Data Mining and Hypothesis Refinement Using a Multi-Tiered Genetic Algorithm

Christopher M. Taylor and Arvin Agah

Department of Electrical Engineering and Computer Science,
The University of Kansas, Lawrence, KS 66045 USA

ABSTRACT

This paper details a novel data mining technique that combines set objects with an enhanced genetic algorithm. By performing direct manipulation of sets, the encoding process used in genetic algorithms can be eliminated. The sets are used, manipulated, mutated, and combined, until a solution is reached. The contributions of this paper are two-fold: the development of a multi-tiered genetic algorithm technique, and its ability to perform not only data mining but also hypothesis refinement. The multi-tiered genetic algorithm is not only a closer approximation to genetics in the natural world, but also a method for combining the two main approaches for genetic algorithms in data mining, namely, the Pittsburg and Michigan approaches. These approaches were combined, and implemented. The experimental results showed that the developed system can be a successful data mining tool. More important, testing the hypothesis refinement capability of this approach illustrated that it could take a data model generated by some other technique and improves upon the overall performance of the data model.

KEYWORDS: Data mining, genetic algorithms, intelligent systems, multi-tiered

1. INTRODUCTION

As more and more information is added to already immense databases, it is becoming increasingly difficult and important that the information can be analyzed. After all, what good is it to have all the information in the world but not know what

Address for correspondence: Arvin Agah, Dept of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS 66045 USA; Tel: (785) 864-8821; Fax: (785) 864- 3226; Email: agah@ku.edu

the information means or what it might indicate? Within all collections of similar data, patterns exist. These patterns can be used to discover the meaning about the data and what they represent. For example, an analysis of the medical records of a single hospital can reveal the types of illnesses common to the area the hospital serves. The analysis could reveal the types of treatment that were successful versus those that were not as effective.

This type of analysis, called *data mining*, uses computer science techniques to search for patterns within data. With the vast amounts of information that are contained in the databases of the world, data mining has the potential to reveal patterns regarding human health, the human genome, economics, consumer spending, marketing demographics, and a host of other areas. Many different data mining packages exist, taking a variety of approaches to finding patterns. Some use a detailed algorithm for finding patterns. Others use an approach that simulates evolution, so that solutions "evolve" from random processes. This approach is known as a *genetic algorithm* (GA).

Although they have been shown to be very successful, GAs only partially mimic evolution. They use a "roulette wheel" combined with a fitness function to simulate natural selection, with the more fit members of the population having a higher probability of selection. Thus, the more fit members of the population are more likely to be selected to produce offspring. In real-world reproduction, only half the genes from each parent are passed to the children. These genes may or may not be the good genes that make each individual more fit. Thus, a child may not necessarily be as fit as the parents, but over many generations the inferior genes will occur in smaller percentages of the population and could potentially be eliminated completely.

The amount of data used in real world data mining is significant. Having a technique capable of searching through all possible patterns and selecting those that are the most descriptive can be a very time-consuming process. For this reason GA approaches have been utilized for data mining. Genetic algorithms are capable of searching large solution spaces and are useful for finding solutions to problems that are difficult to find using more direct approaches. The strengths of GAs make them a good choice for data mining applications.

In data mining, the use of sets is critical to the analysis of the data. Yet, most set operations are omitted from current approaches. Furthermore, a translation or encoding of the sets is performed. Thus a fundamental piece of the problem is separated from the solution. The motivation for this paper is to develop a new tool for data mining that uses the sets as part of the solution process and takes advantage of set properties and operations. This tool is incorporated into an enhanced GA.

This paper details a novel data mining tool that combines set objects with an enhanced GA (Taylor, 2009). By performing direct manipulation of sets, the encoding process used in GAs can be eliminated. The sets can be directly used, manipulated, mutated, and combined, until a solution is reached. A rule can be achieved by combining sets together into a clause, separated by set operators (e.g., AND, OR, NOT, XOR, DIFFERENCE). Thus, a rule could look like: A AND B OR C AND NOT D $\rightarrow$ Z. This collection of sets and operators can be resolved to a single set, which is a subset of Z. In this way, rules can be constructed to describe a pattern within another set. Furthermore, the rules can then be collected into a set. The resulting solution is thus a set of rules, hereafter referred to as a description. Each generation of the GA is comprised of multiple descriptions.

Through generating generic set objects, the properties and advantages of sets can be used over and over regardless of the hierarchy—a set is the same as a set of sets. New sets can easily be created and manipulated by whatever algorithm is employed, and the need for multiple alternative data structures (such as arrays of bit streams to represent the population of a GA) can be reduced. This approach more closely mimics biology in that the "organisms" used in the genetic algorithm are multi-cellular. The smallest sets are combined together to represent a pattern in the data (a rule); these are then gathered into an even larger set to describe the data as a whole (a description). In organisms, cells combine to form tissues, which combine to form organs, which combine to form the organism. Thus, the inspiration for the use of set theory in this work is from biology.

The GA proposed by this paper introduces a second roulette wheel to the natural selection process. As in traditional GAs, the first roulette wheel is used to select the two parents to generate children. Here, the second roulette wheel is used to increase the probability of selecting the more favorable genes from each parent to pass to the child. The organism is the collection of rules; the genes are the individual rules itself. Thus, two collections of rules are selected, and then the rules from each description are evaluated and selected for the children, with higher probability going to those rules with higher fitness values.

We hypothesize that the approach presented in this paper will produce a robust data mining technique capable of finding obscure patterns within data. The technique is robust in that it can be used on data from a wide variety of domains without having to change any of the programming or to encode rules in a different way. Obscure patterns are not observable by humans because the patterns are very complex and the data sets are often very large. The addition of the second roulette wheel should help the process to converge more rapidly to a solution, while ensuring that the stronger rules are preserved.

2. BACKGROUND AND RELATED WORK

2.1 Set Theory

The most fundamental concept for this project is that of Set Theory. Set Theory can be defined as the mathematical science of the infinite (Jech, 2002). Set Theory studies the properties of sets, which are then used to formalize all the concepts of mathematics. The concepts related to this paper include membership ($a \in A$), cardinality (number of elements in a set), subset ($A \subseteq B$), compliment of a set, the intersection of two sets ($A \cap B$), the union of two sets ($A \cup B$), and the exclusive or operator (A XOR B).

2.1.1 *Crisp Sets.* The sets used in classical set theory are often called *crisp sets* because the size of the sets, the number of elements, and the identity of the individual elements are all very well defined. An example of a crisp set could be the number of books on a specific bookshelf. Membership in the set is easily determined—if a book is on the bookshelf, then it is in the set; books not on the shelf are not in the set. The number of elements in the set can be easily calculated by counting the books on the shelf. Questions about membership are also easily answered (e.g., Does the set contain *War and Peace*? or *What are the titles of all books in the set?*). Many real-world situations are difficult to define in terms of crisp sets.

2.1.2 *Rough Sets.* Because crisp sets are sometimes difficult to represent precisely, they are sometimes defined using two *rough sets*. A rough set approximates the upper and lower bounds of a set. The upper and lower bounds are sets themselves. For example if X is a set that is difficult to define as a crisp set, then A can be the lower bound for set X, and B can be the upper bound for set X. The definition for set A would be such that all elements of A are definitely elements of set X, i.e., $A \subseteq X$. Set B is given a much broader definition so that it contains at least some of the elements in X so that $B \cap X \neq \varnothing$ (Pawlak, 1994).

Rough sets are useful when dealing with uncertainty or ambiguity. If the membership of a particular element is unknown, a rough set can be created to include the unknown element. Rough sets can also be used to 'make space' for an element that does not fit well into a crisp set, but must be placed into a single set. For example, considering two bookshelves where one comprises entirely mathematics books and the other only philosophical books, the placement of a history book is an issue as it does not belong on either shelf. Regardless of which shelf the new book is placed on, the definition for the books on that shelf becomes a rough set.

2.1.3 *Fuzzy Sets.* Sometimes elements only partially belong to a set and the definition of membership becomes a little blurry, which is where *fuzzy sets* are useful. Fuzzy sets allow an individual element to belong partially to multiple sets at the same time, a concept that cannot be handled with crisp sets. Each element in a fuzzy set is given a membership value indicating the degree to which the element belongs in the set (Straccia, 1998). For example, with two sets of weekdays and weekends, to which set does Friday belong? Using fuzzy sets, Friday could be assigned to set weekdays with 60% membership and to set weekends with 40% membership. Thus, fuzzy sets allow for partial membership and can even be used to make claims such as a specific element is more of a member of a set than another element.

2.2 Data Mining

Data mining is the process of finding patterns that lie within large collections of data. Contrary to more traditional data analysis methods, which begin with a hypothesis and then test the hypothesis based upon the data, data mining approaches the problem from the opposite direction. Thus, data mining is discovery-driven rather than assumption-driven (Radivojevic et al. 2003). As the process searches through the data, patterns are automatically extracted.

In general, data mining objectives can be placed into two categories: descriptive and predictive (De Raedt et al. 2001). The goal of descriptive data mining is to find general patterns or properties that lie within the data set. This search often involves aggregate functions such as mean, variance, count, sum, etc. Descriptive data mining reports patterns about the data themselves. Predictive data mining, however, attempts to infer meaning from the data to generate a model that can be used to predict future data, which is often done by finding data elements with a given result and analyzing the properties those data elements have in common. The common properties should be a reasonable predictor for the given result.

Another important concept is the difference between supervised and unsupervised learning. Supervised learning takes place when the data have been pre-classified, i.e., the items in the data have already been placed into groups or have been assigned some value or result. For example, in a database about house values, each item in the database will contain values such as to the number of bedrooms, square footage, etc. In supervised learning, each house will also be assigned a monetary value. The goal of the data mining process is then to find the patterns that result in a given value. Unsupervised learning, on the other hand, occurs when the data are not pre-classified. In these cases, the data mining process cannot make value judgments. The process can find correlations within the data but is not able to make any inferences about what those patterns might mean. Thus, unsupervised learning is descriptive, whereas supervised learning is predictive. To make predictions, the data must be classified, or given value, by some outside source.

Some of the difficulties involved in data mining are problems with the data itself. When collecting data from different sources, the various sources often have different formats for the data, collect different information about the data, and have different protocols about the data. For example, one set of records might contain a person's age, whereas a similar set of records from another source might not. Further compounding the problem, even within the same source, the data can be erroneous or even missing. Perhaps the ages for some, but not all, of the people are recorded in the data. This situation can make finding patterns in the data very difficult when the data are incomplete or inconsistent. Thus, a major task in data mining is in how to handle these anomalies in the data that are not actually part of the data.

There are so many different approaches to data mining that it is difficult, if not impossible, to list them all, and the different techniques vary as widely as the data they are used to analyze. Some data mining techniques implement neural networks, clustering algorithms, regression modeling, genetic algorithms, data visualization, parallel data mining, and many more different approaches (Yin et al. 2004; Arumugam & Rao, 2005; Shah & Kusiak, 2004; Choenni, 2000; Ratanamahatana, 2008). Rather than describe the various approaches, this paper will focus on the techniques directly relevant to the approach used in this paper.

2.2.1 *Decision Tables.* Decision tables are very similar to tables in a database.

Each line in the table is called a *tuple* and represents one specific item such a house, an employee, a business, a car, etc. Each column in the data is an *attribute* and is used to describe each tuple in the table. What distinguishes a decision table from a database table is that the decision table has some decision associated with each tuple. The attributes can be thought of as a condition, with the decision being associated with that condition. Table 1 is an example of a decision table indicating the conditions for which patients will receive different percentages of reimbursement from their health insurance provider. There are 8 tuples, each representing a different condition under which a patient might apply for a reimbursement from the insurance company. There are three types of visits, namely, office, hospital, and lab. The other two attributes only have two possible values of yes or no. Each tuple can be described by using the values of the attributes. The decision in this table is the reimbursement attribute. The combination of an attribute with a value is called a feature (Kohavi and Provost, 1998). For example, (type_of_visit, office) is a feature – it specifies that the type of visit was an office visit. Some of the combinations of attributes in Table 1 are missing, such as a hospital visit that is also a participating physician. In order to cover all possible combinations of features in the first three attributes of the table, only 12 tuples would be required. Real world data, however, can have millions or more tuples.

TABLE 1

Decision table indicating conditions for reimbursement from insurance.

Tuple #	Deductible Met	Type of Visit	Participating Physician	Reimbursement
1	yes	office	yes	90
2	yes	office	no	50
3	yes	hospital	no	80
4	yes	Lab	no	70
5	no	office	yes	0
6	no	office	no	0
7	no	hospital	no	0
8	no	Lab	no	0

2.2.2 *Rule Induction.* Rule induction is the process of taking the data and searching for meaningful patterns that can be described in terms of features. The result is a set of rules that describe the patterns in the data. Each rule consists of two parts, namely, the *antecedent* and the *consequent* (Siler, 2005). The antecedent (left-hand side) of the rule is the condition that must be met for the rule to be applicable. The consequent (right-hand side) of the rule is the action or decision that follows if the antecedent is true. If the antecedent is true, then the consequent follows. A sample rule from Table 1 is:

```
(deductible_met, no) → (reimbursement, 0%)
```

This rule specifies "*if the deductible is not met then there is 0% reimbursement.*" The example rule describes tuples 5, 6, 7, and 8 as to how those conditions are reimbursed. The rule does not cover tuples 1 through 4. More rules must be induced to cover the first four tuples. Thus, by generating a proper set of rules, all the patterns in the decision table can be described.

2.2.3 *Rules with Set Operators.* A consequent can be described by a single set in the antecedent but this occurs rarely in real-world data. A consequent usually requires a combination of sets to provide an adequate classification that does not include members from other consequents. These sets have to be separated by set operators. The set operators describe how the sets relate to one another. Without set operators, the meaning of the rule is ambiguous. For example:

```
(deductible_met, yes) (participating_physician, yes) →
          (reimbursement, 90%)
```

does not have a set operator. From Table 1, it is apparent that for the consequent to be true, a tuple must be a member of both sets in the antecedent, requiring an AND operator between the two sets. If an OR operator had been used, then tuples 1 through 5 would be members of the antecedent. Tuples 1 through 4 are members of (deductible_met, yes) and tuple 5 is a member of (participating_physician, yes). Regardless of the antecedent, only tuple 1 would be a member of the consequent. The AND operator makes the rule correct. There are times when the OR operator is more appropriate:

```
(age, 15) OR (age, 17) → (age_group, teen)
```

The four set operators used by the approach in this paper are:

- Y AND Z – must be a member of both sets Y and Z.
- Y OR Z – is a member of at least one of the sets Y and Z.

- Y XOR Z – is a member of only one of the sets Y and Z.
- NOT Z – is not a member of Z.

Using sets and set operators, complex statements can be constructed describing the data. This is necessary because the patterns within the data rarely allow for a single set to be used as the antecedent for a rule. Statements can be constructed with sets and set operators that allow rules to make unique classifications.

2.3 Genetic Algorithms

Genetic algorithms (GAs) are based upon the evolutionary principles of natural selection, mutation, and survival of the fittest (Dulay, 2005). The GA approach is to generate a large number of potential solutions in a search space and to evolve the solutions to the problem. One of the big keys to a successful GA is in the development of a good fitness function. The fitness function is how each potential solution is evaluated by the algorithm, and is in essence how the problem to be solved by the algorithm is defined. For example, if the purpose of the GA is to design a car, then the fitness function will provide a means for evaluating the fitness of any car design.

When developing a GA, one should decide how each solution will be represented in the algorithm. For simplicity, a string of bits is most often used. The bits can be used to represent any part of the solution. Taking the car example, some bits might represent the color of the car, others the size of the wheels, and others the gas mileage of the car. It is the responsibility of the fitness function to understand what the bits mean and how to use them to evaluate the fitness of each potential solution.

2.3.1 *Process of Genetic Algorithms.* The GA begins by generating an initial population of potential solutions, which could be random. Each member of the population is then examined and its fitness is evaluated. Next, the next generation is produced from the current generation. There are many ways of generating the next generation, but the two most popular techniques involve crossover and mutation. In crossover, two members of the population are chosen at random with higher probability given to the more fit members of the population. These two members are then combined to produce two offspring. This grouping is usually performed by selecting a position in the bit sequence and exchanging the two sequences after that position (the crossover point). For example, given the following two members of a population: 11101010 and 10010100, if these were crossed over at position 4, the resulting members would be: 11100100 and 10011010. The result is two new members in the next generation. In theory, these two new members should be reasonably more fit because they resulted from fit members in the previous generation. Each member of the population is assigned a biased probability of selection. Because of this increased probability of selection, the most fit members of a population are more likely to be selected for crossover. However, there is always a possibility that a less fit member will be selected instead.

Usually, the crossover process continues until the size of the next generation is the same as the population size of the previous generation. After crossover has taken place, mutation is then applied to each member of the population. A typical mutation function is to assign a probability for flipping (0 becomes 1, or 1 becomes 0) each bit. For instance, if with a 10% mutation value, each bit of each member of the population has a 10% chance of being flipped. After mutation is completed, each member of the new generation is evaluated for fitness and the process repeats for another generation. This process of evolving new populations continues until some criteria is met, such as: (1) the overall fitness of the population reaches a certain value, (2) the overall fitness over several generations does not change more than a specified value, or (3) after a certain number of generations have been reached.

There are many different ways to perform crossover and mutation, and many other genetic operators have been used in GAs. Regardless of the details, the overall process remains the same: generate an initial population, evaluate the members of the population, generate a new population based upon the more fit members of the previous generation, and repeat the process until a certain stop criteria is achieved.

2.3.2 *Applications of Genetic Algorithms.* Genetic algorithms are powerful search tools (Goldberg, 1989). Genetic algorithms are capable of pouring through a large number of potential solutions to find good solutions. Scheduling has been an area where GAs have proven useful. The GA searches the space of potential schedules and finds those schedules that are most effective and maximize the desired criteria (such as minimizing idle time). For example, GAs are used by some airlines to schedule their flights (Dulay, 2005). An application of GAs to a financial problem (tactical asset allocation and international equity strategies) resulted in an 82% improvement in portfolio value over a passive benchmark model, and a 48% improvement over a non-GA model used to improve the passive benchmark (Dulay, 2005). Genetic algoriths have also been applied to problems such as protein motif discovery through multiple sequence alignment (Mendez et al. 2003), and obtaining neural network topologies (Taylor & Agah, 2006).

2.4 Confusion Matrices and Fitness Metrics

Two common measures when discussing the results of a data mining model are *sensitivity* and *specificity*. Sensitivity (true positive rate) measures the percentage correctly identified as positive out of the total number of positives. Specificity (true negative rate) measures the percentage correctly identified as negative out of the total number of negatives. These measures can be presented using a confusion matrix (Hamilton, 2005), as shown in Table 2, where *a* are negative tuples classified as negative, *b* are negative tuples classified as positive (false positives), *c* are positive tuples classified as negative (false negatives), and *d* are positive tuples classified as positive (Kohavi and Provost, 1998).

TABLE 2

A 2 × 2 confusion matrix

		Predicted	
		Negative	Positive
Actual	Negative	a	b
	Positive	c	d

Using the confusion matrix, a number of fitness metrics can be defined:

```
Accuracy = (a+d)/(a+b+c+d)
Sensitivity (True Positive Rate) = d/(c+d)
Specificity (True Negative Rate) = a/(a+b)
Precision = d/(b+d)
False Positive Rate = b/(a+b) = 1 - Specificity
False Negative Rate = c/(c+d) = 1 - Sensitivity
```

Coverage is another metric which is used when discussing the fitness of a data mining model. The coverage of a model is the proportion of the data for which there is a rule. Thus, a model which has 90% coverage provides rules that classify 90% of the tuples. Coverage only means that a classification is made; coverage does not, however, measure the accuracy of the classification. Using these seven metrics, the results of a data mining model can be evaluated.

2.5 Genetic Algorithms in Data Mining

There are currently two different approaches to rule discovery in data mining using genetic algorithms: the Michigan approach and the Pittsburg approach (Au et al. 2003). The *Michigan* approach represents a rule set by the entire population, with each member of the population representing a single rule. The *Pittsburg* approach represents an entire rule set with a single chromosome, thus each member of the population represents an entire set of potential rules for describing the data.

Data sets are either single-class or multi-class, which refers to the number of possible values for the decision class. If there is only one decision value, then all rules will describe that value—thus it is a single-class set. In a multi-class set, there are multiple decision values in the data (but each rule describes a single value). The Michigan method is useful for multi-class problems but suffers in that no way exists to ensure a high coverage of the data by evaluating a single rule at a time. The Pittsburg approach has been used to learn rules for a single class. To induce rules for multiple classes, the algorithm must be run multiple times.

In their experiments, Au et al. (2003) used the Pittsburg approach successfully. The authors compared the results of their data mining model, DMEL, developed by a GA, to the results developed by C4.5, a well-known data mining algorithm that uses decision trees (Quinlan, 1993). The experiment included seven different data sets that were diverse in nature. In each instance, the GA produced more accurate results than C4.5, ranging from 0.3% to 13% improvement in accuracy.

In their work, Flockhart and Radcliffe (2005) concluded that GAs are well suited to undirected data mining but can also be used for directed data mining. Undirected data mining is the most common form, where the program simply looks for patterns and describes them. In directed data mining, the user specifies the type of information in which they are interested. Using the Michigan approach, GA-MINER was able to find interesting, non-trivial rules within the data sets used for the experiment. The authors also posed an idea for hypothesis refinement in which the user can seed the genetic algorithm with a rule or set of rules which the GA can use as an initial population. The GA can refine the initial hypotheses to produce a better model.

3. RESEARCH METHODOLOGY

This project is an effort to provide a technique for data mining that essentially combines the Michigan and Pittsburg approaches, thus performing both methods at the same time. The technique proposed in this paper uses a combination of set theory and GAs. The desired outcome is an approach that can generate a set of rules to describe any data set, without using any of the conventional data mining techniques. The approach has been dubbed *Arcanum* (Latin for secret or mystery). Similar to the Pittsburg approach, each member of the population in Arcanum represents an entire set of rules that describe the data. A population of 50 would contain 50 different sets of rules for describing the data. Similar to the Michigan approach, Arcanum is able to find patterns for each decision class at the same time. In this manner, Arcanum combines both techniques to find a set of rules that describes every decision class in the data with only a single run. A set of rules is referred to as a description throughout this paper.

3.1 Generating Sets from Input Data

Arcanum uses a decision table as input. Each unique feature in the decision table is used to create a set. Thus (deductible_met, yes) and (deductible_met, no) are two sets that would be generated from Table 1. Arcanum allows multiple columns of the decision table to have the same attribute name, which is useful when multiple features exist for each tuple with respect to a particular attribute. For example, objects often have multiple colors. Generating a separate attribute for each color (e.g., Color1, Color2, etc.) limits the ability to find patterns in the data because matches occur only within the same attribute. In other words, (color1, black) would not match (color3, black) because they are from two different attributes, even though the value of the attributes are the same. By having three columns in the table called color, however, each tuple can have multiple values for the attribute and Arcanum can still match the colors regardless of the order in which the colors appear in the table.

Each feature set contains the tuples to which it applies. From Table 1, the feature set (deductible_met, yes) would contain tuples 1, 2, 3, and 4. Each unique tuple can then be described by using a logical AND of the appropriate sets. For instance, in Table 1, tuple 1 can be uniquely identified as:

```
(deductible_met, yes) AND (type_of_visit, office) AND
        (participating_physician, yes)
```

The last feature of tuple 1 is unnecessary because the first three features uniquely identify it. Arcanum also creates a Universal Set. The Universal Set includes all the tuples. Arcanum ignores any other sets that are equal to the Universal Set because the knowledge gain from such sets is negligible and often obvious from a cursory examination of the data.

3.1.1. *Numerical Attributes.* When dealing with numerical attributes, the values rarely match exactly. For example, (temperature, 98.5) and (temperature, 98.6) would be separate sets even though they should probably be combined into a single set. To deal with this, Arcanum uses a binary discretization method to create partitions for numeric attributes. This discretization is performed locally on each attribute, meaning that the discretization process is performed on each numeric attribute independently of any other numeric attribute.

The discretization begins by creating a list of break points between features in the decision table. For Table 1, the only numeric attribute is the reimbursement attribute. The first break point occurs halfway between the smallest value and the next largest value. The next break point occurs between the next two larger values and so on. In Table 1, the smallest value is 0 and the next largest is 50, thus the first break point is 25. The next break point occurs between 50 and 70, then between 70 and 80, and then between 80 and 90. Thus, the list of break points for Table 1 includes 25, 60, 75, and 85.

Using the list of break points, intervals are then created. Two intervals are created for each break point. For Table 1 there will be eight intervals to partition the reimbursement attribute, because there are 4 break points. For any break point, the two intervals are from the smallest value to the break point and from the break point to the largest value. Using the break point of 25 from Table 1, the first interval is [0, 25] and the second interval is (25, 90]. These two intervals are then used to generate sets. The set (reimbursement, [0, 25]) contains tuples 5, 6, 7, and 8. The set (reimbursement, (25, 90]) contains tuples 1, 2, 3, and 4. Each remaining break point is then used to partition the attribute and generate sets.

Using this discretization process, Arcanum can then induce rules based on these intervals of values rather than trying to use each separate value.

3.1.2 *Uncertainty.* In the real world data, often there are times when an attribute value for a tuple is not known, leading to uncertainty in the data. Arcanum allows two special symbols in a decision table to denote uncertainty. The "?" symbol is an attribute value for a tuple indicating that no value exists for the attribute for the respective tuple, and that the attribute value should be ignored for the tuple. This symbol can be used in the case of multiple values for the same attribute, such as the color in previous examples. If there are three columns for color, but a particular tuple only has two colors, the "?" can be placed in one column to tell Arcanum to ignore that column for that tuple. The "?" can also be used to create more certainty in Arcanum. By ignoring unknown attribute values, only those values that are known are used to induce rules. Therefore, the resulting rules are certain because they make no assumptions about the unknown values, as they are ignored.

The "*" symbol is an attribute value for a tuple that also indicates uncertainty; it is handled differently, however. When the "*" is used, Arcanum adds the tuple to all sets involving that attribute. The "*" is treated as if it contains every known value for the attribute. For example, if a ninth tuple were added to Table 1 and contained a "*" for the first attribute, then the tuple would be added to the sets (deductible_met, yes) and (deductible_met, no). This symbol allows Arcanum to handle uncertainty by using possible values for the attribute when inducing rules, but means that the rules are less certain because assumptions have been made about some attribute values.

Both uncertainty symbols can be used within the same decision table, allowing the user to specify that rules involving particular attributes must be certain, but rules involving other attributes can be less certain. In cases where there are no missing attribute values, neither symbol is necessary.

3.2 Modes of Operation

Arcanum can run in either the directed mode or the undirected mode. In the directed mode, the user specifies which attributes in the decision table are of interest. Only the attributes designated by the user will be used as consequents for rules, i.e., the rules induced by Arcanum will only describe those attributes. All other attributes can be used in the antecedent of rules, but not as consequents. This mode is used when the user knows the type of information that is being looked for. In the undirected mode, Arcanum will use all attributes as consequents of rules. Essentially, Arcanum will try to find patterns for every feature in the decision table. By default, Arcanum operates in the undirected mode, attempting to describe the data set as completely as it is possible.

3.3 The Process

After generating the sets from the input decision table, Arcanum employs a genetic algorithm to produce a description of the data. The outline of the process in Arcanum is as follows:

1. Seed the initial generation of descriptions
2. Evaluate rules
3. Evaluate descriptions
4. Generate next generation
5. Manipulate rules
6. Repeat steps 2 through 5 until stop criterion is reached

3.3.1 *Seeding the Initial Generation.* The initial generation of descriptions can be seeded in one of two ways. Arcanum can randomly generate descriptions or the user can provide Arcanum with an initial description that Arcanum will then attempt to refine through the genetic algorithm. By default, Arcanum will randomly generate an initial population of descriptions. For each description, Arcanum will generate a random rule for each feature of interest. Each of the initial rules will contain only one set in the antecedent, which will cover any possible one-to-one relationships between features.

If the user chooses to provide an initial set of rules, then Arcanum will use these rules to seed one-third of the initial population. Another third of the population will be seeded with randomly modified versions of these rules. The last third of the population will be composed of randomly generated descriptions. The decision to split the initial population into thirds was made to provide a diverse initial generation. The chances of finding beneficial crossovers should be increased while still allowing for some of the initial rules to survive.

3.3.2 *Evaluating the Rules.* Usually, Sensitivity (true positives) is associated with Specificity (true negatives) when evaluating a classification method. The technique discussed here does not deal with negative classification. If a tuple does not match a rule, then we cannot conclude that the tuple is not part of the class. We can conclude that the tuple is not part of the subset of the class described by the rule. A different rule might show that the tuple is part of the class, just a different subset. To determine if it does not belong to a class, the tuple must be compared with all the rules that describe that class. Therefore, specificity does not apply to individual rules. Precision has been chosen as a substitute because it provides information on the number of exceptions to a rule.

Each rule in each description is assigned a value based on the sensitivity, also known as *recall* and *precision* of the rule. Precision identifies the frequency with which the rule is true. Recall is the proportion of the rule consequent, the decision class, which is classified by the rule. A recall of 75% means that three-fourths of the rule's consequents are classified by the antecedent. The fitness value of the rule is calculated as follows:

$$(weight_{precision} * precision) + (weight_{recall} * recall)$$

Each rule can have a fitness value ranging from 0 to 1. The weights are used so that preference can be given to models with high precision or high recall, depending on the needs of the user.

3.3.3 *Evaluating the Descriptions.* When evaluating a description, the purpose is to measure the collective results of the rules contained in that description. It is possible to have a description that contains very good rules that only classify a small portion of the data set and make no classification for a large portion of the data. Also possible is having a description that classifies all the data, while the classifications are inaccurate. To ensure that descriptions provide classifications for a large part of the data, and that the classifications are correct, the accuracy and coverage metrics are used. The fitness of a description is the weighted sum of its accuracy and coverage. The accuracy of a description is the percentage of correct classifications (true positives and true negatives). An accuracy of 25% means that only one-fourth of the classifications in the description are correct. The coverage of a description is the percentage of the data set for which a classification is made, regardless of whether the classification is correct. A coverage of 80% means that 20% of the data remains unclassified by the description. The fitness value of the description is calculated as follows:

$$(weight_{precision} * accuracy) + (weight_{recall} * coverage)$$

Thus, each description can have a fitness value ranging from 0 to 1. Notewothy is that this equation uses the same weights specified for precision and recall because accuracy reflects the precision of the entire model, while coverage reflects the recall of the entire model.

3.3.4 *Generating the Next Generation.* Based upon the fitness values assigned to the descriptions and rules, a new generation of descriptions is generated. Creating a new generation of descriptions is done through a process of selecting two parent descriptions and combining rules from each of them to produce two child descriptions. The children then become part of the new generation. Typically, this selection is done with a roulette wheel approach. Arcanum uses two roulette wheels, one within the other, to produce the children.

First, the fitness values of all descriptions are normalized, providing a selection probability for each member of the population. The higher the fitness value of a description, the more likely it is to be selected as a parent for the next generation. Two descriptions are then selected based upon these weighted values. The two descriptions are then used as parents. Each child will receive one-half of its rules from each parent, which is done through another roulette wheel process similar to the one used to select the parents. Rules with higher fitness values are more likely to be selected to pass on to the child. A child cannot receive the same rule from the same parent more than once; the same rule, however, can potentially be received once from each parent, resulting in a duplicate. This process is similar to the way genes are passed in biology.

Using this process, two children are produced from the selected parents. After the children are produced, two new parents are selected, which can result in the same parents being chosen. Because of the weighted roulette wheel process used in selecting parents and rules, there is a chance of producing children in the next generation that have the exact same genes. The probability of generating children with the same genes is inversely proportional to the number of genes being received from each parent. In other words, the more genes each child receives from a parent, the less likely it is that their children will have the exact same genes. Although the production of duplicate children could indicate convergence on a solution, it is not necessarily desirable—just because two descriptions have the same rules does not mean that those are the best rules. Some variance is still necessary to ensure that some possibilities are not being overlooked. This variance is handled in rule manipulation.

3.3.5 *Manipulating the Rules.* The rule manipulation step simulates the role of mutation in evolution. Each rule of each description in the new generation has a chance of undergoing a random mutation. The probability for any given rule undergoing a mutation is equal to one minus the fitness of the rule. Thus, the higher the fitness score of the rule, the less likely it is to mutate. Only one operator can be applied to a specific rule or description per generation. Thus, if one mutation is applied, then none of the others can be applied to that rule or description until the next generation. The possible operations are:

- ChangeOperator: Randomly change one of the set operators to a different operator
- ChangeSet: Randomly change one of the sets in the antecedent to a different set
- ComplicateRule: Add AND with a random set to the end of the rule. Chance of selection is inversely proportional to precision. The lower the precision, the higher the chance that ComplicateRule will be applied to the rule. When precision is low, it indicates that there are many exceptions to the rule – it is too general. Adding to the antecedent can help make the rule more specific and remove some of the exceptions, thus improving precision.
- SimplifyRule: Delete the end of a rule after a randomly selected position. Chance of selection is inversely proportional to recall. The lower the recall, the higher the chance that SimplifyRule will be applied to the rule. When it is low, recall indicates that the antecedent excludes many members of the consequent —it is too specific. Removing operators and sets from the end of the antecedent can make it more general and potentially include more members from the consequent that were previously being excluded.
- AddRule: Unlike the other mutation operations, which affect only rules, the AddRule operator applies to a description. Each description has a chance for having a new rule added. This chance is inversely proportional to the coverage of the description. The lower the coverage, the greater is the chance that a new rule will be added to the description to help improve the coverage. The new rule will consist of a single random set as the antecedent and a single random set as the consequent.

3.3.6 *Linear Dropping.* After a solution has been reached, each rule in the solution will undergo a process called *linear dropping*. Due to the random manner in which rules are constructed in this process, it is possible that they might contain extraneous information in the antecedents of the rules. For example, a rule from Table 1 could be:

```
(deductible_met, yes) AND (type_of_visit, lab) AND
(participating_physician, no) → (reimbursement, 70%)
```

In reality, only the first two sets are needed; the addition of AND (participating_physician, no) does not add any new information to the rule. Therefore, it can be dropped from the rule.

To perform linear dropping, the process will start with the set closest to the consequent in the rule. In the example, the process would start with (participating_physician, no) because this set is furthest to the right (closest to the consequent). This set becomes the drop set. The rule will then be evaluated as if the drop set were no longer part of the rule. If the result from the evaluation of the new antecedent is the same as the result of the old antecedent, or if the result is a subset of the consequent, then the antecedent is changed to no longer contain the drop set (including the operator to the left of the set). The process continues by selecting a new drop set, the set immediately to the left of the previous drop set.

Once the linear dropping process is complete, each rule should contain only the minimal amount of information required by the antecedent to describe the consequent, or a subset of it.

4.0 EVALUATION

To determine the performance and efficiency of Arcanum, the technique had to be evaluated. The evaluation required: a group of data sets to be mined, other published results for the data sets, and some metrics that could be compared between the published results and the results from Arcanum. From a review of the literature, the most common metrics used appeared to be either accuracy or error. The accuracy metric is calculated by Arcanum, and the error metric is calculated as (1 – accuracy).

4.1 Data Sets

The specific data sets for experimentation were selected from the University of California at Irvine (UCI) Knowledge Discovery in Databases (KDD) archive (Hettich and Bay, 1999). The UCI KDD archive contains a large collection of data sets that have been used by several research groups. Using this collection provides two distinct advantages: (1) access to several diverse data sets from a single source; and (2) other researchers have published the results of their data mining on these data sets, thus providing a means of measuring the success of Arcanum by comparing it with those results.

When selecting the data sets to be used for evaluating Arcanum, several factors were considered–namely, selecting six types of data sets: one with all nominal attributes, one with all numeric attributes, and a mix of nominal and numeric attributes; each of these with and without some missing values. Higher priority is given to those sets with missing attribute values, as most real-world data sets have missing values. Numeric attributes are also very important because Arcanum uses a discretization method when dealing with them.

Another important factor in choosing data sets is the volume of available literature with respect to the data set. When more results can be obtained from other researchers using the same data set, more comparisons can be made, allowing more insight into the results from Arcanum. Even if the accuracy of the results are similar, it is still of interest to compare the actual results to see if the resulting rule sets are similar.

4.2 Separating Training and Testing Data Sets

Separating the data sets into two separate sets is crucial: one to be used for training and the other to test the results obtained from training. The data sets in the UCI KDD archive are not split into these separate sets, thus some preprocessing must take place before they can be used in Arcanum. Each data set selected from the UCI KDD archive is run using a K-fold cross-validation method, with $K = 10$ (Souza et al. 2002). Thus, each data set is separated into 10 subsets. For each run of the algorithm, nine subsets are used for training and the remaining set is used to test the resulting data model. In the next run, a different testing set is used, which is done 10 times, so that each subset is used once as a test set. The results for all 10 runs are averaged together for the final evaluation.

4.3 Metrics

A number of metrics were selected to measure the effectiveness of the overall description (accuracy and coverage), as well as the effectiveness of each individual rule (precision and recall). Because each rule indicates only membership and not lack of membership, the measurement of negative classification can be done only for the entire description. The metrics measured and reported in this paper are:

- Accuracy: Measured for the entire description, this is the percentage of correct classifications (true positives and true negatives) using the derived rules.
- Coverage: Measured for the entire description, this is the percentage of the data set for which a classification is attempted.
- Precision: Measured for each rule, this is the percentage of examples from the antecedent of the rule that are also in the consequent of the rule. The inverse (1 – Precision) is the percentage of exceptions to the rule.
- Recall: Measured for each rule, this is the percentage of the decision class that is covered by the rule.

4.4 Comparisons

Two other approaches have been implemented based on of the set theory platform to help evaluate the performance of Arcanum by providing their own descriptions of data sets, as well as run-time comparisons. The approaches can also be used to generate rules to be introduced into Arcanum to test the rule-seeding portion of the program. The two approaches are Learning by Examples Module (LEM2) and Markov Chain Monte Carlo (MCMC).

4.4.1 *LEM2.* The LEM2 algorithm (Learning by Examples Module) is a proven data mining technique (Chmielewski & Grzymala-Busse, 1996). The pseudo-code for the algorithm was available, so it was implemented to test the set theory portion of Arcanum's code, and to generate benchmark rules. LEM2 can also serve as a run-time benchmark because it follows a heuristic algorithm rather than the GA used by Arcanum. This LEM2 implementation has been used on several data sets, including a non-trivial data set involving protein sequences, with success.

4.4.2 *Markov Chain Monte Carlo.* Markov Chain Monte Carlo (MCMC) is a technique used to gather random samples from a probability distribution (Ridgeway & Madigan, 2002). First, a Markov model of the data is constructed. In this specific implementation, the Markov model is derived from the training data set. Each attribute is given an equal probability of selection, thus if there are five attributes, then any attribute has a one out of 5 chance of being selected. Within each attribute, the features of the attribute are assigned probabilities based on their frequency within the attribute. Therefore, the features that occur more often have a higher probability of selection.

To generate rules in this manner, a random attribute is selected, and then a feature is randomly selected from the attribute. Another attribute and feature is then selected and added to the rule. This process continues until the decision attribute is reached, completing the rule. This random rule is then tested against the data set to measure the precision and recall of the rule. If a rule falls below the threshold value for either metric, then the rule is discarded. If a rule is accepted, then the rule is added to the rules in the description and another random rule is generated. This process continues until a pre-specified (by the user) number of rules have been sampled.

When used for data-mining, this technique is slow and not nearly as accurate as LEM2. Nevertheless, the random sampling can sometimes find additional patterns within the data that LEM2 ignores because the examples have already been covered by a different rule. Thus, the MCMC implementation can be useful for generating rules that can be used to provide seed rules for Arcanum, which MCMC can then improve upon. The results from MCMC can also serve as a lower limit for measuring the success of Arcanum—it should perform at least as well as MCMC.

4.5 Efficiency

One common measure of GA efficiency is the number of generations required to reach a solution. By the nature of genetic algorithm, however, the number of generations cannot be directly controlled. Sometimes an upper limit is placed on the number of generations, so that the limit is exceeded the program terminates. The number of generations required by the GA is affected by several parameters, such as the frequency of mutation, crossover method, encoding method, calculation of the fitness function, and size of the population. Of these five parameters, only the frequency of mutation and the size of the population can be adjusted in Arcanum. The other three parameters are all integral to the technique utilized by Arcanum, as described in a previous section. The population size should be high to have more diversity within the population. With more diversity comes an increasing chance of finding better solutions. As the population size increases, however, so does the amount of physical memory required to represent it in the computing platform. The frequency of mutation has to be adjusted by evaluating the performance of Arcanum on some trial data sets to determine which frequency of mutation produces the fewest number of generations before a solution is reached.

5.0 EXPERIMENTAL RESULTS

Testing of Arcanum was done in two phases. The first phase was to determine if Arcanum could be used to perform data mining. The second phase tested the ability of Arcanum to perform hypothesis refinement, i.e., taking a previously generated data model and attempting to improve it. This section contains the results and discussion of both phases of testing.

5.1 Data Sets

To test the data mining ability of Arcanum, 13 data sets were chosen from the University of California, Irvine, Knowledge Discovery in Databases repository (Hettich & Bay, 1999). These data sets were chosen because of the availability of published results from other data mining techniques. The sets used, and their characteristics, are listed in Table 3. As shown, most of the data sets were comprised of entirely numeric data, and thus requiring discretization. Only three of the data sets contained any nominal, or text, data. Five of the sets were missing at least some attribute values. Using these data sets, Arcanum was tested on both nominal and numeric attributes, and each of those was tested both with and without any missing attribute values.

5.2 Data Mining

The data mining results for Arcanum and the other techniques are included in Table 4. The value reported for Arcanum is the accuracy of the model multiplied by the coverage of the model. All of these results from Arcanum used a population size of 500, a maximum of 50 generations, a precision weight of 70%, and a recall weight of 30%. Thus, accuracy was more highly valued in the models than coverage. GAssist is a genetic algorithm for data mining, developed using the Pittsburg approach (Bacardit & Garrell, 2007). J4.8 is a Java implementation of the C4.5 algorithm and is included as part of the WEKA software (Witten & Frank, 2005). These two techniques were ideal for comparison because C4.5 is still considered one of the very best data mining algorithms, and GAssist is a genetic algorithm used for data mining.

TABLE 3

Description of data sets used for testing Arcanum.

Data Set	Attribute Values	Missing Attribute Values
Breast.C	Mostly nominal	None
Breast.W	Entirely numeric	Very few
Cleve	Nominal and numeric	Very few
Glass	Entirely numeric	None
Hepatitis	Entirely numeric	Many
House-votes	Entirely nominal	Many
Iris	Entirely numeric	None
Ionosphere	Entirely numeric	None
Liver.Bupa	Entirely numeric	None
Lymph	Entirely numeric	None
Pima.Diabetes	Entirely numeric	None
Wdbc	Entirely numeric	None
Wpbc	Entirely numeric	Very few

TABLE 4

Comparison of Results from Arcanum to GAssist and C4.5.

Data Set	Arcanum	GAssist	J4.8 (C4.5)	
Breast-c	64.8	70.5	75.5	
Breast-w	95.3	N/A	95.0	+
Cleve	57.6	80.4	76.8	
Glass	43.0	66.8	66.8	
Hepatitis	48.6	89.8	83.9	
House-votes	95.9	97.1	96.3	+
Iris	94.6	95.3	96.0	+
Ionosphere	67.1	92.0	91.5	
Liver.Bupa	49.8	62.4	N/A	
Lymph	64.3	80.8	77.0	
Pima.Diabetes	74.6	74.7	73.8	+
Wdbc	92.7	94.3	N/A	+
Wpbc	50.3	75.3	N/A	

In 5 of the 13 data sets examined, Arcanum was able to develop a data model that was comparable to GAssist and C4.5 (marked with the "+" symbol). In these five cases, Arcanum was within 2% of the accuracy of the other methods, and in two of the cases obtained a better data model than C4.5. Although certainly some improvement must be done before Arcanum can compete on all levels with these algorithms, the results show the potential for Arcanum. When compared with GAssist, Arcanum gains further appeal because it has significantly fewer parameters than GAssist. A paper on GAssist reported the use 19 parameters for GAssist (Bacardit & Butz, 2004). Arcanum has only, at most, eight parameters if counting the type of model validation scheme used. For the purposes of this paper, only four parameters were ever varied: the size of the population, the maximum number of generations to be run, the weight placed on precision, and the weight placed on recall.

As previously stated, Arcanum parameters focused heavily on precision over recall and thus favored models with high accuracy, even though the coverage of the model might be low. In the cases in which Arcanum did not perform as well as GAssist and C4.5, the predictive accuracy of Arcanum's models was high but had limited coverage. In other words, when the model made a prediction, the estimate was highly accurate, but the model failed to make any prediction for a large percentage of data. This result indicates that one of the areas of improvement for Arcanum might be in improving the fitness function to help ensure greater coverage in the resulting data models.

No readily apparent way exists to stereotype the data sets upon which Arcanum would be successful. The data sets upon which Arcanum produced successful results included sets with nominal attributes, sets with numeric attributes, sets that had large numbers of missing attribute values, sets with only a few missing attribute values, and sets with no missing attribute values at all. From the results discussed in this section, apparently Arcanum has the potential to perform data mining on a level comparable with other successful techniques, such as GAssist and C4.5. The results indicate that such performance is possible using the multi-tiered genetic algorithm technique described in this paper.

5.3 Hypothesis Refinement

With the data mining proof of concept phase completed, Arcanum was then tested on whether it could be used to improve a previously generated data model. During this phase, data models from different techniques were used as a starting point for Arcanum. These data models included some generated by Arcanum in the data mining stage, some models generated by the LEM2 algorithm, and a data model generated by C4.5, using the J4.8 implementation included in WEKA. The six data models chosen from Arcanum represent the two worst data models from Arcanum: Glass and Hepatitis; three models with very good results: Breast-w, House-votes, and Wdbc; and one model in the middle: Breast-c. The refinements were performed while keeping the same population size and number of generations, 500 and 50, respectively. Each model was run with equal weighting of precision and recall. Table 5 shows the results after this refinement. The results for the Glass data set were actually significantly lower using a 50-50 weighting scheme (the result was 34.4%), so it was run again using a 70-30 weighting scheme to see if there would be improvement. The result shown for Glass in Table 5 is the result after the 70-30 weighting scheme.

TABLE 5

Results of hypothesis refinement on Arcanum data models

Data Set	Original	After Refinement
Breast-c	64.79%	71.47%
Breast-w	95.28%	74.93%
Glass*	43.0%	51.48%
Hepatitis	48.63%	51.94%
House-votes	95.88%	95.67%
Wdbc	92.69%	91.12%

Whereas only half of these data models were improved after refinement, the results show a great deal of potential in using Arcanum for hypothesis refinement. After refinement, the Arcanum model for the Breast-c set yielded better results than the GAssist model, as the Arcanum model was refined to 71.47% versus the 70.5% of GAssist. The three models that failed to improve were already very good, comparable with both C4.5 and GAssist, with the Breast-w model already better than the C4.5 result.

Of the six models originally generated by Arcanum, the program was able to improve the quality of its own data models through hypothesis refinement. One model was improved to the point where it was comparable with both GAssist and C4.5. Of the models that failed to improve, each was already comparable with GAssist and C4.5, and one was even better than C4.5. The results indicate that hypothesis refinement is a useful tool and certainly has the potential to take a data model and make it better.

To further test the abilities of Arcanum's hypothesis refinement, three data models generated by the LEM2 algorithm were used for hypothesis refinement. The results, as shown in Table 6, used a population size of 500, ran for 50 generations, a 50-50 weighting scheme, and K-fold cross-validation with K value of 10.

TABLE 6

Results of hypothesis refinement on LEM2 data models

Data Set	LEM2 Result	After Refinement
Iris	85.33%	94.79%
Hepatitis	59.56%	71.78%
House-votes	70.8%	96.68%

As shown, Arcanum was able to improve the results of each of the three data models from LEM2 using hypothesis refinement. In each case, the resulting model is better than the one obtained when Arcanum performed data mining on the same data set. After refinement, the data model for Hepatitis is significantly better than the one obtained using just Arcanum. The Iris and House-votes models are both comparable with GAssist and C4.5, with the House-votes model slightly better than C4.5. Although significant improvement was seen in the LEM2 models after hypothesis refinement, the results reflect the success of the original data models obtained from Arcanum. The Arcanum data models for Iris and House-votes were both comparable with GAssist and C4.5, whereas the data model for Hepatitis was inferior. This finding remained true even after hypothesis refinement on the LEM2 models—Iris and House-votes were comparable, whereas Hepatitis, even though remarkably improved, was still inferior to the GAssist and C4.5 models for the same data set.

Another series of experiments were run to test Arcanum's hypothesis refinement ability. A data model was obtained from C4.5 (using J4.8 included in WEKA). The resulting decision tree was then converted into a set of rules that could be used as input to Arcanum. For these experiments, the Breast-c data set was chosen because it was a set upon which Arcanum had not performed comparably with GAssist or C4.5, because plenty of room existed to potentially improve the C4.5 model above the 75.5% accuracy it had achieved, and because the C4.5 model was already better than the GAssist model.

As Table 7 shows, trying to improve upon the C4.5 model was difficult, but improvement was finally achieved by doubling the population size used by the Arcanum genetic algorithm. The resulting model made a slight sacrifice in coverage in order to improve overall accuracy, producing a slightly better model than the original. Thus, Arcanum was able to use hypothesis refinement to improve a C4.5 data model; and it was able to do so using a data set upon which Arcanum did not perform as well as C4.5.

TABLE 7

Results of hypothesis refinement on C4.5 Breast-c data model

Weighting Scheme	Result
30/70	66.6%
50/50	63.5%
70/30	70.6%
80/20	62.8%
50/50 (population 1,000)	71.1%
70/30 (population 1,000)	76.1%

Each set of experiments, using data models produced from three different techniques, showed that Arcanum is able to perform hypothesis refinement to improve upon previously generated data models. This result indicates that the hypothesis refinement capability of Arcanum is a useful tool for data mining; as using Arcanum, previously constructed data models can be improved. Even in the cases for which Arcanum fails to improve upon a data model, the worst-case situation is that the original model is used; yet, the potential for improving the model using hypothesis refinement makes attempting the process a worthwhile task.

6. CONCLUSION

The contributions of this research are two-fold: the development of a multi-tiered genetic algorithm (GA) technique and its ability to perform not only data mining but also hypothesis refinement. The multi-tiered GA is not only a closer approximation to genetics in the natural world but also a way of combining the two competing schools of thought for GAs in data mining.

This paper has shown how the Pittsburg and Michigan approaches to using GAs for data mining can be combined using a multi-tiered approach. This technique was implemented in a project called Arcanum. Testing performed with Arcanum showed that the technique can be a successful data mining tool. More important, testing of the hypothesis refinement capability of this approach showed that Arcanum could take a data model generated by some other technique and improve upon that data model. Once a data model has been achieved, performing hypothesis refinement upon it using the Arcanum technique can improve the overall performance of the model. This capability shows Arcanum to be a valuable step in the data mining process. In the worst case, Arcanum is unable to improve the previous model. Thus, attempting hypothesis refinement has no drawbacks as the outcome is either the original model or one that is better than before.

6.1 Observations

Numerous observations were made during the course of this research that are worthy of discussion. For example, from empiric observations the optimum weights for precision and recall appear to be 70/30 or 30/70. In nearly every case when multiple models were constructed using different weights, the combination of 70% precision and 30% recall produced the best result. The models obtained using 30% precision and 70% recall were often the second best in terms of the results. A number of experiments were performed to observe what happened when the weights were set to extremes. When the weights were set to 100% precision and 0% recall, the resulting model contained a single rule that was 100% accurate (and described only a small portion of the data). When the weights were switched, the resulting model contained a handful of rules that were so generic that each described a large portion of the data, but did so erroneously. Apparently, the algorithm is constrained between 70/30 and 30/70.

During one experiment, we observed an interesting phenomenon in the performance, as shown in Figure 1. The black lines show the average fitness of the population over time. There is a different line for each of the different K-fold runs. The gray lines indicate the fitness of the best specimen encountered. Again, the multiple lines correspond to each of the K-fold runs. It can be seen the something happened during one the K-fold runs. Given that the graph spans 50 generations, some local optima were overcome around generation 25. Whatever happened was of significance and resulted in a significant improvement in average fitness, as well as an improvement in the best solution, which continued to improve at a rapid rate, much faster than it had achieved at any point prior to this discovery.

This phenomenon has a number of possible explanations. It could be that during this particular K-fold run the training data were optimal. Also possible, although unlikely, is that the population received a large number of clones of the best solution, which would explain the sudden upswing in average fitness but does not explain why fitness continued to improve at such a tremendous pace. Another possibility is that the mutation operators in the GA happened to overcome some local optima at that particular point which none of the other runs were able to achieve.

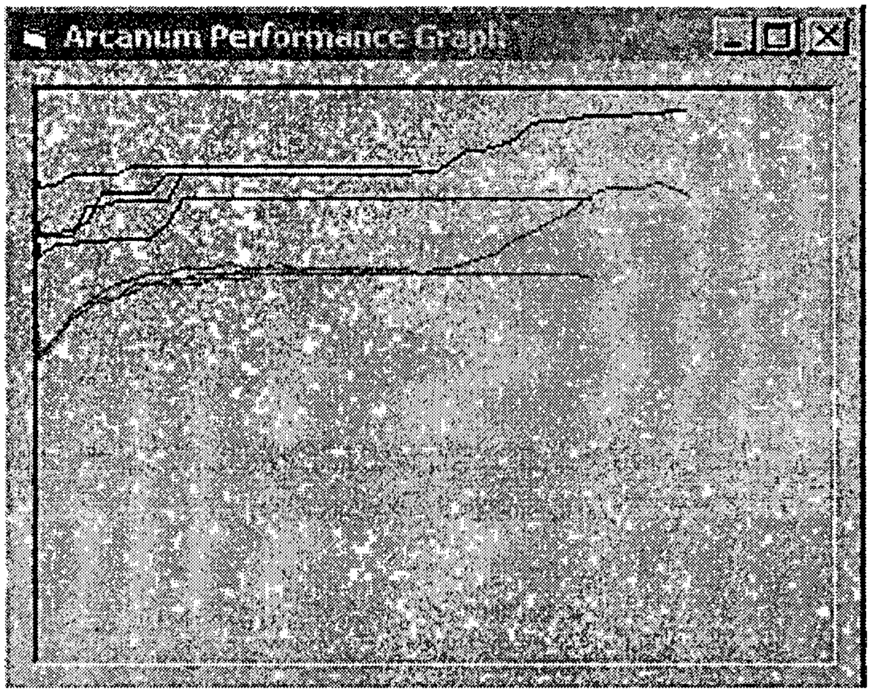

Fig. 1: An Arcanum performance graph.

6.2 Future Work

As discussed in the previous section, some interesting results were observed during experimentation. Two of the possible explanations suggest some future work that should be pursued. The case of the genetic operators perhaps overcoming some local optima suggests that some additional work should be done to explore the possibility of increasing the frequency of mutation based upon the consistency of the average fitness level. In other words, if the average fitness of the population has been consistent over time, then the chances for mutations should increase to bring about increased genetic diversity within the population. This approach would help ensure that the population does not become genetically stagnant with a shallow gene pool from which to create new generations, and thus failing to improve upon the best specimen. This outcome could be achieved using a chance of mutation based upon simulated annealing techniques, in which the chance of mutation increases as the average fitness of the population remains relatively consistent.

The second possibility suggested is that the population receives a number of clones of the best solution, resulting in a sudden increase in average fitness. This approach by itself, however, fails to explain why both the average fitness and the fitness of the best solution continued to improve at significant rates. The other piece of the puzzle might be found in Arcanum's hypothesis refinement. If the population were seeded with clones of the best solution, or even slightly modified versions, this procedure would be similar to performing hypothesis refinement, which was shown to be very successful. It is possible that a better way to perform data mining in Arcanum would be through utilizing hypothesis refinement. Whenever a new best solution is found, the method is treated as if it were a hypothesis and the population is seeded with it, which could make use of the proven strength of the Arcanum technique.

The multi-tiered GA technique described in this paper and the experiments performed using it, indicate that GA has significant potential in the realm of data mining and in the pursuit of finding patterns within complex data sets. The multi-tiered approach also has the potential to use GAs to explore data in more than two dimensions, perhaps opening more opportunities for the application of GAs to other complex problems.

REFERENCES

Arumugam, M.S., and Rao, M.V.C. 2005. Novel hybrid approaches for real coded genetic algorithm to compute the optimal control of a single stage hybrid manufacturing systems. *International Journal of Computational Intelligence*, **1**(3), 189-206.

Au, W.-H., Chan, K.C.C. and Yao, X. 2003. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions of Evolutionary Computation*, 7(6), 532-545.

Bacardit, J., and Butz, M.V. 2004. Data mining in learning classifier systems: comparing XCS with GAssist. *Illinois Genetic Algorithms Laboratory*, IlliGAL Report No. 20040XX

Bacardit, J., and Garrell, J.M. 2007. Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach learning classifier system. *Revised Selected Papers of the International Workshop on Learning Classifier Systems 2003-2005*, Lecture Notes in Computer Science, Springer, 59-79.

Chmielewski, M.R., and Grzymala-Busse, J.W. 1996. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, **15**(4), 319-31.

Choenni, S. 2000. Design and implementation of a genetic-based algorithm for data mining. In *Proceedings of the 26th International Conference on Very Large Data Bases*, 33-42.

CiteSeer. 2008. *What is data mining*? http://citeseer.ist.psu.edu/69212.html.

De Raedt, L., Blockeel, H., Dehaspe, L.. and Van Laer, W. 2001. Three companions for data mining in first order logic. In *Relational Data Mining, edited by* Dzeroski, S., and Lavrac, N, Springer-Verlag, 105-39.

Dulay, N. 2008. *Genetic algorithms*. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html.

Flockhart, I.W., and Radcliffe, N.J. 1996. A genetic algorithm-based approach to data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon.

Goldberg, D. 1989. *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley.

Hamilton, H.J. 2008. *Confusion matrix*. http://www2.cs.uregina.ca/~hamilton/courses/831/notes/confusion_matrix/confusion_matrix.html.

Hettich, S., and Bay, S.D. 1999. The *UCI KDD archive*. Department of Information and Computer Science, University of California at Irvine, http://kdd.ics.uci.edu.

Jech, T. 2002. Set Theory. In *The Stanford encyclopedia of philosophy*, edited by Zalta, E.N. http://plato.stanford.edu/archives/fall2002/entries/set-theory/.

Kohavi, R., and Provost, F. 1998. Glossary of terms, *machine learning*, special issue on applications of machine learning and the knowledge discovery process,**30**(2-3).

Mendez, J., Falcon, A., and Lorenzo, J. 2003. A procedure for biological sensitive pattern matching in protein sequences, In: *Proceedings of the First Iberian Conference on Pattern Recognition and Image Analysis*, Mallorca, Spain, 547-55.

Pawlak, Z. 1994. Rough sets present state and further prospects. In: *Proceedings of the Third International Workshop on Rough Set and Soft Computing*, San Jose, California, 72-76.

Quinlan, R. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Diego, California.

Radivojevic, Z., Cvetanovic, M., and Milutinovic, V. 2003. Data mining: a brief overview and recent IPSI research, *Annals of Mathematics, Computing, and Teleinformatics*, **1**(1), 84-91.

Ratanamahatana, C. 2008. *CloNI: Clustering of √N-Interval Discretization*. http:// www.cs.ucr.edu/~ratana/CloNI.pdf.

Ridgeway, G., and Madigan, D. 2002. A sequential Monte Carlo method for Bayesian analysis of massive datasets. *Journal of Knowledge Discovery and Data Mining*, **7**, 301-19.

Rogers, B. 2008. *Decision tables examples: medical insurance*. http://faculty.sxu. edu/~rogers/sys/decision_tables.html.

Shah, S.C., and Kusiak, A. 2004. Data mining and genetic algorithm based gene/SNP selection. *Artificial Intelligence in Medicine*, **31**, 183-96.

Siler, W. 2008. *Rule-based reasoning: antecedent and consequent*. http:// members.aol.com/wsiler/chap03.htm.

Souza, J., Matwin, S., and Japkowicz, N. 2002. Evaluating data mining models: a pattern language. In: *Proceedings of the Ninth Conference on Pattern Language of Programs*, Urbana, Illinois.

Straccia, U. 1998. A fuzzy description logic. In: *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, Wisconsin, 594-9.

Taylor, CM. 2009. *A multi-tiered genetic algorithm for data mining and hypothesis refinement*. Ph.D. Dissertation, Electrical Engineering and Computer Science Department, University of Kansas.

Taylor, C.M., and Agah, A. 2006. Evolving neural network topologies for object recognition. In *Proceedings of the Sixth International Symposium on Soft Computing for Industry*, World Automation Congress, Budapest, Hungary.

Two Crows Corporation. 2008. *Data Mining Glossary*. http://www.twocrows. com/glossary.htm.

Wikipedia. 2008. *Rough set*. http://en.wikipedia.org/w/index.php?title=Rough_=set &oldid=37222395.

Wikipedia. 2008. *Fuzzy set*. http://en.wikipedia.org/w/index.php?title=Fuzzy_set &oldid=42908464.

Wikipedia. 2008. *Decision table*. http://en.wikipedia.org/w/index.php?title= Decision_table&oldid=45687584.

Williams, G. 2008. *Data mining desktop survival guide*. http://www.togaware.com/ datamining/survivor/Cross_Validation.html.

Witten, I.H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco.

Yin, L., Huang, C.-H. and Rajasekaran, S. 2004. Parallel data mining of Bayesian networks from gene expression data. In *Proceedings of the Eight International Conference on Research in Computational Molecular Biology*, 122-3.