own perspective. They would provide statistics of different features from their departments, which they consider are relevant to the problem. The final solution is then obtained by comparing the views of different experts. Having specific views helps in reducing the complexity and increasing the correctness of the solution. It also provides modularity and extendibility in a very natural way. This approach enhances communication among departments, which combines the visions of different experts, resembling a discussion bench.

The case organization in the CIDTree is similar to a discrimination network [Kolodner, 1993] where the differences among the cases under a node are considered deviations. But the deviations are not used only for retrieval but also to organize cases similar in a certain view and arrive at the most discriminating feature causing the behavioral change. This is done by traversing from the leaf cases towards the common parent node

recording the deviations. This leads to discovery of the cause that discriminates the cluster of cases, good or bad, in a particular view with some measure of probability.

## 3 Phases in CIDTree

The CIDTree method has two phases. The construction of the discrimination tree based on defect clusters constitutes the first phase and the induction process the second phase. During the first phase, the defect view is used to cluster the data according to a specific set of behavioral patterns, the cause of which is to be traced. The classes formed under the defect view are then used in building up a discrimination tree for the remaining set of views. So there is a discrimination tree for each view as in figure 3. Investigating the possible causes for the defect by comparing the defect cluster with those formed in process views forms the second phase of the method.

### 3.1 Clustering

Clustering of data in this process is done in two steps. The parameters used in quality control form the basis of the quality discrimination tree which defines the defect view. Clustering of cases according to the defect or outcome, the cause of which is to be traced is the first step. The next step is automatic clustering of data in

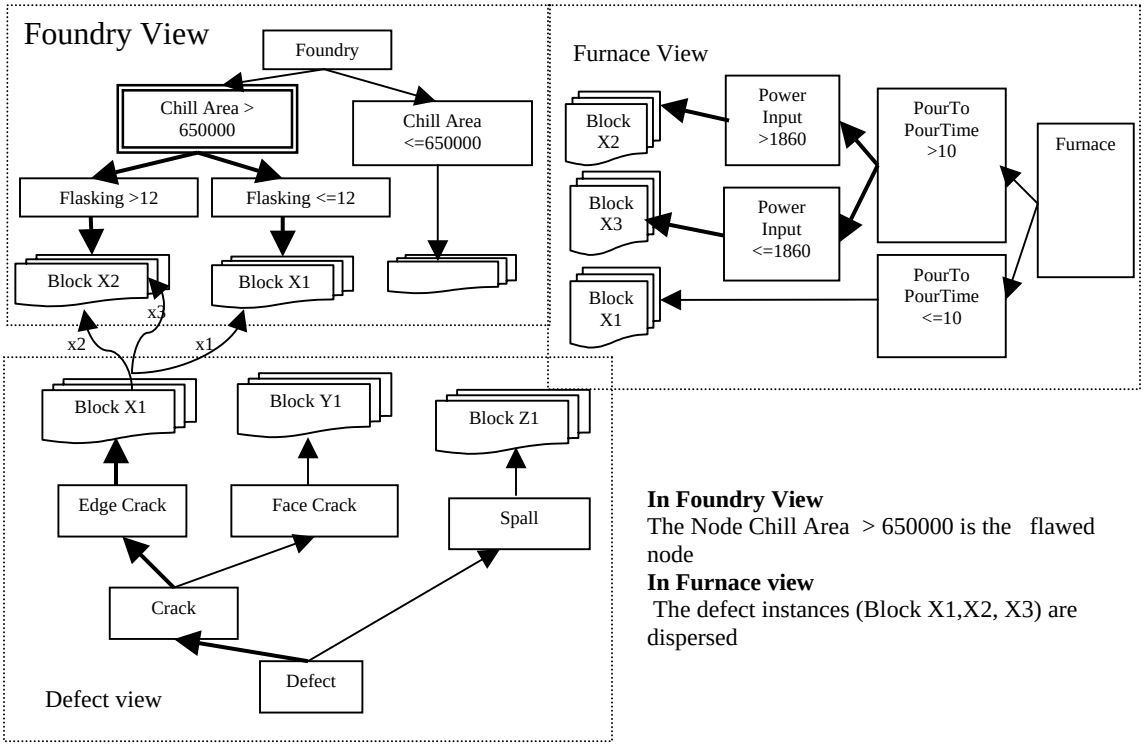


Figure 3. Interaction of Defect View with other Views

different process views. Structurally the defect view is similar to the other process views constituting the case description parameters, but functionally it is used to cluster cases with similar outcome in a top down fashion wherein the process views are used in a bottom up approach to trace the cause for failure and its probability. The defect view is thus to be included in all separate analysis done on each different process view, to create the defect clusters, which decides the discrimination tree.

### 3.2 Cause Induction

The induction process during the second phase is carried out as follows.

- In each view, traverse from the leaves towards the root till most of the defect cases come under a common node.
- The value of the common node suggests the reason for the outcome because they show the deviation from the rest of the cases. If the sibling of the common node contains considerable number of non-defective cases, the value of the sibling node may be suggested as a solution for the problem.
- This suggestion is for the defect in the last leaf node in the defect view (e.g., in figure 3 the suggestion is for edge cracks). By going higher levels in the defect view, the procedure may be repeated for more general defects (for set of all cracks in the example in figure 3).

## 4 Suitability of the Domain

Domains like manufacturing and diagnosis may use CIDTree approach to trace the cause for reoccurring problems to the parameters involved in the processes followed. A machine learning approach like this is specifically applicable when the domain experts have difficulty following any theoretical methods of problem solving. In such domains their own practices are experience based and not theory based. The CIDTree methodology for identifying causes for defects was tested in a domain, manufacturing refractory blocks, to analyze the cause for a particular type of defect occurring in the blocks. The production processes have many parameters and take many days to be completed. The features considered here are the design specifications, manufacturing parameters and the defect descriptions of each refractory block manufactured in the industry. [Khemani, 2002]. The common technical defects which lead to rejection of the block are cracks, spalls and porosity. It is not always easy to pinpoint the cause when a manufactured part is defective. Accumulated experience however enables some experts to decide upon the process modification needed [Michael, 2002]. Thus a method to trace the relevance of the production parameters to the different types of defects like spalls, cracks, shell pockets, which often occur in the blocks manufactured would be desirable.

## 5 Building the Discrimination Tree

Defect clusters are formed manually by a query from the problem analyst describing the characteristics of the behavior or defect to be analyzed. The Induction algorithm C4.5 [Quinlan, 1993] has been used to build the induction tree for different process views. The training set of data for the C4.5 is obtained from the database, with the class labels augmented depending on their membership in the defect clusters. The method used here functions by partitioning the “fractions” of a sample having missing value into more than one branch at a test node. Thus the information provided by the leaf nodes, regarding the frequency of the total instances classified

in the node and the number of erroneous classifications are fractional in nature, e.g (3.0/1.1) as in node 15 in figure 4. Since the error rate is very low, the further calculations use rounded values of these measures. In node 15 classified as a non-defective (N) branch in figure 4, N= 3.0 and error = 1.1 so actual N=1.9 and D=1.1

## 6 Cause Induction Procedure

There are two clusters from the defect view, one for the defective cases (D) and the other for the non-defective cases (N). An induction tree has been built for the same two clusters but for different set of attributes, from the furnace view under consideration (figure 4). The induction tree has clustered the data based on their values in the corresponding view, the nodes in the induction tree contain only relevant attributes in the order of their relevance. The following factors need to be considered to evaluate the importance of a node, with respect to the defective and non-defective cases. Figure 5 explains the application of the formula.

- Percentage of defect cases over total defect frequency, classified under the node.
- Percentage of defect cases in the node.
- Percentage of good cases over the total good cases classified into sibling node.
- Percentage of good cases in the sibling node.

Percentage Measure

$C(D) = D_1/D$ = Percentage of defects over total defect frequency, classified under the node.

$S(D) = D_1/(N_1+D_1)$ = Percentage of defect cases in the node.

$C(N) = N_1/P(N)$ = Percentage of good cases over total good cases, classified into the node.

$S(N) = N_1/(N_1+D_1)$ = Percentage of good cases classified into the node.

Importance Measure

$A_1 = C(S) * S(D)$ = importance of the node with respect to defects.

$B_1 = C(N) * S(N)$ = importance of the node with respect to good cases.

Relative Importance Measure

$A_1/B_1$ = Relative importance of the node with respect to defects.

$B_1/A_1$ = Relative importance with respect to good cases.

Relative Importance Measure for the sibling node

Let $A_2$ = importance of the sibling with respect to defects.

Let $B_2$ = importance of the sibling with respect to good cases.

$A_2/B_2$ = relative importance of the sibling node with respect to defects.

$B_2/A_2$ = Relative importance of the sibling node with respect to good cases.

Score Calculation for set of sibling nodes

$Score = (A_1*B_2)/(B_1*A_2)$ = Relevance of the node in causing the defect.

After simplifying the above:

$Score = ( D_2^2*N_1^2 ) / (N_2^2*D_1^2)$

1/Score = Relevance of the sibling node in causing the defect.

The aim of the analysis is to select a pair of nodes under the same parent, such that each has high importance with respect to the alternative classes.

## 7 Utility

The refractory manufacturing domain has three different shops, the foundry, the furnace and the finishing, each providing the basis for a view. Defect clusters have been created for two different kinds of defects in the blocks, namely the edge cracks and spalls. Then the induction trees for the views foundry and furnace were built for the defects edge cracks and spalls. The Induction tree in a furnace view (figure 4) and the corresponding calculations (Table 1) have been provided for analyzing the causes for spalls. The PourToPourTime and the PowerInput parameters in the furnace process have been considered as the cause for spalls and the following suggesions have been provided for avoiding spalls.

Suggesion 1. If Pour To Pour Time <= 10 then make sure Pour To Pour Time > 2
Suggesion 2. If Pour To Pour Time > 10 then power Input should be less than 2118 Watts.

## 8 Related work

Machine learning techniques, such as a decision-tree method like ID3 and C4.5 have been used to classify objects and find discriminating behaviors. Dimension Based Induction method [Han, 1998] summarizes the major differences of a target class from a contrasting class. But the purpose of the CIDTree is to find the one most discriminating factor in the process that makes the major difference in the behavior of the target class from the rest. Also the target class, which is the defect class in this context, is very small compared to the contrasting class, here the non-defective class. And finally it acts over an induction tree and not a database as the Dimension Based Induction method does.

View: Furnace D – Frequency of defective blocks Total Test Cases: 206
Defect: Spall N – Frequency of non-defective blocks Total defective cases: 38
D=41 N=165

```
1  Pour to Pour Time <= 10 :                          D=19  N=143
3  |  Pour to Pour Time > 2 : None (146.0/6.2)        D=6  N=140
4  |  Pour to Pour Time <= 2 :                        D=13  N=3
5  |  |  Flasking Thickness L4 > 11 : None (2.0/1.0)  D=1   N=1
6  |  |  Flasking Thickness L4 <= 11 :                D=12  N=2
7  |  |  |  Flasking Thickness L2 <= 8 : None (3.0/2.1)  D=2   N=1
8  |  |  |  Flasking Thickness L2 > 8 : defect (11.0/1.3)  D=10  N=1
2  Pour to Pour Time > 10 :                           D=22  N=22
9  |  Power Input > 2118 : defect (20.0/2.5)          D=18  N=2
10 |  Power Input <= 2118 :                           D=4   N=20
11 |  |  FlaskingMedia = Sand: None (0.0)             D=0   N=0
12 |  |  FlaskingMedia = Molochite: None (19.0/2.5)   D=2   N=17
13 |  |  FlaskingMedia = SandMolochite:               D=2   N=3
14 |  |  |  Power Input <= 1820 : defect (2.0/1.0)    D=1   N=1
15 |  |  |  Power Input > 1820 : None (3.0/1.1)       D=1.1 N=3.0 – 1.1 (Round )
```

Figure 4. Induction Tree Built by C4.5 for Furnace View for the Defect Spall in Refractory Block Manufacturing Domain

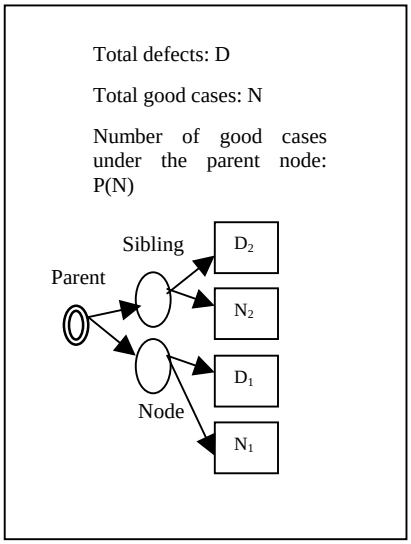


Figure 5. Distribution of Cases at the Nodes and Leaves

## 9 Conclusions and Future Work

The CIDTree suggests solutions for problems based on previous experience and assists in improving the quality of the behaviors in the domain. The basic idea is to identify a cluster of defective cases, and then try to structure them according to the parameters defining a specific view. The structure we use is the discrimination tree. If in this discrimination tree the cluster identifies with some internal node, then its parent is the discriminating feature. Furthermore, the other children of the parent are also suggestions for the remedial or correct values for the parameters. This is based on the observation that the non-defective cases are located below the other children. Our future work is towards the development of a tool to aid in the process of selecting clusters and querying specific memory structures for possible causal correlation. Also on the agenda is more extensive testing for the CIDTree procedure.


## Acknowledgements

This research is funded by the Carborundum Universal Ltd. This paper was reviewed by Michael S.M.from Carborundum Univ., Ltd. We acknowledge the contributions of Kiran P. and Ananda Rabi Dhar of A.I. & D.B. Lab.


## References


[Durfee, 1991] Edmund H. Durfee. The Distributed Artificial Intelligence Melting Pot. *IEEE Transactions on*


| Node No | Nodes | D Defect | N None | Score = $(D2^2*N1^2)/(N2^2*D1^2)$ | 1/Score | |
|---|---|---|---|---|---|---|
| 1 | PtPt <= 10 | 19 | 143 | | | |
| 2 | PtPt > 10 | 22 | 22 | 56.64 | | |
| 3 | PtPt > 2 | 6 | 140 | | | Suggestion |
| 4 | PtPt <= 2 | 13 | 3 | 10223 | | Cause |
| 5 | FTL4 > 11 | 1 | 1 | | | |
| 6 | FTL4 <= 11 | 12 | 2 | 36 | | |
| 7 | FTL2 <=8 | 2 | 1 | | | |
| 8 | FTL2 > 8 | 10 | 1 | 25 | | |
| 9 | PowInp > 2118 | 18 | 2 | | 2025 | Cause |
| 10 | PowInp <=2118 | 4 | 20 | | | Suggestion |
| 11 | FM=Sand | 0 | 0 | | | |
| 12 | FM=Molochite | 2 | 17 | | | |
| 13 | FM=SandMolo | 2 | 3 | 32.11 | | |
| 14 | PowInp <=1820 | 1 | 1 | | 4 | |
| 15 | PowInp > 1820 | 1 | 2 | | | |

Table 1. Investigating the Nodes of the Induction Tree and Suggesting Possible Causes

*Systems, Man and Cybernetics (special section on DAI)* 21(6):1301--1306, 1991.

[Han, 1998] J. Han, S.Nishio, H. Kawano and W. Wang. Generalization-Based Data Mining in Object-Oriented Databases Using an Object Cube Model. *Data and Knowledge Engineering*, 25:55 – 97, 1998.

[Han, 2001] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[Khemani, 2002] Deepak Khemani, Radhika Selvamani, Ananda Rabi Dhar, Michael S.M. InfoFrax: CBR in Fused Cast Refractory Manufacture. *In proceedings on the European Conference on Case Based reasoning*, Lecture Notes in Computer Science 2416 Springer, 560 – 574, 2002.

[Kolodner, 1993] Janet Kolodner. *Case-Based Reasoning*, Morgan Kaufmann Publishers, 1993.

[Michael, 2002] Michael, S. M., and Khemani D. Knowledge Management in Manufacturing Technology. *In Proceedings of the International Conference on Enterprise Information System*, Ciudad Real, Spain, Vol. I, 506-512, 2002.

[Schank, 1999] Roger. C. Schank. *Dynamic Memory Revisited*, Cambridge University Press, 1999.

[Quinlan, 1993] J.R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[Watson, 1997] Ian Watson. *Applying Case Based Reasoning: Techniques for Enterprise*

*Systems*, Morgan Kaufmann Publishers, 1997.