

Case-based reasoning in agriculture : analysis of the aphid case decision-support system
by David Clinton Belote

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University
© Copyright by David Clinton Belote (1993)

Abstract:
The Russian Wheat Aphid (RWA) is a very costly pest of wheat and barley in the western United States. The Aphid Case decision-support system has been developed to help disseminate expert knowledge about RWA control to decision makers. The knowledge engineering paradigm used, case-based reasoning (CBR), has several features that make it appear desirable for such a system. The advantages and disadvantages of the CBR paradigm are discussed. Experiments are performed on the Aphid Case database to determine the accuracy of the CBR technique in the domain, and these results are compared to experiments using neural network and decision tree algorithms. The results are promising, though future experiments on larger and more varied data will be. required to determine exactly how accurate and useful the Aphid Case CBR paradigm can be.

CASE-BASED REASONING IN AGRICULTURE: ANALYSIS OF THE APHID CASE DECISION-SUPPORT SYSTEM

by

David Clinton Belote

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 1993

N378
B418

APPROVAL

of a thesis submitted by

David Clinton Belote

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

December 2, 1993
Date

John T. Paxton
Chairperson, Graduate Committee

Approved for the Major Department

December 1st 1993
Date

J. Dunbigh Stanley
Head, Major Department

Approved for the College of Graduate Studies

12/13/93
Date

R L Bean
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature David C Blet

Date December 2, 1993

This thesis is dedicated to the Lord, the true author of all things.

ACKNOWLEDGMENTS

I would like to give special thanks to Dr. John Paxton and Dr. Jim Berry for helping to make this thesis possible. Dr. Paxton has helped to guide me through the intricacies of complex algorithms as well as giving me simple, good advice. This thesis has benefited immeasurably due to his able help. Thank you, John.

Dr. Berry has proven to be an able project leader and innovator for the Aphid Case project. Many of the insights and new ideas that are incorporated into the program were inspired by him. The project described in this thesis would not have existed were it not due to a great deal of effort and ability on his part. Thank you, Jim.

I would also like to acknowledge the work done by Will Lanier, who gathered the data that was used for the Aphid Case database, as well as helping with the design of the Aphid Case interface.

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF FIGURES x

ABSTRACT xi

1. INTRODUCTION TO CASE-BASED REASONING 1

    Origin 1
    Basic Paradigm of CBR 1
    Advantages of CBR 2
    Disadvantages of CBR 2
    Major Areas for Research 3

2. THE APHID CASE PROJECT 4

    Background of the Project 4
    Goals of the Aphid Case Project 5
        Major Emphasis 5
        Major Sub-goals 5
        Nature of the Domain 5
    CBR is the Best Choice for the Project 6
        Choices Available 6
        CBR Fulfills Major Sub-Goals 6
        CBR Works Well Given the Nature of the Domain 9

3. THE APHID CASE PROGRAM 10

    Hardware 10
    Software 10
    Basic Data Structure 11
        Case Name 11
        Category 11

TABLE OF CONTENTS - Continued

Features 11
Tactical Advice 12
General Advice 12
General Information 13
The CBR Algorithm 14
Weighting of Features 14
Symbolic Matching 14
Database Filter 15
User Advice 16
Database Controls 17
Display 17
Case Modification 17
Change Specific Feature Values 17
Change Data Format 17

4. RESULTS 19

Goal of the Experiments 19
Database Used 19
Accuracy of CBR 20
Experiment 20
Analysis 21
Accuracy of CBR as a Function of Database Size 22
Experiment 22
Analysis 22
Comparison with Neural Networks 23
Experiment 23
Results 25
Comparison With Decision Trees 26
Experiment 26
Results 26
Experimental Conclusions 27
General Conclusions 28

TABLE OF CONTENTS - Continued

5. AREAS FOR FURTHER EXPLORATION.....31

What is Aphid Case's highest potential accuracy rate?.....31
How to merge Aphid Case databases?.....32
Should we merge CBR with other machine learning techniques? 32
Neural Networks.....33
Decision Trees 33

REFERENCES CITED.....35

LIST OF TABLES

Table Page

1. General categories and tactical advice in Aphid Case 13
2. Classifications of cases in experimental database 20
3. Tactical advice and category accuracy of CBR algorithm 21
4. Settings used for neural network experiments 25

LIST OF FIGURES

Figure Page

1. Accuracy of CBR algorithm as a function of training set size 23
2. Accuracy of neural net algorithm as a function of database size.....25
3. Accuracy of DT1 algorithm as a function of database size.....27
4. Accuracy of DT2 algorithm as a function of database size.....27
5. Comparison of DT1 and DT2 algorithms 28
6. Comparison of neural net, CBR, DT1 and DT2 algorithms.....30
7. Linear regression of neural net, CBR, DT1 and DT2 algorithms 30

ABSTRACT

The Russian Wheat Aphid (RWA) is a very costly pest of wheat and barley in the western United States. The Aphid Case decision-support system has been developed to help disseminate expert knowledge about RWA control to decision makers. The knowledge engineering paradigm used, case-based reasoning (CBR), has several features that make it appear desirable for such a system. The advantages and disadvantages of the CBR paradigm are discussed. Experiments are performed on the Aphid Case database to determine the accuracy of the CBR technique in the domain, and these results are compared to experiments using neural network and decision tree algorithms. The results are promising, though future experiments on larger and more varied data will be required to determine exactly how accurate and useful the Aphid Case CBR paradigm can be.

CHAPTER 1

INTRODUCTION TO CASE-BASED REASONING

Origin

Case-based reasoning (CBR) is a technology developed in the field of Artificial Intelligence that uses past experience to solve new problems. CBR attempts to use memory, in the form of stored knowledge, to guide problem-solving. This approach to computer reasoning was brought to public attention by Roger Schank in his book Dynamic Memory [1], but really began to gain attention in the field of AI in the late 1980's.

Basic Paradigm of CBR

In CBR, domain knowledge is stored in the form of cases. Each case is made up of some set number of features. Different cases will have different values for these features. Besides values for each feature, each case in the database will also contain an answer or a solution that corresponds to that specific case. For example, consider a simple CBR system that identifies fruits. There could be three features: 1) color, 2) shape, 3) volume; and one solution: 4) name of fruit. One case could have the following values: 1) yellow, 2) elongated, 3) $40cm^3$, 4) banana. Another case may have these values: 1) orange, 2) round, 3) $30cm^3$, 4) orange. Now assume a new case is entered to be classified with these values: 1) yellow, 2) elongated, 3) $32cm^3$. The cases in the database are compared with this new case. When the most similar case is found, the solution

to that case (probably banana in this example) is applied as the solution to the new case, and the fruit is thereby identified.

Advantages of CBR

CBR has gained popularity largely due to many problems that have been encountered with rule-based systems. For example, because knowledge in a CBR system is simply stored as a set of cases, it is relatively straightforward to gather knowledge. And updating knowledge can be as easy as adding a new case to the database. Secondly, because a solution is based upon a past experience, the reason for that solution can be explained to the user by simply displaying that previous case. Thirdly, because there is no need to understand and generalize domain theory to fit them into rule form, CBR can perform well in domains in which there is little understanding of underlying causal relationships [2]. Finally, CBR is also able to tolerate incomplete data better than many other paradigms.

Disadvantages of CBR

There are several weaknesses of the CBR approach. First, it is a form of supervised learning: it requires a teacher to enter in the correct solutions to the cases in the database. Secondly, it does not find the reasons behind the relationships it reveals, it simply deals on a case-by-case level. And thirdly, inconsistencies in the database can easily go unnoticed. Finally, CBR introduces two difficult research issues that are the topics of the next section.

Major Areas for Research

There are two difficult issues concerning the CBR paradigm which must be solved to have a useful CBR system. First we must find a way to efficiently search the database. Most complex domains will have a database consisting of a great number of cases. To compare a new case with every case in the database could easily become too time consuming; some efficient form of indexing or database filtering must be applied. The second issue deals with how we should determine which case in the database is the most similar to the new case, when no case matches exactly. A number of different solutions have been suggested for these two issues [3]. We will later explore some possible solutions.

While CBR is still a relatively new paradigm, there have been several successful implementations of the technology. For example, a CBR system called Prism was created to classify telex messages received by a large bank. Originally the bank used a rule-based system that had an accuracy level of 72% and took two to seven seconds per classification. The Prism system achieved an accuracy of 77% and processed each telex in 0.19 - 0.25 seconds, prompting the bank to switch to the CBR paradigm [4]. Other domains in which CBR has been at least partly successful include medical diagnosis, recipe creation, and legal reasoning.

CHAPTER 2

THE APHID CASE PROJECT

Background of the Project

The impetus for the creation of Aphid Case, the project described in this thesis, is the very costly Russian Wheat Aphid (RWA). The RWA is an especially damaging pest that can destroy fields of wheat and barley. It was not found in the United States until March, 1986 [5]. However by 1991 it had already cost American producers over $300 million in damages and control costs [6], and was found in seventeen western states [7]. Unfortunately, because the RWA is such a new pest in the United States, very little is known about it. In addition, there are only a few experts, making it difficult for the majority of producers to have adequate access to good information about the RWA. In answer to these problems and the high toll that the RWA was taking on American production of wheat and barley, the United States Department of Agriculture (U.S.D.A.) sponsored a major research proposal. This proposal had three main goals [8]:

1. Study RWA effects upon wheat growth
2. Study how differing planting dates affect RWA damage
3. Develop decision support software to help disseminate useful knowledge about the RWA

Aphid Case was one of the tools to come out of the third goal.

Goals of the Aphid Case Project

Major Emphasis

As stated above, the major emphasis of Aphid Case is to help disseminate useful advice in the domain of managing or avoiding RWA infestations of wheat and barley crops. Aphid Case is not designed to create new domain knowledge, but rather to share the expertise held by a few individuals with the great many people to whom it would be extremely useful.

Major Sub-goals

To best achieve this main goal, the team designing Aphid Case chose four major sub-goals:

1. Use a data representation and user interface that makes initial data gathering as easy as possible, because this is a very complex domain and access to domain experts' time is limited.

2. Allow all types of new knowledge to be easily entered by users in the future, because more information is constantly being learned about the RWA.

3. Make sure that Aphid Case is simple and pleasant to use because most users will not be overly familiar with computers.

4. Make sure the reasoning technique used to make recommendations is understandable by the average user because Aphid Case is a decision support system rather than a system claiming always to give optimal advice.

Nature of the Domain

The final factor we considered when designing Aphid Case was the nature of the RWA management domain. Little is understood about cause and effect

relationships in this domain [9]. So we knew we had to use a knowledge engineering technique that could cope with a weak domain theory.

CBR is the Best Choice for the Project

Choices Available

We considered various possible alternatives for building Aphid Case. Perhaps the most common technique used for decision support systems is the rule-based paradigm. Other common techniques include decision trees and neural networks. Yet we chose CBR because it seemed to have several nice features to help us fulfill our objectives.

CBR Fulfills Major Sub-Goals

1. Our first sub-goal was that the system allow data to be gathered very easily and efficiently. Rule-based systems often require the domain expert to express his knowledge in the form of rules. Quite often the expert will find this very difficult or impossible to do. It is a very time-consuming and intensive process, leading to the common problem of the knowledge-acquisition bottleneck. In this situation, acquiring the necessary domain knowledge for the computer system becomes so difficult and slow that the rest of the development process is delayed.

However ease of knowledge acquisition is a principle strength of CBR. Knowledge is simply represented by a set of cases, each of which contains some number of features describing a certain situation. Knowledge Acquisition just means entering cases into the database. This can be done very quickly and efficiently. In fact, during an interview, the knowledge engineer can enter the

data as the domain expert is speaking. This saves an enormous amount of time for both parties.

2. The second major sub-goal was that Aphid Case should have a knowledge base that could be easily updated in the future. It can be very difficult to add new information to a rule-based system, because it is hard to know exactly what will happen to its overall functioning when a new rule is introduced. Often there will be unexpected and unwanted effects upon the system's output. And neural networks must be retrained in order to take advantage of new information. This retraining can be a very time-consuming process.

Once again, however, because of CBR's very simple knowledge structure, it is very easy to add new information to its database. One simply adds a new case to the database. Now, it should be noted that this does not guarantee that new information will always increase the accuracy of the system, for as in the rule-based system, unexpected and unwanted effects can result.

3. Our third sub-goal was to make sure that Aphid Case would be perceived by the users as simple, straightforward, and intuitive. In achieving this goal, it is possible that the backward-chaining rule-based system would have been slightly preferable to a CBR or neural network approach. Most CBR implementations ask for the values of all features. But a backward chaining system is able to ask for values of only the features needed to arrive at a solution. This may be beneficial to the user. However, we have implemented a filtering technique (discussed in chapter 3) that helps make Aphid Case easier to use and lessens this handicap.

4. Our fourth sub-goal was that the reasoning process of Aphid Case be easy to understand by the average user. Aphid Case is designed to give useful

advice and suggestions, but not completely replace the need for human decision-making and guidance. Therefore it is important that the user understand how Aphid Case determines its results. Rule-based systems usually explain their reasoning by listing long chains of rules. These can be hard to follow. And it is extremely difficult, often impossible, to understand exactly how a neural network creates a specific output.

But CBR has a great advantage in the area of explaining its results. Since these results are based on one or more cases in the database that are similar to the user's newly entered case, the system can simply display these similar cases to the user as the reason for classifying their input in the way it was classified. The user can look for himself to see if he agrees that the previous examples are similar enough to the new situation to warrant taking the advice of the system. In fairness, it must be admitted that how easily the user understands the reasoning process behind the given classification depends upon how simple and straightforward the similarity evaluation function is in the system. CBR has the advantage, that if this is a key priority, this similarity function can be made very simple and straightforward (with some possible sacrifice in accuracy), and this is what we have chosen. There is no complex chain of rules, as in rule-based systems; nor a potentially complex and confusing tree to follow, as in the decision tree paradigm; nor a mysterious black-box that simply receives input and delivers output, as in the case of neural networks.

CBR Works Well Given the Nature of the Domain

As was remarked earlier, the domain of RWA management is poorly understood. Since "the **strength** of a domain theory depends on the certainty associated with the relationships between terms," [10] it is clear that this is a weak-theory domain. Generalization-based methods such as rule-based systems use a highly compiled form of knowledge, as compared to the more raw form found in CBR. However, the more highly compiled knowledge is, the longer and less reliable the chains of inference become. In strong-theory domains this is acceptable; but in weak-theory domains it is best to keep the chains of inference as short as possible. The simple matching nature of a basic CBR system helps to do this [11]. Therefore we felt that CBR would be the best technique for the RWA management domain.

CHAPTER 3

THE APHID CASE PROGRAM

Hardware

The first consideration involved in the actual creation of the Aphid Case program was what kind of hardware would be required to run it. The target market for Aphid Case is threefold: 1) RWA experts, 2) agricultural extension agents that give advice to producers, and 3) producers themselves. To allow these various users to be able to run Aphid Case, we decided to aim for the standard IBM PC compatible market. Because we wished to take advantage of the ease of use and public familiarity of a windows-style graphical user interface we decided to produce a program that ran under Microsoft Windows. Thus we decided to require a machine that could adequately run Microsoft Windows. This suggests any computer with at least 4 MB of RAM and an Intel 386-compatible or greater CPU.

Software

Next we needed to determine what programming language to develop our windows program in. We chose KnowledgePro for Windows 2.0, which is a high-level language that allowed us to focus our attention on algorithmic details rather than being too concerned with various low-level windows details. KnowledgePro also included nice features such as object-oriented programming,

extensive list handling, and options for interpreted or compiled code, making it an attractive development environment.

Basic Data Structure

The basic data structure used in Aphid Case is the case. A case consists of the following six items (described in detail below):

- Case Name
- Category
- Features
- Tactical Advice
- General Advice
- General Information

Case Name

Each case in Aphid Case has a simple case name. The case name consists of the general category that the case falls under as well as a numeric identifier to make the name unique.

Category

Each case belongs to one of five general categories that reflect the type of management that is suggested for that situation. The five general categories are listed in Table 1.

Features

A case consists of values for forty-seven different features describing potentially relevant current conditions, such as "Crop," and historical conditions,

such as "History of RWA Infestation." For instance, the feature "Crop" may have the value "Winter Wheat," "Spring Wheat," "Barley" or "unknown" (which appears as a blank space in Aphid Case). It is these values that Aphid Case uses to determine how similar different cases are. These features were acquired by interviews with experts in the RWA domain. They were asked to list what features are most important to know values for in order to be able to suggest good pest management advice. Then each expert was asked to rank the importance of these features. Finally, all of their various opinions were merged and a ranked list of forty-seven features was the result. The features in Aphid Case are ordered according to the expert rankings. The values that each feature can have were also obtained from the experts. Some features may have as few as two possible legal values to be chosen from, while others may have up to twenty.

Tactical Advice

Each case is assigned an expert-supplied tactical advice. This is a fairly specific piece of advice, telling the producer what should be done about the RWA infestation in his or her field. Each type of tactical advice falls under one of the five general categories as shown in Table 1.

General Advice

General Advice is advice that goes beyond the simple and straightforward format of the tactical advice. For instance, suggestions about what should be done in a year, or what could have been done better several months previously, or additional suggestions not included in the tactical advice could all be included under this category.

General Category	Tactical Advice
Chemical	apply granular insecticide apply insecticide spot spray insecticide apply foliar spray apply tank mix apply seed treatment
Cultural	delay seeding control green bridge seed at optimum date for agronomics livestock grazing use as set aside acres
No Risk	no risk of RWA damage
Too Late	no treatment: too late
Wait or Scout	wait and scout wait, irrigate, and scout scout until threshold reached wait until weather warms scout volunteer wheat and crop irrigate and scout scout until weather moderates

Table 1: General categories and tactical advice in Aphid Case

General Information

General Information includes any information that is useful to keep track of but is neither important for the classification of the case, nor contains any advice for the producer. Information such as the name of the person who entered the case into the database, the address of the producer whose field was described in the case, and similar kinds of information can be placed under this category.

The CBR Algorithm

The heart of the Aphid Case program is the CBR algorithm that takes a newly entered case and searches the database in an attempt to find the most similar case. There are many different ways this can be done within the CBR paradigm.

Weighting of Features

For instance, the simplest method would be to simply count how many matching values the new case had with each old case, and choose as most similar that old case with the most matches. This was done in the early CBR system Proximity [12]. However, it was quite clear from our discussions with RWA experts that some features were far more important than other features in classifying a case, so we decided to assign numeric values to each of the features. The higher a feature was ranked, the greater the numeric value assigned to it. Our final rankings varied from 60 for the most important feature to 1 for the least important. Instead of simply counting features that match, the system keeps track of total match strength between two cases by adding the numeric values of those features whose values match.

Symbolic Matching

Another important issue to be addressed was what to do if the new case and the database case values for a certain feature did not exactly match, but were similar. For instance, perhaps the new case had a value of "10-15%" for the feature "% Infested Tillers" while a database case had a value of "15-20%." Should the total match strength between the two cases increase somewhat because of the closeness of the values? This could help the accuracy in some cases, and could reduce the danger of picking a poor choice as the best

matching case. However, we chose to only increase the total match value when the match was exact, for the following reasons: 1) The system is much easier to understand for the average user if only exact matches contribute to the final match strength score. 2) In a domain as poorly understood as RWA control, placing any kind of partial value depending upon "closeness" would largely be pure guesswork. 3) Finally, the system is designed to be easily modifiable in the field. If, each time a new legal value for a feature was added, the user had to relate its similarity values to other existing legal values for that feature, much more effort and knowledge would be required. For these reasons we chose to treat all feature values, including those that seemed to have an intrinsic order, as symbolic.

Database Filter

There was a final problem to be solved regarding the CBR algorithm. Because each case consisted of forty-seven features, Aphid Case required the user to enter in values for as many of those features as possible. Thus the user had to face a screen of forty-seven blanks to be filled in every time he or she wanted to enter in a new case. After displaying our system to domain experts, we came to realize that this was an intolerably large number of fields for a user to be faced with. We solved this problem using a technique we call a "database filter." Instead of initially presenting the user with a screen of forty-seven features for which values were desired, Aphid Case begins by requesting the values of just a few (generally 2 to 4) very important features. These are features that the experts feel must have the same values for any two cases to be considered similar. Then the database is searched, and only those cases whose features match all of these filtering values are kept as possible matches to the

new case. These remaining cases are then compared one to another to determine which of their features make them each different from one another. Generally they will have differing values for between ten and thirty features. Next the user is presented with these remaining features to be entered. Once this is complete, Aphid Case searches the remaining cases, finding each case's total match score with the new case. The highest scoring case is then declared the winner and the user is advised to note the category it belongs to, as well as the tactical and general advice pertaining to that case. Thus, the user was probably spared having to enter between ten and thirty unnecessary values by the use of the database filter. And there is no danger of excluding an important case by the use of the filter, because the features that are used for the database filter are only those that are determined to be necessary matches for two cases to be considered similar.

User Advice

As stated above, after the necessary feature values have been entered by the user, the program finds the case in the database that best matches the newly entered case. The general category and tactical advice pertaining to this case is displayed, and the user is given information such as the strength of the match, differences between this case and the new case, other high-scoring cases, and how to best interpret these results. The user may also save the new case in the database, with either the program's recommended advice, or with the user's recommendation should the user believe the advice was incorrect. At this point, it is hoped that the user will either follow the advice generated by Aphid Case, or has at least been offered some factors to consider in making his or her decision.

Database Controls

To achieve our goals of ease of use and field maintainability, we incorporated a number of database and knowledge representation controls.

Display

Aphid Case allows the user to select several options for display:

- Display all the names of the cases in the database
- Display specific cases
- Display cases with one or more specific feature values

Case Modification

The user can also choose to add a case to the database, delete a case from the database, or modify a case that is already in the database.

Change Specific Feature Values

The user may choose to change a specific feature value in any of the cases in which it appears. For instance, the user may wish to change every instance of "Barley" which appears under the feature "Crop" to "Irrigated Barley". Aphid Case makes this kind of search and replace procedure simple and straightforward.

Change Data Format

Aphid Case also incorporates an innovative approach to remaining very flexible to new types of information as more knowledge is gathered on the RWA. Not only can new cases be easily added to the database, but the actual format of cases can be changed from within the program itself. While running Aphid Case the user can change the following:

- The features that are stored for each case: add, delete or modify the features that Aphid Case stores for each case.
- The legal values allowed for each feature: add, delete or modify any of these values as new cases or better knowledge becomes available.
- The numeric weights for each feature: modify any of the numeric weights used for each feature.
- The features that are used for the database filter: select a new set of database filter features at any time.

These features make Aphid Case very flexible in the face of new domain knowledge. We believe that they will allow Aphid Case to continue to improve and grow as new information about the RWA becomes available. The danger, of course, is that having such an easily modifiable system will allow improper changes to occur, resulting in a degradation of performance. We will deal with this topic in chapter 5, "Areas for Further Exploration."

CHAPTER 4

RESULTS

Goal of the Experiments

The goal of the following experiments was to determine whether or not the CBR algorithm we had developed was a suitable tool for the Aphid Case decision support system. We have seen that it fulfilled our objectives as an easy to build, easy to use, and easy to maintain system; however, most importantly, did it give good, useful information to the user? Secondly, because it is a relatively new system, its database of cases is fairly small and limited. So we would like to know how its accuracy and effectiveness would be affected by an increased database. And finally, how does the accuracy of the Aphid Case CBR algorithm compare in this RWA domain with that of a neural network approach and a decision tree approach? Is it comparable, or does it suffer for its simplicity and easy maintainability?

Database Used

The database used for all experiments is the actual Aphid Case database that was gathered through interviews with seven experts on the RWA in the western United States. This database consists of 124 different cases covering all five of the major categories as shown in Table 2.

Category Tactical Advice	Number of Cases	% of Database
Chemical	46	37
apply granular insecticide	4	3
apply insecticide	18	15
spot spray insecticide	3	2
apply foliar spray	13	11
apply tank mix	6	5
apply seed treatment	2	2
Cultural	9	7
delay seeding	3	2
livestock grazing	5	4
use as set aside acres	1	1
No Risk	3	2
no risk of RWA damage	3	2
Too Late	6	5
no treatment: too late	6	5
Wait or Scout	60	48
wait and scout	25	20
wait, irrigate, and scout	5	4
scout until threshold reached	24	19
scout volunteer and crop	2	2
scout until weather moderates	4	3

Table 2: Classifications of cases in experimental database

Accuracy of CBR

Experiment

The testing methodology used to test the overall accuracy of the CBR system was to select at random 80% of the cases of the database (99 cases) to form the training set. Then the remaining 20% (25 cases) served as the testing set. Each test case was entered into the CBR algorithm and the highest scoring case of the training set was selected as its most similar case. If the two cases had the same value for their tactical advice, then this was deemed to be the correct

tactical advice. If not, it was considered to be the incorrect tactical advice. Similarly, if the two cases had the same value for their (more general) category, then this was deemed to be the correct classification in terms of general category. And if not, then this was considered an incorrect classification. Then another random selection of 80% of the database was created and the tests were repeated. This procedure was repeated a total of five times.

Type of Accuracy	Average Result	Best Result	Worst Result
Tactical Advice	51%	68%	44%
General Category	74%	84%	68%

Table 3: Tactical advice and category accuracy of CBR algorithm

Analysis

We see that the CBR algorithm chose the correct one out of the sixteen possible values for tactical advice an average of 51% of the time. This result was slightly higher than expected given the relatively small database (there are only an average of eight cases that have each given tactical advice value). However, it is too low to be of great use, since this means that it supplies the incorrect tactical advice nearly one of every two times it is run. We conclude that while this is a promising result, given that we expect the database to grow rapidly, it is still too small to justify relying on the current version of Aphid Case too heavily for a specific tactical advice.

However, the result for the general category is more promising. In these series of tests we ranged from 68% to 84% accuracy in this broader classification, averaging 74%. Thus, as would be expected, Aphid Case does a better job at giving more general advice. Yet though it is more general, this is

still very useful advice, and could prove to greatly help the decision support process as producers attempt to control RWA infestations. There will simply need to be more human input in determining specific actions. So until the database gets large enough to allow for further testing, we conclude that the tactical advice suggested by Aphid Case *may* be useful to the user, and the category suggested is very likely to be of use.

Accuracy of CBR as a Function of Database Size

Experiment

In order to determine how the general category classification accuracy of the CBR algorithm varies according to the size of the database, we performed the same set of experiments described above on training sets from 10% to 90% of the Aphid Case database at 10% intervals. For each of these nine intervals, we randomly created five separate training sets, for a total of forty-five training sets. For each training set, we ran the remaining cases of the database through as a testing set. Thus the testing set for the 10% training set consisted of the other 90% of the database, while the testing set for the 90% training set consisted of just 10% of the database, and was thus prone to somewhat more erratic behavior. However, because we repeated each experiment a total of five times, our average results proved fairly stable and predictable.

Analysis

The results from this set of experiments show the strong correlation between the size of the training set and the accuracy of the algorithm. The average accuracy at 10% of the database is 42% and improves at every interval except one, to end at 75% at 90% of the database. The fact that the accuracy

consistently improves is very promising. While we do expect the rate of improvement to decrease as the database training set becomes larger, the average accuracy may potentially be in the 80-90% range when the size of the database doubles.

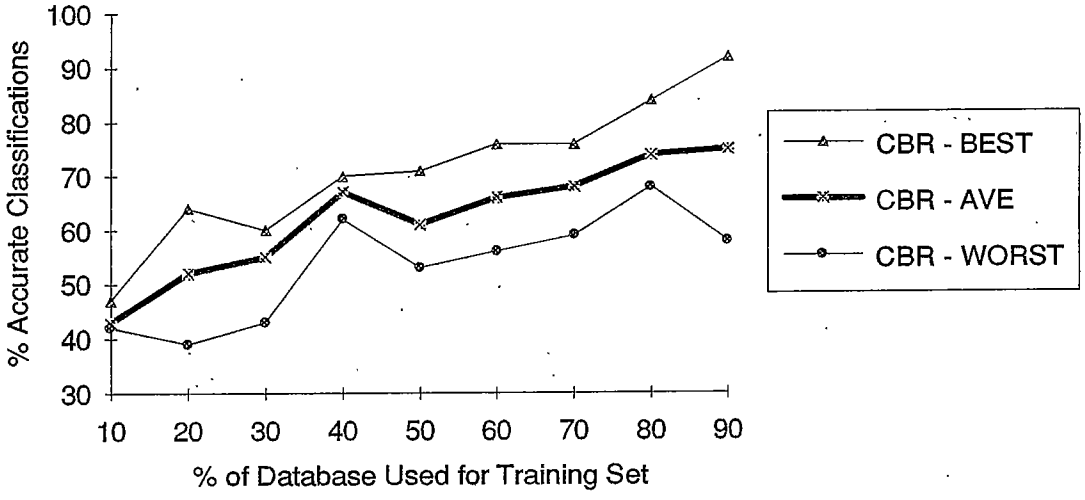


Figure 1: Accuracy of CBR algorithm as a function of training set size

Comparison with Neural Networks

Experiment

The NETS 2.01 neural network package produced by NASA was used to evaluate how the neural network approach would work (in terms of classification accuracy) on the Aphid Case database in comparison with the CBR approach. The input layer consisted of fifty-six nodes. Forty-four of the features were of such a nature that their values could be ordered, and thus needed to be assigned only one node each. For instance, for "Economic Potential" the node received a value of 0.1 if the feature value was "low", 0.5 for "medium", and 0.9 for "high". If the value was missing it was assigned the average value, in this

case 0.5. The legal values for the remaining three features could not be reasonably ordered, so 3 nodes were used for "Crop", 3 nodes for "Grazing or Harvest", and 6 nodes for "Available Insecticides". The reason for not treating all the features as purely symbolic as they are treated by the CBR algorithm, was because this would have led to a very huge network consisting of perhaps 300 input nodes, and presenting a very real challenge to train. Further, it is not advisable to have 300 input nodes when the total number of examples you have to provide the network is at most 112.

Because we were training the network to determine the correct one of five categories for the case, our output layer consisted of 5 nodes, one for each category. In order to avoid the possibility of "memorization" by the network, it is important to make the middle layer fairly small. A middle layer of 30 nodes was chosen as an initial experiment. Using the 112 input examples, it took many time-consuming attempts to successfully train the network with this topology, so it was decided to not try to train a network with fewer nodes in the middle layer. In fairness, however, it should be noted that given the relatively small number of examples available, the network still did not have few enough connections to avoid the possibility of "memorization" completely. We chose a fully connected network, and were able to train each network using the default values provided by the NETS program. Due to the long training times for the networks, only one network was trained for each of the nine different sized training sets, rather than the five separate tests used in the case of the CBR testing.

Input Layer	56 Nodes
Hidden Layer	30 Nodes
Output Layer	5 Nodes
Maximum Weight Value:	0.357
Minimum Weight Value:	0.001
Global Learning Rate:	0.268
Global Momentum:	0.900

Table 4: Settings used for neural network experiments

Results

The network performed similarly to the CBR algorithm, though more sporadically due to the fact that fewer experiments were performed to average out the output results.

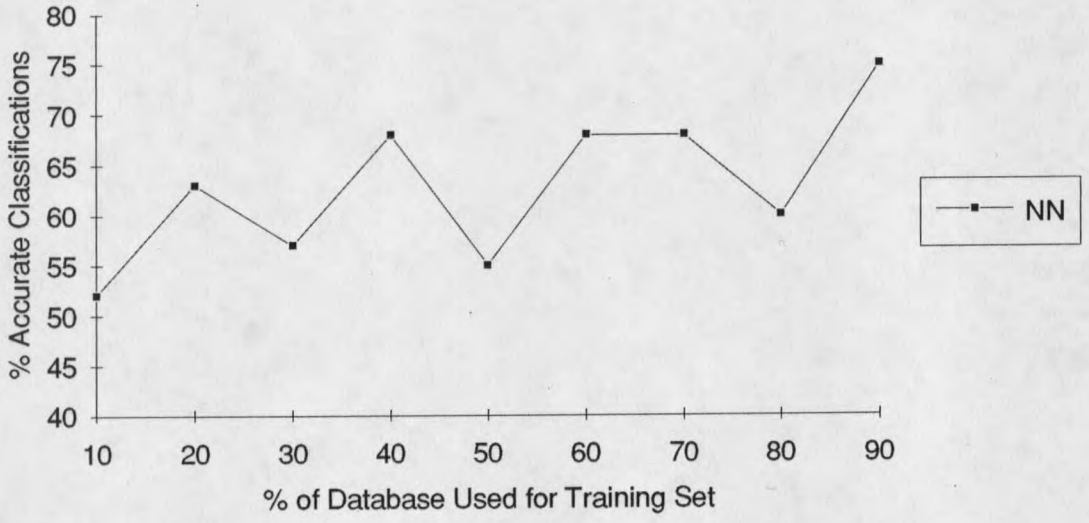


Figure 2: Accuracy of neural net algorithm as a function of database size.

Comparison With Decision Trees

Experiment

We chose to build decision trees to compare their accuracy with that of the CBR algorithm. The type of decision trees we decided to use were binary trees. This type of tree always splits into two branches at each decision point. One branch takes a certain value, and the other branch takes all other values. We chose the binary decision tree because it has proven more accurate in domains that are highly uncertain and full of missing information such as the Aphid Case database [13]. We also needed to decide how to handle missing information while building the trees. One way is to treat the missing information as simply another value for the feature. This approach is labeled DT1. A second method is to branch only on a specific actual value, and not treat missing values as true values. This approach is labeled DT2. We implemented both of these types of decision trees, using the same forty-five databases that were created to test the CBR algorithm.

Results

The decision tree in which missing values were treated as true values (DT1) tended to be the more accurate of the two methods (figures 5 and 6). However, the CBR algorithm outperformed both of the decision trees in a consistent manner for all of the training sets except the smallest.

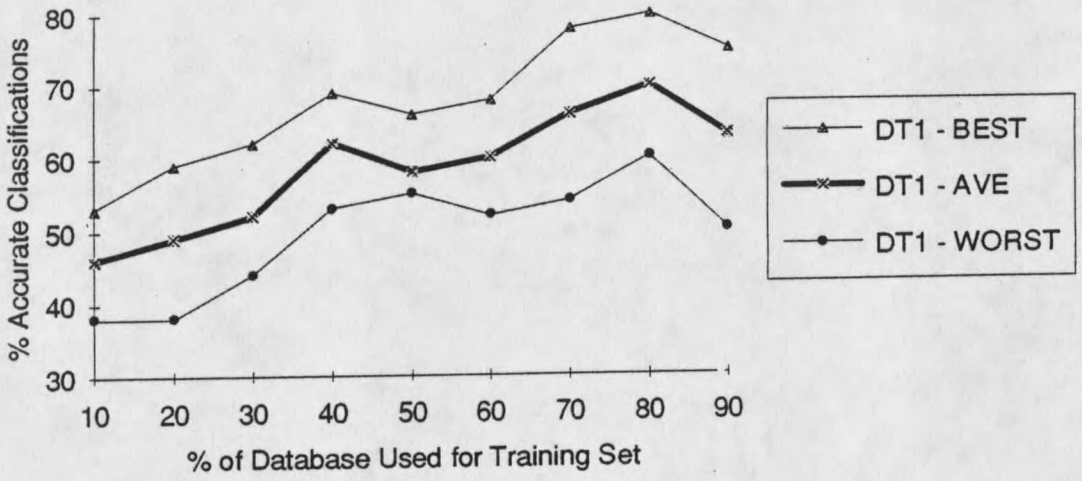


Figure 3: Accuracy of DT1 algorithm as a function of database size

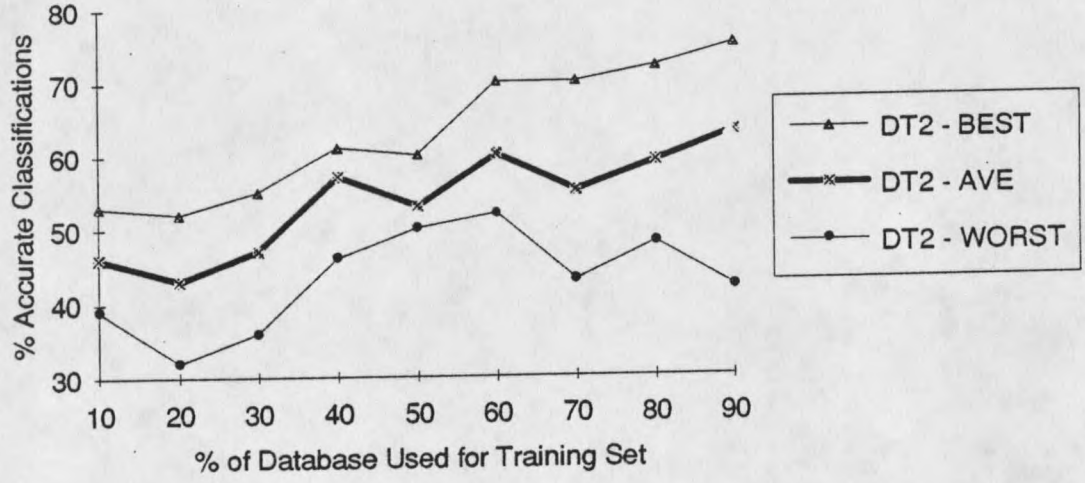


Figure 4: Accuracy of DT2 algorithm as a function of database size

Experimental Conclusions

From these tests results, we can conclude that the CBR algorithm seems to perform only average when attempting to give highly specific tactical advice based only on the current fairly small database. However, it seems able to fairly

accurately suggest general categories of action such as "Chemical" or "Wait or Scout". And it seems to improve almost linearly with the size of the training set used (though it does appear to begin leveling as it reaches the 80% accuracy area). Finally, not only is CBR much easier to update and modify, but it appears to be at least as accurate as a neural network or decision tree approach (if not more accurate). For these reasons, we believe that the CBR approach seems to be well suited to this highly complex and poorly understood domain. See figure 6 for a comparison of the results of these three different techniques, and figure 7 to see the results of applying linear regression to each of the graphs.

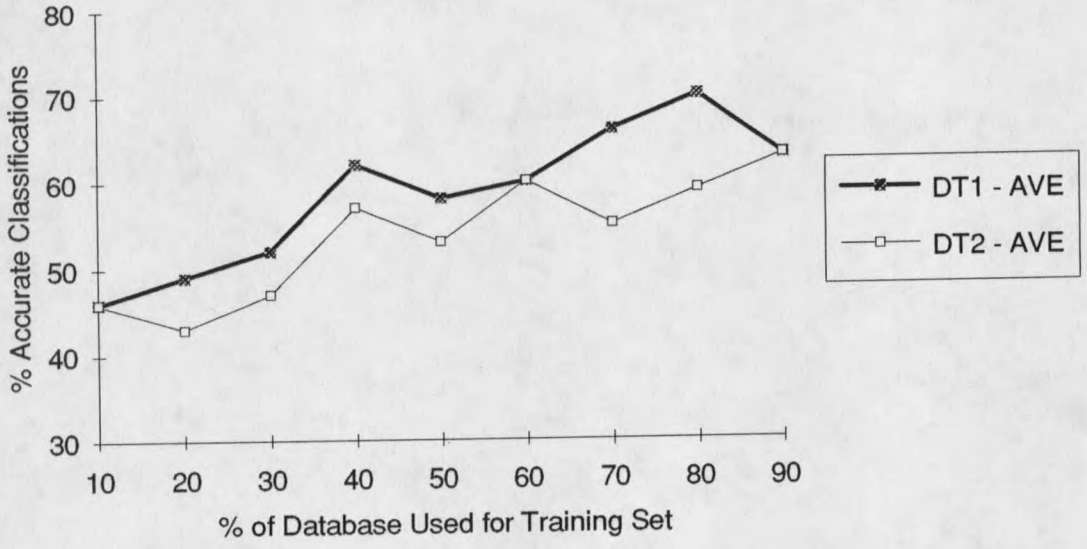


Figure 5: Comparison of DT1 and DT2 algorithms

General Conclusions

We feel that we have achieved our primary goals of Aphid Case: we have developed an easy to use, easy to maintain, and fairly accurate decision-support system for the domain of RWA management. The most innovative aspect of

Aphid Case, the extremely flexible and modifiable database format, offers both great opportunity and great danger. The opportunity is that the system can easily grow as our knowledge of the RWA increases; the great danger is that giving too much control of the knowledge and its format to many different people will make it hard to gather and assimilate the large amount of information that it will acquire throughout the western United States. Only time will tell if Aphid Case will continue to be a useful decision support system for RWA control.

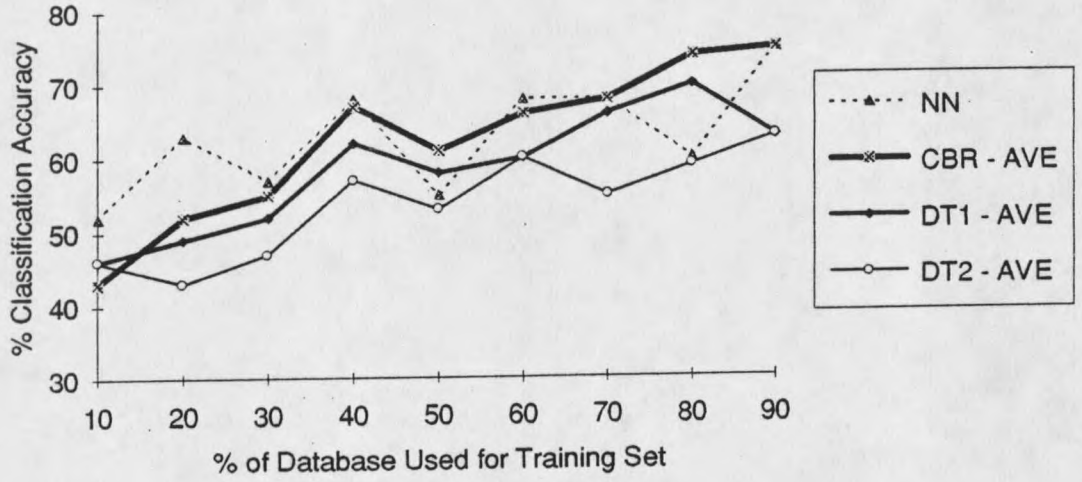


Figure 6: Comparison of neural net, CBR, DT1 and DT2 algorithms

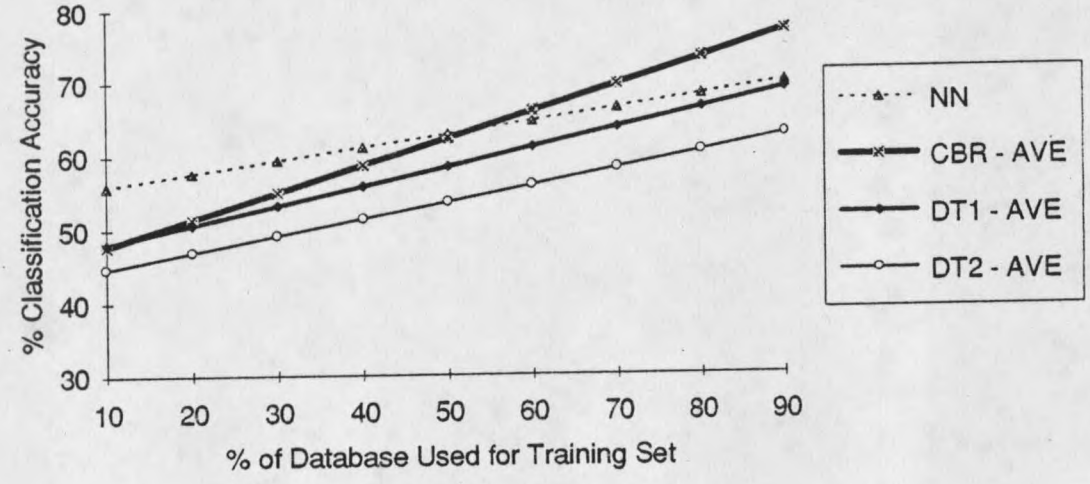


Figure 7: Linear regression of neural net, CBR, DT1 and DT2 algorithms

CHAPTER 5

AREAS FOR FURTHER EXPLORATION

There are a number of areas related to Aphid Case that would be well suited for further exploration. Among these are: 1) What will be the highest accuracy that can be hoped for when the database size becomes much larger? 2) How should various versions of the Aphid Case databases be merged as they slowly diverge from one another? 3) Is there any way to combine the CBR algorithm with either a decision tree or neural network approach to increase accuracy or improve the user interface?

What is Aphid Case's highest potential accuracy rate?

We saw that when we used 90% of our database as a training set (112 cases) our average accuracy rate reached 75% for correctly determining the category that a case would fall under. It appeared that the CBR accuracy graph (see figure 1) was continuing in an upward trend at this point. How high would it go before it began to level off, and what size of database would accomplish this value? For instance, if we doubled the training set to equal 224 cases, would our accuracy rate approach 90% or higher? Until Aphid Case has been distributed and many more cases are added to its database, this issue remains in doubt. However, we look forward to repeating this testing as soon as a larger database becomes available.

How to merge Aphid Case databases?

A primary emphasis of the Aphid Case system has been that it will be field maintainable. To achieve this, we give future users great control over the contents and formats of cases that are stored in the database, not to mention which cases they choose to store as well. We have hoped that this will allow Aphid Case to continually improve its accuracy and reliability as more scientific and case history data becomes available.

However, there is a potential problem that arises from the great flexibility that has been built into the system. It is possible that the Aphid Case database that is in use in (for instance) California will, over time, become very different than that in use in Montana. Or perhaps two California Aphid Case sites will become very different. Not only may the cases that are stored vary greatly, but even which features represent a case may come to be very different. If this occurs, what should be done? Should the databases remain forever distinct and independent, or should they somehow be merged to capture the knowledge of both sites? If the latter is the case, how do you merge databases in which there are varying features from one database to another, not to mention varying legal values and numeric weights assigned for each feature. We see this issue becoming very important for Aphid Case within one to two years, and believe that it is worthy of some study.

Should we merge CBR with other machine learning techniques?

There is a great deal of interest currently in merging various machine learning techniques such as neural networks, rule-based systems and CBR to

come up with tools to better handle complex sets of data [14]. We see several possible avenues of exploration for Aphid Case.

Neural Networks

One potential way to merge a neural network approach with our CBR algorithm would be to train a network to rate how similar two given cases were. Note that this is different than training a network to classify a case, for we would not ask the network to go that far; rather it would simply return a value for how similar any two cases were. Thus we may not have to retrain the network every time a new case is entered, as we would need to do in the previous version. We could possibly add many new cases, and simply allow our previously trained network to rank similarities. Several issues arise from this approach: (1) How do you train this kind of network? (2) Could it accurately rank similarities on cases that were not involved in its training (such as new cases)? (3) Would this type of similarity matching be more accurate than our much simpler numeric weights method?

Decision Trees

One way to merge a decision tree based system with the CBR system would be to use a decision tree to help modify the previously set numeric weights of the features. For instance, perhaps the feature "Plant Date" was given a fairly low numeric weight, but the decision tree formed from the database placed this feature at or very close to the root node of the decision tree. And perhaps "Forecast" was given a high numeric rating by the experts, but the decision tree either did not use it at any nodes, or only used it very far down in the tree. Given these circumstances, there may be an algorithm that would allow the numeric weights of "Plant Date" and "Forecast" to be increased and decreased

respectively. And perhaps this would be a more efficient way of determining optimum numeric feature weights than simply going through many tests of the CBR algorithm and varying weights (since there are forty-seven different features, each of which can have one of sixty different weights, in the current database alone there would be approximately 60^{47} possible weight settings).

Decision trees could also be useful in conjunction with the Database Filter. Instead of presenting the user with all of the conditions for which those cases that passed the filter had some differing values, a decision tree algorithm could be used to find that condition for which an answer from the user would best divide the remaining cases. This most informative attribute could be discovered by using the same information gain algorithm used by decision trees. Once the value for this condition was received, the next most useful condition could be found, and the user could be prompted for it. In this way, it may be possible to present fewer conditions to the user than is presented by the current Database Filter technique.

REFERENCES CITED

[1] Schank, R. C. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, 1982.

[2] Porter, B. W., R. Bareiss, and R. C. Holte. *Concept Learning and Heuristic Classification in Weak-Theory Domains.* Artificial Intelligence, vol. 45, 1990, pp. 229-263.

[3] Riesbeck, C. K. and R. C. Schank. *Inside Case-based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[4] Barletta, R. An Introduction to Case-Based Reasoning. *AI Expert*, October, 1991, pp. 43-49.

[5] Johnson, G. D. *The Russian Wheat Aphid: Identification, Biology and Management,* Montana State University Extension Service, EB 49, September, 1989.

[6] Linden, E. Will We Run Low On Food? *Time Magazine*, August 19,1991, p. 48.

[7] Pike, K. S. and D. Allison. *Russian Wheat Aphid: Biology, Damage, and Management.* Pacific Northwest Extension Publication, PNW371, June, 1991.

[8] Berry, J. S. *RWA 1991 Annual Progress Report.* 1991.

[9] Berry, J. S. *Decision Support Software Development and RWA/Wheat Plant Computer Simulation for Economic Threshold Development.* U.S.D.A. grant proposal, 1989.

[10] Porter, pp. 229-263.

[11] Ibid.

[12] D. Kibler and D. W. Aha. *Learning Representative Exemplars of Concepts: An Initial Case Study.* Proceedings of the Fourth International Workshop on Machine Learning, Morgan Kaufman, 1987. pp. 24-30.

[13] Cheng, Fayyad, Irani, and Qian. *Improved Decision Trees: A Generalized Version of ID3.* Proceedings of the Fifth International Conference on Machine Learning. Ann Arbor, Michigan, 1988. pp. 100-106.

[14] Hedberg, S. *New Knowledge Tools.* Byte Magazine, July, 1993, pp. 106-111.

MONTANA STATE UNIVERSITY LIBRARIES
3 1762 10208748 1