2005

Fuzzy Membership Function Initial Values: Comparing Initialization Methods That Expedite Convergence

Stephanie Scheibe Lee
Virginia Commonwealth University

School of Engineering
Virginia Commonwealth University

This is to certify that the thesis prepared by Stephanie Scheibe Lee entitled Fuzzy Membership Function Initial Values: Comparing Initialization Methods that Expedite Convergence has been approved by her committee as satisfactory completion of the thesis or dissertation requirement for the degree of Masters of Science in Computer Science.

Dr. Lorraine M. Parker, Associate Professor of Computer Science, School of Engineering

Dr. Branson W. Murrill, Associate Professor of Computer Science, School of Engineering

Dr. Amita G. Chin, Associate Professor of Business, School of Business

Dr. David Primeaux, Interim Chairman, Department of Computer Science

Dr. Robert J. Mattauch, Dean, School of Engineering

Dr. F. Douglas Boudinot, Dean of the School of Graduate Studies

June 16, 2005

FUZZY MEMBERSHIP FUNCTION INITIAL VALUES: COMPARING INITIALIZATION METHODS THAT EXPEDITE CONVERGENCE

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science at Virginia Commonwealth University.

by

STEPHANIE SCHEIBE LEE
Bachelors of Liberal Studies in Computer Science, Mary Washington College, 2001

Director: DR. LORRAINE M. PARKER
ASSOCIATE PROFESSOR OF COMPUTER SCIENCE

Virginia Commonwealth University
Richmond, Virginia
June 2005

Acknowledgement

I would like to thank Dr Lorraine M. Parker for her assistance, without which this thesis would not have been completed.

I would like to thank all the faculty and staff of the Department of Computer Science at Virginia Commonwealth University for everything I have learned here.

Table of Contents

List of Tables

Table Page

List of Figures

Figure Page

Abstract

FUZZY MEMBERSHIP FUNCTION INITIAL VALUES: COMPAIRING INITIALIZATION METHODS THAT EXPEDITE CONVERGENCE

By Stephanie Scheibe Lee, Bachelors of Liberal Studies in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2005

Director: Dr. Lorraine M. Parker
Associate Professor of Computer Science

Fuzzy attributes are used to quantify imprecise data that model real world objects. To effectively use fuzzy attributes, a fuzzy membership function must be defined to provide the boundaries for the fuzzy data. The initialization of these membership function values should allow the data to converge to a stable membership value in the shortest time possible. The paper compares three initialization methods, Random, Midpoint and Random Proportional, to determine which method optimizes convergence. The comparison experiments suggest the use of the Random Proportional method.

CHAPTER 1 INTRODUCTION

1.1 Fuzzy Data

Traditional data typically used in relational database systems is precise, crisp data. An example of precise or crisp data would be a person's age. The value assigned to a person's age has no ambiguity. If a person has lived a total of sixty-five years, their age is precisely 65 years old. There is no other possible value for that person's age.

The Relational Model does not accurately capture the imprecision encountered when modeling human opinion as it relates to real world objects [6]. Modeling human opinion for real world entities is better accomplished using a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth values between "completely true" and "completely false", called *fuzzy logic* [18]. Using a fuzzy data model helps database developers represent non-precise values that are relative to a real world entity and provide the means to quantify that data. For example, in addition to a person's age, the database can also store a value that quantifies whether or not a person is considered to be Old.

Defining a fuzzy subset Old, will answer the question "to what degree is person of Age (x) Old?" For the people for whom it is desired to quantify the extent to which they are Old, a

degree of membership in the Old fuzzy function based on the person's Age can be assigned.

1.2 Fuzzy Membership Functions

Defining the fuzzy membership function in terms of age allows the assignment of a crisp value that implies a fuzzy meaning. For the example, consider the fuzzy membership function defined in Figure 1 below:

Figure 1. Fuzzy Function Definition

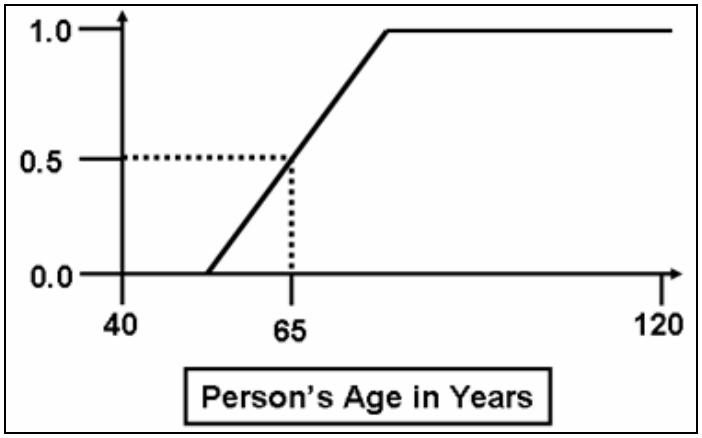
$$\text{Old}(x) = \{\ 0, \text{if } 0 > \text{Age}(x) < 40$$
$$0.010\text{-}00.99, \text{if } 41 \geq \text{Age}(x) < 90$$
$$1, \text{if Age}(x) \geq 90\ \}$$

A person between the ages of 0 to 39 years of age would be assigned a value of 0 in the fuzzy attribute within the database, implying that they are not old. Any person ranging in age from 40 to 89 would be assigned a value within the rage of 0.01 to 0.99. Any person over the age of 90 would be fully included in Old by assigning a value of 1.0. The values contained within the database are crisp values mathematically, however semantically they represent a fuzzy value.

Defining a fuzzy membership function extends the two-valued logic for a membership value into degree of membership in a function within a range [0, 1]. For a person with a

high inclusion value, there is a high confidence that they belong to the class of Old people. Conversely, people with a low inclusion value in the class Old, (say from .5 to 0) shows a low confidence that they are Old. For a person that is 65 years old, their corresponding Old fuzzy value might be .0.5, signifying that there is some degree of oldness as it relates to that specific person.

Figure 2. Determining of a Fuzzy Value



1.3 Hedges

A *hedge* is another term for a fuzzy modifier. Hedges are adjectives applied to fuzzy attributes further narrowing down their meaning. Hedges operate using the same principles of formal logic [8].

Fuzzy logic expresses knowledge about subjective concepts such as Old and Tall, and hedges are used to further extend those subjective definitions. Examples of a hedge could

be Very, Slightly, or Extremely. The application of the hedges is typically directly related to the fuzzy attributes being quantified. There are infinite numbers of hedges that could be applied to any fuzzy attribute [33]. In following the current example, defining hedges could be done as follows:

1. Slightly Old – 0.0 to 0.4,
2. Old – 0.41 to .075, and
3. Very Old - .76 to 1.0

To locate all the "Very Old" people in a database the fuzzy attribute is queried requesting all the Old values greater than 0.75. The results would be a list of people who are all older than 68 years of age.

1.4 Inference

Inference is the application of the fuzzy membership function and a set of applied rules on an input value to approximate a resulting fuzzy output value [16]. Inference methods examine the underlying meaning of the data [32]. This application of reasoning to semantically uncertain data provides insight into that data's meaning [15]. Inference uses the same logical models as fuzzy logic (i.e. union, intersection, etc) [5].

For example, a possible inference rule might state that if a person is very old AND very tall, that person is of the male gender. In querying the database to retrieve the set of very old and very tall people, all the people returned should be male.

1.5 Fuzzy Logic in Commercial Applications

Fuzzy Logic has been applied, within commercial applications, to the fields of Artificial Intelligence and Machine Learning [18][14]. Machine Learning in commercial applications has been implemented using fuzzy logic controllers for appliances such as Washing Machines, Thermostats and Shower Heads. The fuzzy logic usage within the washing machine determines spin cycles and clothing agitation based on estimated washer load [1]. The thermostat and showerhead use fuzzy logic to alter temperature, ultimately conserving energy for the consumer [30]. Applications have been developed in other commercial sectors such as Finance, Pattern Recognition and Data Analysis [18]. Artificial Intelligence applications using fuzzy logic have emerged in systems such as Traffic Controllers and Air Flight Predictors. Traffic controllers sense traffic and divert it using fuzzy logic algorithms [23]. Air Flight Predictors use fuzzy logic to predict future the location of an aircraft based on current flight paths and speed [13].

Research in the field of fuzzy logic been implemented in mainstream commercial software products such as Matlab and rFLASH. Matlab provides a companion fuzzy logic toolkit

[25]. Rigel Corporation also distributes rFLASH, an assembly language code generator to implement fuzzy logic routines [28].

Fuzzy Logic expresses knowledge about subjective concepts such as Old and Tall, which can be converted into crisp numeric ranges. Using membership functions to quantify non-crisp values provides a method to capture a range of membership; however, the initialization of these values is a tricky matter. The optimization of fuzzy data requires that the object membership function values be representative of the object's degree of membership within the defined function. The objective of determining the best initial membership function values is to optimize the normalization or convergence of the data to the best value for that object within the range of the function.

Machine Learning suggests the random initialization of the membership or learning function data. Choosing other implementation methods has provided no increased learning ability since learning is accomplished by executing different algorithms randomly to learn or retrieve the data within the system [26]. If there was a distinct pattern to the learning process, the order or method of initialization could aide the learning process. The current system allows user's to query a database. User's then provide feedback on the results determining if they met their subjective criteria. The prototype provides no method for automating the querying process across varying ranges. A text file installed with the application loads the query text into the query window. If the user has knowledge of the system they may alter the query text before submission. Since the user has the ability to

alter the query text, the querying process can appear random based on the user's preference.

CHAPTER 2 EXPERIMENT OVERVIEW

2.1 Project Overview

Previous research within the Virginia Commonwealth University, School of Engineering's Department of Computer Science's Database Research Group has created a prototype Fuzzy Database with a Natural Language Querying System with Community Defined Membership Values [17] [4]. A fuzzy database was implemented to allow a user to query and retrieve facial images based on two independent factors: 1) Eye Color, and 2) Shape of the Face. This prototype database provides a community of users the ability to provide feedback on the query results, ultimately altering the fuzzy membership function values until convergence within the community is reached.

The ultimate goal of the prototype was to seek an implementation that would provide a basis for further research. Several issues were simplified when developing the prototype allowing expansion of the research in the following areas:

1. Compound Queries of Fuzzy Attributes,
2. Fully Integrating a Natural Language Querying System,
3. Initialization Methods of Fuzzy Attributes, and
4. Prototype Implementation with Spatial Image Data.

For learning systems, developing the fuzzy data provides a particular problem of interest. The prototype used subjective evaluation and elicitation to adapt the fuzzy data until

convergence was revealed. Convergence for a system can be measured by several methods:

1) Quality of the learned data,
2) Delta time to learn the data, and
3) Data Stability

The above examples are different types of convergence measures. Convergence can be measured on alternate factors depending on the type of the system in which convergence is being achieved [26]. For the purpose of this paper, convergence is based on the total number of queries needed to provide subjective data stability. Quality of the resulting data with respect to a user community has neither been implied nor tested.

There are several alternative methods to mature fuzzy data that do not require convergence to the community defined consensus [18]:

1. Ad-hoc forms – Choosing a central value for all data points.
2. Converted frequencies or probabilities – Value defined in a membership function from data in similar states.
3. Physical Measurement – Grades of membership data are calculated from a defined membership function.

These methods are appropriate for systems where calculated fuzzy data is stable over time. The prototype was designed to allow users to query the database using everyday language, returning those images that met the user's subjective criteria. The community would

describe the images. That is, the ultimate membership function value for each image is determined based on the community consensus or convergence. Since each user was expected to potentially describe the same images differently, the fuzzy attribute membership value is altered each time a user provides feedback. At some point, it is expected that the user community opinions will stabilize for each image's values and convergence will be reached. The community convergence can change over time as the population of the community is altered [17]. The initialization of the fuzzy data membership values presents a particular problem with a learning system. Selecting appropriate initial values that do not hinder community convergence of the data is important to expedite convergence [26].

The prototype initialized the fuzzy membership function values randomly. No investigation was performed with respect to different methods of initializing the fuzzy membership functions since it was a prototype and further research was expected. As an extension of the original research, this paper compares various initialization methods of the fuzzy membership function values that may expedite convergence of the fuzzy values [2] [17].

The prototype was modified for this project. The prototype only returned six images based on the user's query to the Graphical User Interface, regardless of the number of images that met that query criteria. The modified version requires the user to view and evaluate all the images that meet the user's query criteria in the Graphical User Interface. Additionally, the weighting factors applied to modify fuzzy attribute values were changed from .01 to

.02, since actual community convergence was not used for the experiment portion of this research. The original prototype selected a small weighting factor to protect the data from exclusion if several successive queries provided the same conclusions about a single image. For example, if the first 25 people indicated that an image met the criteria of the fuzzy attribute being evaluated the attributes values would be incremented 25 times. If the next 50 people indicated the images did not meet the criteria of the fuzzy attributed being evaluated, 50 decrements would occur. If the interval of increment/decrement is small enough, more users are required to push an image below the threshold value eliminating it from further evaluation. Single user convergence was used for this experiment; therefore the weighting factors did not require the protection of the data from exclusion. The complete source code is included in Appendix O since it is a modification of the code contained in [17].

2.2 Experiment Overview

Good initializations of the fuzzy function membership values can expedite the normalization, or convergence of the membership value for each image object contained within the database [26]. Once convergence has occurred for an object, the membership values within the function will stabilize. Additional queries by a stable user community provide no additional insight and hence no need to change the value. It is necessary to further investigate the longevity of convergence based on stability of the user community. The aspect of convergence as it pertains to stability or instability of a community of users is not addressed within this paper; however it should be addressed in future work.

The prototype database has two tables that use fuzzy membership values: Color and Face. For each table, there are three fuzzy attributes values. For the Color table the attribute Eye Color has values: 1) Blue, 2) Green and 3) Brown. For the Face table the width attribute has are 1) Average, 2) Broad and 3) Wide. The prototype database also contains a table of hedges, called Ranges, that stores the numerical hedge values applied to the membership function for each fuzzy value. The range value quantifies the degree of membership as described in Table 1. As users provide feedback on an image, the image's resulting value is moved closer to the Threshold value for the particular hedges range.

Table 1. Fuzzy Modifier Range Values

Modifier Name	High Value	Low Value	Threshold Value
Very (V)	1.00	0.75	0.87
Medium (M)	0.74	0.57	0.30
Slightly (S)	0.29	0.00	0.20

The "Slightly" (S) low range modifier was limited to 0.10 in the original prototype to allow images to fall outside the range of applicable values. After repeated querying of the system if an image attribute value is sufficiently downgraded, the image attribute value would eventually fall out of the viewable range. At that point, the user community could no longer provide feedback on the images applicability. The lower limit of the "Slightly"

allows for the exclusion of a person's image when the fuzzy membership function does not apply. Exclusion of an image attribute value within a function over time was also not researched with the original prototype. This experiment altered the original prototypes low value for the "Slightly" range to ensure all images were evaluated with respect to their fuzzy attribute value within the database. The original prototype range values contained a buffer between 0.0 and 0.1 which allowed values to fall outside the viewable range. If the range was not altered, an image's attribute value could have been initialized to a value outside the viewable range, excluding that image's attribute value from evaluation. The second option to fix this would have been to alter the all initialization functions assignment of values fall within the viewable range 0.11 and 1.0. The exclusion of an image within a fuzzy function by lowering its membership value is not addressed within this paper; however it should be addressed in future research.

The point of this experiment is to test different fuzzy membership value initialization methods. Since the fuzzy functions are all applied to one or more possible descriptors of a single object (i.e. Eye color, where the possible choices are Blue, Green and Brown) a proportional method of initialization seemed logical. If an attribute has high membership value in one function then that same object should not have a high membership value in another function. For instance, if the value assigned for Blue is 0.95, then it would make no sense for the same person to have a value of 0.95 assigned for the Brown or Green attributes. Conversely, the Brown or Green attributes should be 0.02, the proportional

opposite of the Blue attribute values. The three attribute values should sum to 1.0, however some rounding occurs.

There are distinctions between a fuzzy membership function and a proportional value with respect to semantic and mathematical levels of the theory. Semantically, fuzzy logic and probability theory use different notions: Probability and Degree of Membership. Probabilities are the likelihoods that an event does or does not occur. Fuzzy Logic models the extent to which an event occurred or can occur [3]. Even though fuzzy membership values and proportional values operate over the same functional range [0, 1], the semantic difference between a fuzzy statement and probabilistic statement is significant [8]. Proportional values have the minimum requirement of Additivity [18]. The combined value across the attributes must sum to the value 1.0. Additivity does not apply to fuzzy values. Even though probabilities and fuzzy membership function are semantically different, fuzzy membership values can be determined using proportional theory [30]. A random proportional method was chosen for the first initialization type for this experiment.

Another method would be the initialization of all the values to the midpoint value of the function range, in this case 0.50. Choosing the midpoint of the range simplifies the problem by choosing a central value within the range [18]. Convergence to the high and low level range modifiers should have a smaller delta change to convergence. This method would be applicable if the community convergence polling could be automated so the system could adapt by providing queries on ranges based on the number of attributes containing values in that function range [26]. Otherwise, repeated polling in the Medium

(M) range would limit the user community's ability to provide an adequate number of response feedbacks on the attribute values that are cycled out of the Medium (M) range.

Machine learning algorithms for initialization of fuzzy membership suggest random initialization as the best method to produce optimal convergence [26]. When humans attempt to determine these values, their instincts are usually incorrect and delay convergence of the data [26]. [26] examples are specific to machine learning techniques as they relate to defining the initial value of the learning function, which is similar to the community-learning prototype that is being used for this experiment. The original prototype used random initialization of the fuzzy membership function.

2.3 Methodology

The first part of this experiment was to initialize the fuzzy membership values for the pictures within the database using three methods:

1. Independently Random- Generated randomly between the ranges of 1.0 and 0.0.
2. Random Proportional – First value generated randomly between the ranges of 1.0 and 0.0. Remaining two values split the proportional difference. Setting the values to the proportional center (.33) was essentially the same as setting the values to the midpoint of the range in the proportional sense. Adding the initial random value did not guarantee a distribution across all ranges.
3. Fixed- Fixed to the midpoint of the function value, in this case 0.5.

For the purpose of this experiment, convergence is based on one user's opinion, simulating a user community. Since the target user community contains an unknown number of users each having varied opinions, convergence based on a wider community of user's should be evaluated in future work. Additionally, since the results are based on a limited community view, verification with a target community should also be done.

These methods were chosen based on the current research in the field [18]. Each initialization method required the creation of a stored procedure to initialize the membership function. The stored procedure for randomly assigning the initialization of the fuzzy membership function values is contained in Appendix J. The stored procedure contained within Appendix J is the same stored procedure for the original prototype [2] [17]. Two additional stored procedures were created to initialize the fuzzy membership function values for the remaining two initialization methods selected. The stored procedure for the random then proportional initialization of the fuzzy membership function values is contained in Appendix K. The stored procedure for the fixed initialization of the fuzzy membership function values to the midpoint of the range is contained in Appendix L.

The second part of the experiment was to query the database and provide feedback on the query responses. The querying continued until all the query responses provided "correct results" or the data converged with respect to the range noted in Table 1. The term "correct results" is an interpretation of the Color and Range variables. For instance, consider that there are two images contained within the database where the eye color is

“Very Green”. The querying pattern was then repeated for each color/range value pairing and initialization method until the values are essentially same. In other words, the query was run on “Very Brown”, “Medium Brown” and “Slightly Brown” Eyes independently until only the query results returned by the system meet the criteria for each range and color in question and so on for each color and range variable.

The final part of the experiment was to compare the total number of queries until convergence and recommend the optimal fuzzy membership initialization function that would optimize convergence independent of the data type stored in the fuzzy membership function.

The experiment was initially conducted using only the Color table. Because the Color and Face table both contained three fuzzy membership functions values that pertained to one facial object, there was no benefit to performing a duplicate series of queries toward convergence.

CHAPTER 3 RANDOM INITIALIZATION OF MEMBERSHIP FUNCTION VALUES

3.1 Setting the Initial Membership Function Values

All the attribute values for the Color table were initialized using the `initialize_weights` stored procedure in the SQL Server database as seen in Appendix J. This procedure randomly set all the weights within the table to the values noted in Appendix A, Tables 33-37.

3.2 Querying the Database until Convergence

Once membership values were initialized, the database was repeatedly queried for each range value of each eye color. The range quantifiers were queried in the following order:

1. Medium
2. Slightly
3. Very

For Example, the query was repeated until all the responses delivered by the database met the criteria of having medium brown eyes. This querying process was also completed for each Range/Color pairing. The querying order across the range quantifiers was patterned to provide a query response for the Midpoint portion of the experiment. All portions of the

experiment followed the same querying pattern. During the querying, the following data points were tracked:

1. Number of times the query was performed for each membership function,
2. The total number of queries for each modifier, and
3. The color responses per query (i.e., 3 Very Brown, 21 Medium Brown, and 7 Slightly Brown) after convergence.

The post convergence values contain within the Color table after this portion of the experiment was completed are contained within Appendix B, Tables 38-43.

3.3 Results from the Random Initialization

For each attribute in the Color table, three evolutions of querying were completed (one for each color, Brown, Blue and Green; this is referred to as a round. During each round, the queries covered each Range value for each color, Medium, Slightly then Very; this is referred to as a cycle. After a cycle of queries was performed, subsequent cycles were repeated until the Range/Color value combinations converged. During this process, the total number of queries was calculated for each Range/Color pairing; these values are provided under the range/Color pairing heading in the Total # of Queries row in Tables 2-7 below. The number of images is consistent across each round for each Range/Color pairing; these values are provided in each table under the Range/Color heading within the Final # of Images row. Each table also provides the total number of images evaluated and the total number of queries per round in their respective row under the Total heading.

The fuzzy values in the Color table were initialized using the `initialize_weights` stored procedure. For each execution of the stored procedure, three rounds of querying were performed.

Table 2. Random Initialization Query Results (Round 1)

Random	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	62	37	14	11
Final # of Images	31	9	17	5

Table 3. Random Initialization Query Results (Round 2)

Random	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	66	39	18	9
Final # of Images	31	14	16	1

Table 4. Random Initialization Query Results (Round 3)

Random	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	66	44	12	15
Final # of Images	31	4	25	2

It was determined to repeat the process one more time, giving the results in tables 5-7. At that point, the Total number of queries to convergence remained consistent therefore no further evolutions were performed.

Table 5. Random Initialization Query Results (Round 4)

Random	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	57	32	14	11
Final # of Images	31	9	17	5

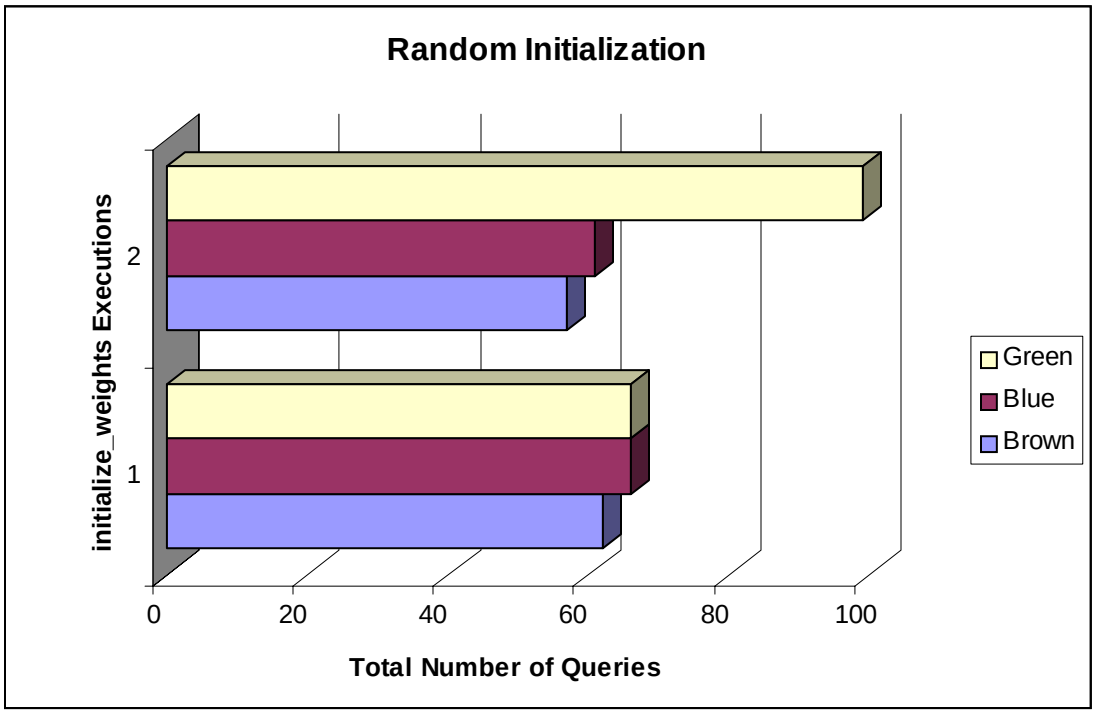
Table 6. Random Initialization Query Results (Round 5)

Random	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	61	37	15	9
Final # of Images	31	14	16	1

Table 7. Random Initialization Query Results (Round 6)

Random	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	99	47	37	15
Final # of Images	31	4	25	2

Figure 3. Random Initialization Query Results



The average number of queries for the Random Initialization Method was 70 queries per attribute. Each attribute's value within the Color table was generated randomly, their values are not dependent by any means, and therefore a high value for any single Eye Color's attribute values, such as the Green attribute in round 2, could be expected

The attributes values for the Eye Color Green, only six images values were to be converged to the "Very" and "Medium" ranges. For the second round of initializations, a 24 of attributes values were initialized to those ranges, requiring

additional queries to push the attribute values into the correct range for convergence. This behavior is typical of randomly assigned values.

CHAPTER 4 RANDOM PROPORTIONAL INITIALIZATION OF MEMBERSHIP FUNCTION VALUES

4.1 Setting the Initial Membership Function Values

During this portion of the experiment, the membership function value was generated randomly for Brown within the valid range [1, 0]. The other two membership function values, Blue and Green, were set proportionately from the first value using the following formula:

$$V_o = \text{Random } [1, 0];$$

$$V_1, V_2 = (1.0 - (V_o)) \ / \ 2.0;$$

For example, the membership value Brown can be generated randomly to the value listed in Table 3 below. The resulting values for Blue and Green are $V_1, V_2 = (1.0 - 0.54) / 2.0$, as seen in the example below.

Table 8. Initializing Random Proportional Values Example

Color	Value
Brown (V_o)	0.54
Blue(V_1)	0.23
Green (V_2)	0.23

The aforementioned example is only one possible value, since the initial value of V_o is generated randomly within the function range [1, 0]; each set of values is potentially different. Values were not limited to mathematically even values. For instance, if the initial random value generated was .43, the resulting generated values would be $V_{1,}$ V_2 = (1.0 – 0.43) / 2.0, or 0.285. Values within the database were limited to two decimal places therefore the initialization values would appear as 0.28 in database.

All the attribute values for the Color table were initialized using the `initialize_other` stored procedure in the SQL Server database as seen in Appendix K. All the values were set randomly proportionally as described in the example in Table 3 above. The values assigned to the table values are noted in Appendix C, Tables 44-49.

The pattern of the initialization was Brown, being set randomly, then Blue and Green being set proportionally from the initial value of Brown according to the formula above. After the initial portion of the experiment was performed, a concern was raised about a possible bias toward Brown. The bias could prevent the Blue and Green color values from distribution across all the possible ranges. This concern was evaluated by re-writing the Random Proportional initialization method to rotate between each color value as first for initialization. That portion of the experiment is detailed in Chapter 6.

4.2 Querying the Database until Convergence

Once membership values were initialized, the database was repeatedly queried for each range value of each eye color. The range quantifiers were queried in the following order:

1. Medium
2. Slightly
3. Very

For Example, the query was repeated until all the responses delivered by the database met the criteria of having very brown eyes. This querying process was also completed for each range/color pairing. The querying order across the range quantifiers was patterned to provide a query response for the Midpoint portion of the experiment. All portions of the experiment followed the same querying pattern. During the querying, the following data points were tracked:

1. Number of times the query was performed for each membership function,
2. The total number of queries for each modifier, and
3. The color responses per query (i.e., 3 Very Brown, 21 Medium Brown, and 7 Slightly Brown) after convergence.

The post convergence values contain within the Color table after this portion of the experiment was completed are contained within Appendix D, Tables 50-55.

4.3 Results from the Random Proportional Initialization

For each attribute in the Color table, three evolutions of querying were completed (one for each color, Brown, Blue and Green; this is referred to as a round. During each round, the queries were covered each Range value for each color, Medium, Slightly then Very; this is referred to as a cycle. After a cycle of queries was performed, subsequent cycles were repeated until the Range/Color value combinations converged. During this process, the total number of queries was calculated for each Range/Color pairing; these values are provided under the range/Color pairing heading in the Total # of Queries row in Tables 9-14 below. The number of images is consistent across each round for each Range/Color pairing; these values are provided in each table under the Range/Color heading within the Final # of Images row. Each table also provides the total number of images evaluated and the total number of queries per round in their respective row under the Total heading.

The fuzzy values in the Color table were initialized using the `initialize_other` stored procedure. For each execution of the stored procedure, three rounds of querying were performed.

Table 9. Random Proportional Initialization Query Results (Round 1)

Random Proportional	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	36	20	14	2
Final # of Images	31	9	17	5

Table 10. Random Proportional Initialization Query Results (Round 2)

Random Proportional	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	72	43	14	15
Final # of Images	31	14	16	1

Table 11. Random Proportional Initialization Query Results (Round 3)

Random Proportional	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	41	27	13	1
Final # of Images	31	4	25	2

It was determined to repeat the process one more time, giving the results in tables 12-15. At that point, the Total number of queries to convergence remained consistent therefore no further evolutions were performed.

Table 12. Random Proportional Initialization Query Results (Round 4)

Random Proportional	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	37	21	14	2
Final # of Images	31	9	17	5

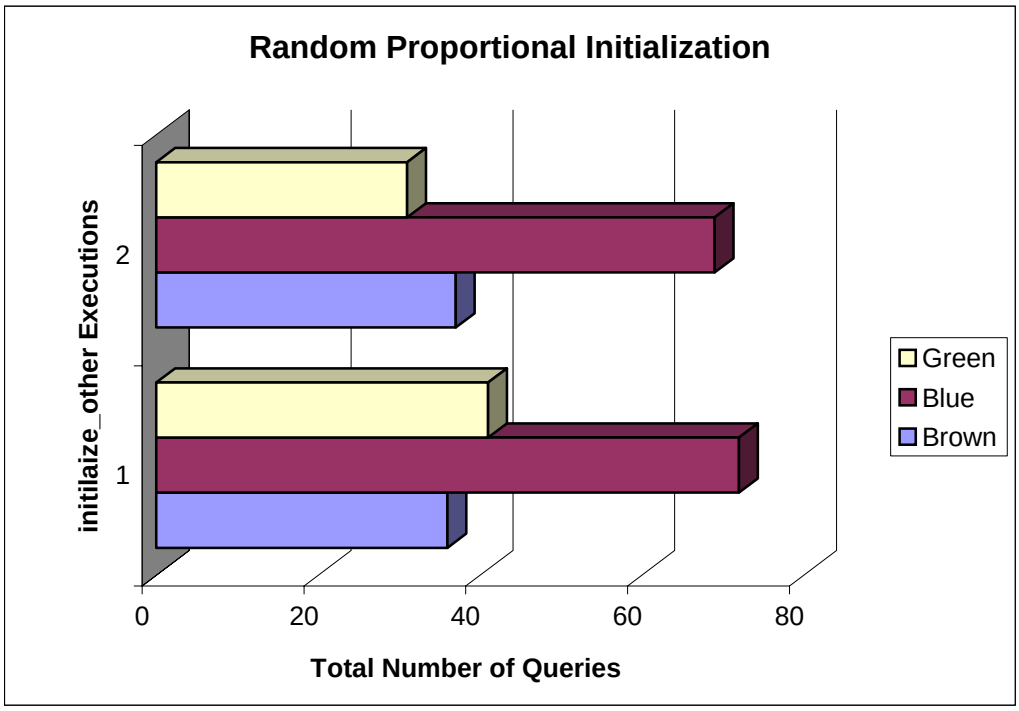
Table 13. Random Proportional Initialization Query Results (Round 5)

Random Proportional	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	69	44	12	13
Final # of Images	31	14	16	1

Table 14. Random Proportional Initialization Query Results (Round 6)

Random Proportional	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	31	29	1	1
Final # of Images	31	4	25	2

Figure 4. Random Proportional Initialization Query Results



The average number of queries for the Random Proportional Initialization Method was 47 queries per attribute. Each attribute's value within the Color table was generated randomly, their values are dependent since the first attribute is random and the other two resulting attributes are the proportional difference as stated in Section 4.1.

CHAPTER 5 FIXED INITIALIZATION OF MEMBERSHIP FUNCTION VALUES TO THE MIDPOINT OF THE FUNCTION RANGE

5.1 Setting the Initial Membership Function Values

All the attribute values in the Color table were initialized using the `initialize_mids` stored procedure in the SQL Server database as noted in Appendix L. All the values were set to the midpoint of the range function, or 0.50. The initial values assigned to the table values are noted in Appendix E, tables 56-58.

5.2 Querying the Database until Convergence

Once membership values were initialized, the database was repeatedly queried for each range value of each eye color. The range quantifiers were queried in the following order:

1. Medium
2. Slightly
3. Very

For Example, the query was repeated until all the responses delivered by the database met the criteria of having very brown eyes. This querying process was also completed for each range/color pairing. During the querying, the following data points were tracked:

1. Number of times the query was performed for each membership function,
2. The total number of queries for each modifier, and

3. The color responses per query (i.e., 3 Very Brown, 21 Medium Brown, and 7 Slightly Brown) after convergence.

The post convergence values contain within the Color table after this portion of the experiment was completed are contained within Appendix F, Tables 59-64.

5.3 Results from the Fixed Initialization to the Midpoint of the Function Range

For each attribute in the Color table, three evolutions of querying were completed (one for each color, Brown, Blue and Green; this is referred to as a round. During each round, the queries were covered each Range value for each color, Medium, Slightly then Very; this is referred to as a cycle. After a cycle of queries was performed, subsequent cycles were repeated until the Range/Color value combinations converged. During this process, the total number of queries was calculated for each Range/Color pairing; these values are provided under the range/Color pairing heading in the Total # of Queries row contained in Tables 15-20 below. The number of images is consistent across each round for each Range/Color pairing; these values are provided in each table under the Range/Color heading within the Final # of Images row. Each table also provides the total number of images evaluated and the total number of queries per round in their perspective row under the Total heading.

The fuzzy values in the Color table were initialized using the `initialize_mids` stored procedure. For each execution of the stored procedure, three rounds of querying were performed.

Table 15. Midpoint Initialization Query Results (Round 1)

Midpoint	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	30	26	2	2
Final # of Images	31	9	17	5

Table 16. Midpoint Initialization Query Results (Round 2)

Midpoint	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	33	29	2	2
Final # of Images	31	14	16	1

Table 17. Midpoint Initialization Query Results (Round 3)

Midpoint	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	34	26	6	2
Final # of Images	31	4	25	2

It was determined to repeat the process one more time, giving the results in tables 18-20. At that point, the Total number of queries to convergence remained consistent therefore no further evolutions were performed.

Table 18. Midpoint Initialization Query Results (Round 4)

Midpoint	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	32	28	2	2
Final # of Images	31	9	17	5

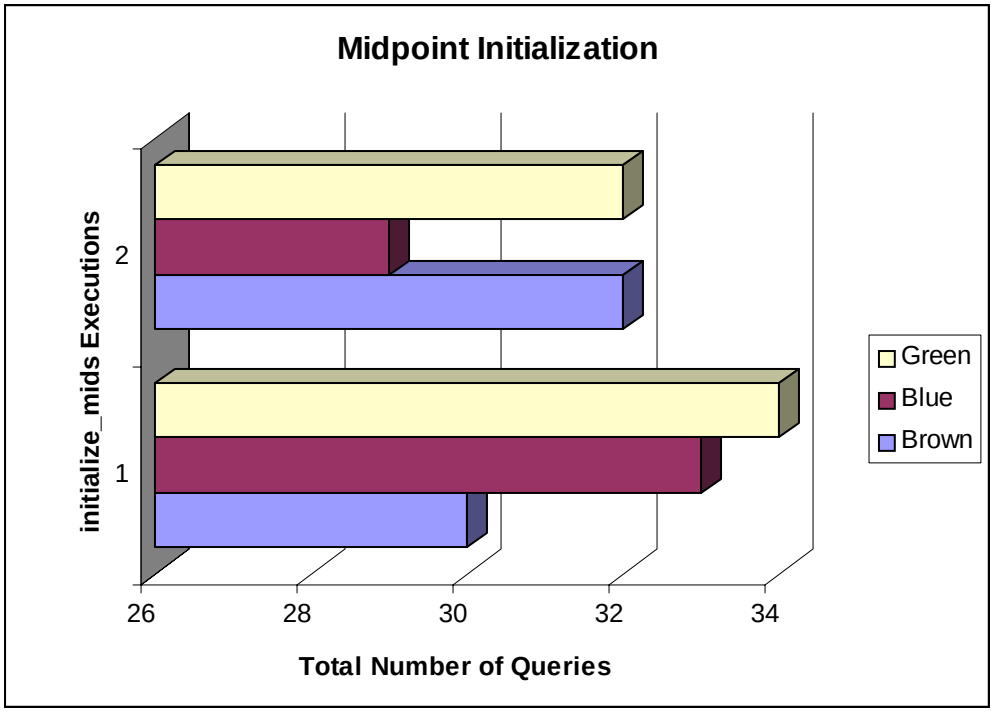
Table 19. Midpoint Initialization Query Results (Round 5)

Midpoint	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	26	22	2	2
Final # of Images	31	14	16	1

Table 20. Midpoint Initialization Query Results (Round 6)

Midpoint	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	32	28	2	2
Final # of Images	31	4	25	2

Figure 5. Midpoint Initialization Query Results



The average number of queries for the Midpoint Initialization Method was 32 queries per attribute. Each attribute's value within the Color table was set to the Midpoint of the function value range, or 0.50. Since the community feedback was simulated with only one user, the total number of queries across the different color attributes was consistent across both rounds. However, each attributes value is not dependent, and therefore a high value for any attribute could be expected for additional rounds of querying based on the specific user's opinion of the images with a target community. A user's opinion on a image may change over times as comparisons to other images affect the user's opinion of the color intensity.

CHAPTER 6 NEW RANDOM PROPORTIONAL INITIALIZATION OF MEMBERSHIP FUNCTION VALUES

6.1 Setting the Initial Membership Function Values

During this portion of the experiment, the membership function value was generated randomly for one color within the valid range [1, 0]. The other two membership function values were set proportionately from the first value using the following formula:

$$V_o = \text{Random } [1, 0];$$

$$V_1, V_2 = (1.0 - (V_o)) / 2.0;$$

For example, the membership value for the color blue can be generated randomly to the value listed in Table 3 below. The values for Brown and Green are defined as $V_1, V_2 = (1.0 - 0.54) / 2.0$, as seen in the example below.

Table 21. Initializing New Random Proportional Values Example

Color	Value
Blue (V_o)	0.54
Brown (V_1)	0.23
Green (V_2)	0.23

The aforementioned example is only one possible value, since the initial value of V_o is generated randomly within the function range [1, 0]; each set of values is potentially

different. Values were not limited to mathematically even values. For instance, if the initial random value generated was 0.43, the resulting generated values would be V_1, V_2 = (1.0 – 0.43) / 2.0, or 0.285. Values within the database were limited to two decimal places therefore the initialization values would appear as 0.28 in database.

The `initialize_other_random` stored procedure still generates the values in the same way as the `initialize_other` stored procedure described in Chapter 4 however; the attribute value that is generated randomly is rotated between Blue, Green and Brown eliminating the bias generated in the original stored procedure.

All the attribute values for the Color table were initialized using the `initialize_other_random` stored procedure in the SQL Server database as seen in Appendix M. All the values were set randomly proportionally as described in the example in Table 21 above. The values assigned to the table values are noted in Appendix G, Tables 65-70.

6.2 Querying the Database until Convergence

Once membership values were initialized, the database was repeatedly queried for each range value of each eye color. The range quantifiers were queried in the following order:

1. Medium
2. Slightly
3. Very

For Example, the query was repeated until all the responses delivered by the database met the criteria of having very brown eyes. This querying process was also completed for each range/color pairing. During the querying, the following data points were tracked:

1. Number of times the query was performed for each membership function,
2. The total number of queries for each modifier, and
3. The color responses per query (i.e., 3 Very Brown, 21 Medium Brown, and 7 Slightly Brown) after convergence.

The post convergence values contain within the Color table after this portion of the experiment was completed are contained within Appendix H, Tables 71-76

6.3 Results from the New Random Proportional Initialization

For each attribute in the Color table, three evolutions of querying were completed (one for each color, Brown, Blue and Green; this is referred to as a round. During each round, the queries were covered each Range value for each color, Medium, Slightly then Very; this is referred to as a cycle. After a cycle of queries was performed, subsequent cycles were repeated until the Range/Color value combinations converged. During this process, the total number of queries was calculated for each Range/Color pairing; these values are provided under the range/Color pairing heading in the Total # of Queries row contained in Tables 22-27. The number of images is consistent across each round for each Range/Color pairing; these values are provided in each table under the Range/Color heading within the

Final # of Images row. Each table also provides the total number of images evaluated and the total number of queries per round in their perspective row under the Total heading.

The fuzzy values in the Color table were initialized using the `initialize_other_random` stored procedure. For each execution of the stored procedure, three rounds of querying were performed.

Table 22. New Proportional Initialization Query Results (Round 1)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	41	28	1	12
Final # of Images	31	9	17	5

Table 23. New Proportional Initialization Query Results (Round 2)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	64	35	13	16
Final # of Images	31	14	16	1

Table 24. New Proportional Initialization Query Results (Round 3)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	23	21	1	1
Final # of Images	31	4	25	2

It was determined to repeat the process one more time, giving the results in tables25-27. At that point, the Total number of queries to convergence remained consistent therefore no further evolutions were performed

Table 25. New Proportional Initialization Query Results (Round 4)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Brown	Slightly Brown	Very Brown
Total # of Queries	55	37	16	2
Final # of Images	31	9	17	5

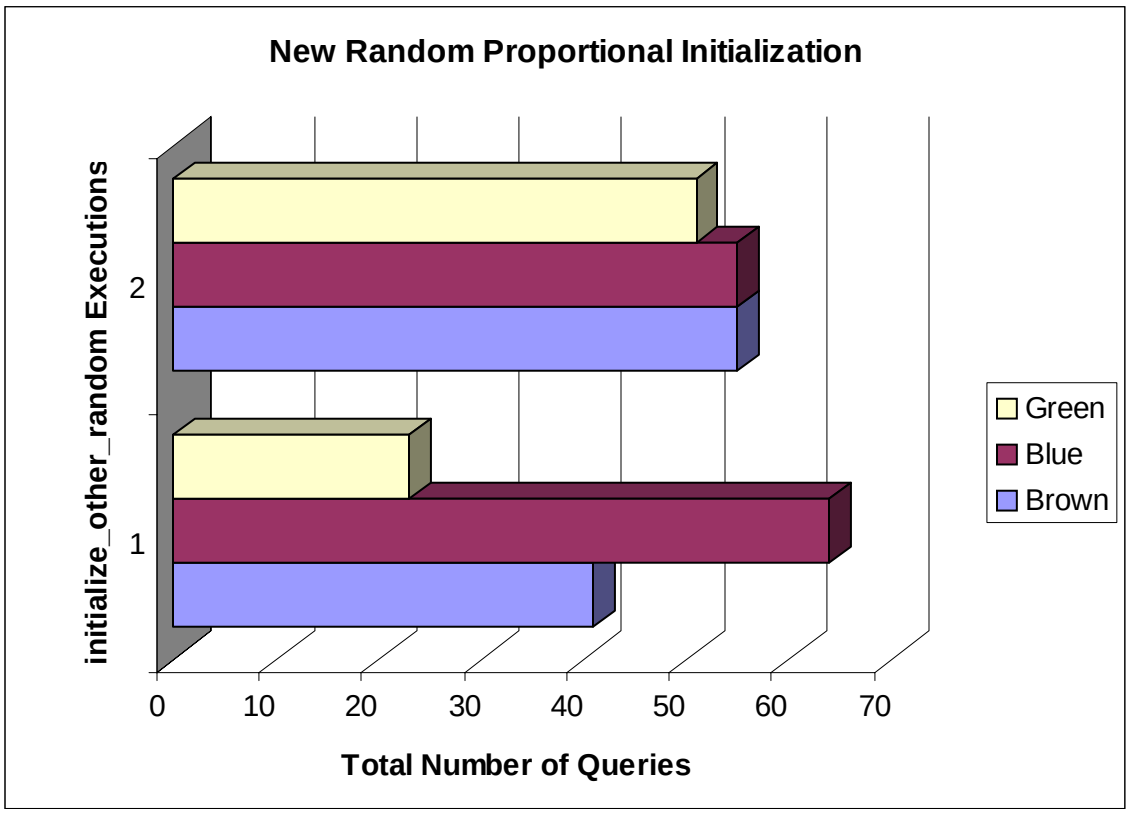
Table 26. New Proportional Initialization Query Results (Round 5)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Blue	Slightly Blue	Very Blue
Total # of Queries	55	29	11	15
Final # of Images	31	14	16	1

Table 27. New Proportional Initialization Query Results (Round 6)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Green	Slightly Green	Very Green
Total # of Queries	51	40	9	2
Final # of Images	31	4	25	2

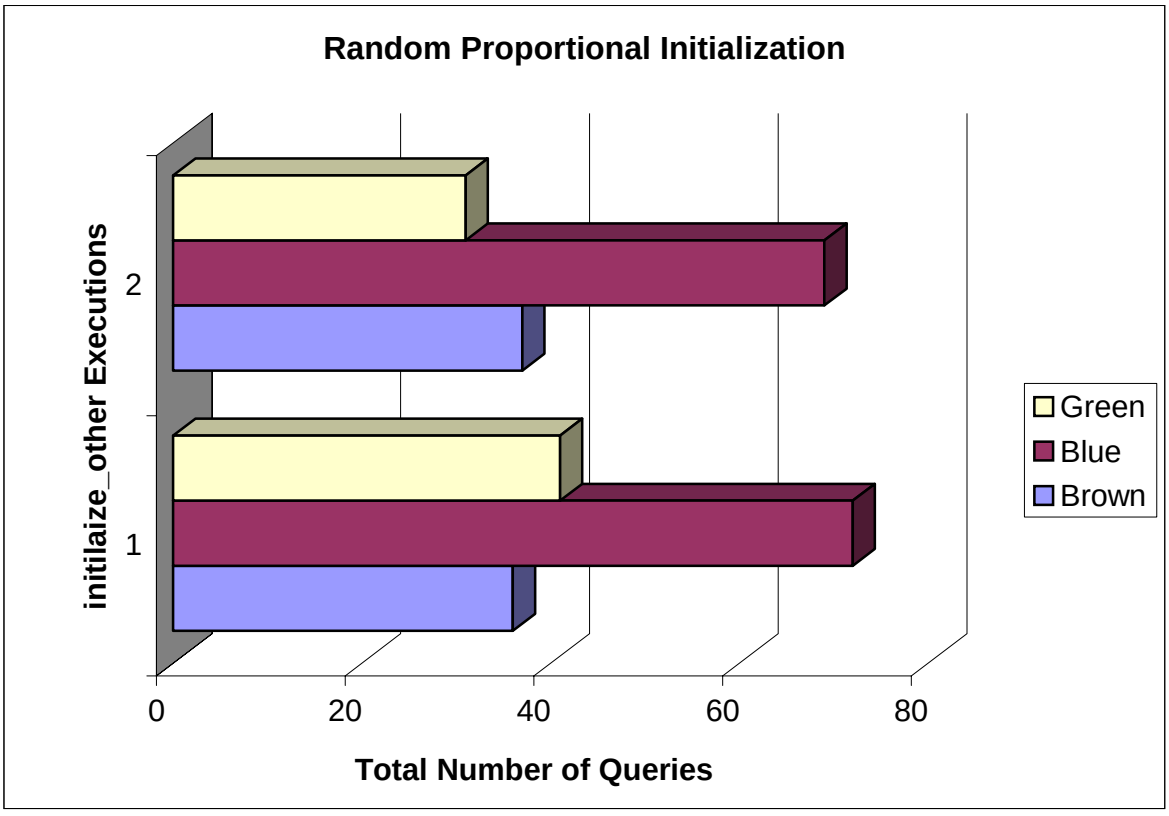
Figure 6. New Random Proportional Initialization Query Results



The average number of queries for the New Random Proportional Initialization Method was 48 queries per attribute. Each attribute's value within the Color table was

generated randomly, their values are dependent since the first attribute is random and the other two resulting attributes are the proportional difference as stated in Section 6.1. The initialization was rotated between the Brown, Blue and Green attribute values randomly selecting one attribute for the random initialization then setting the remaining two proportionally from the first. This was portion was a follow up to the initial random Proportional Initialization described in Chapter 4 and seen below, to eliminate the bias towards the Brown attribute.

Figure 7. Random Proportional Initialization Query Results

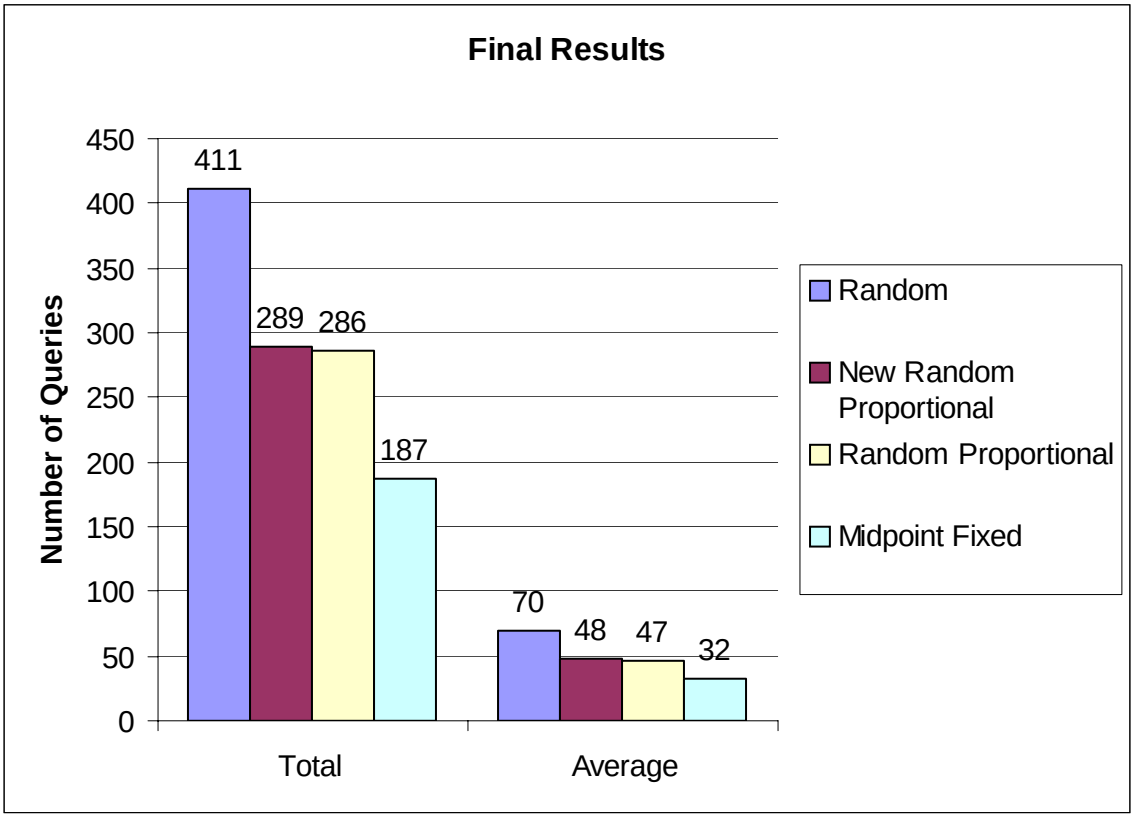


The proposed bias that resulted from performing the portion of the experiment detailed in Chapter 4 was eliminated based on the results. The total number of queries to convergence was consistent for both the Random Proportional and New Random Proportional method. The initial determination of the random attribute did not have a significant affect on the total number of queries till convergence.

CHAPTER 7 CONCLUSIONS

Machine learning theory proposes that random initialization is the best fuzzy membership function initialization method to optimize convergence [26]. However, based on Figure 7, the Midpoint Initialization produced the least number of queries to achieve convergence.

Figure 7. Final Results for all Initialization Methods



Although no reference was found that discouraged the initialization by the Midpoint method, it seemed like a logical result. The queries were performed starting at the "Medium" range followed by the "Slightly" then "Very." By setting all the fuzzy values to

the midpoint, which falls in the “Medium” range, the “Very” and “Slightly” values converged by default. The order of the querying across the Ranges started in the Medium range since the Midpoint initialization method only contained values in that range. Starting the queries in any other range would not return results for that initialization method. All initialization methods contained results within that range initially. Altering the order of the querying would have added one additional query to the Midpoint Initialization method for the Very range. The prototype was developed to solicit user feedback based on a specific query. Since the prototype was designed to gain feedback, changing the ordering of the querying would be impractical.

Since the results were contrary to the Machine Learning findings on initializing the membership values, some additional experiments using the Face table were performed to verify the results [26]. Unlike the Color table, where experiments were conducted on all fuzzy attribute values for Eye Color, this test only covered the Broad value of the Face attribute. The Average and Narrow attributes were ignored.

The Face table was initialized using the same stored procedures used for the Color table portion of the experiment. The initialization and convergence values for the Broad table are contained in Appendix I, Tables77-80.

For the face attribute in the Face table, one evolution of querying was completed; this is referred to as a round. During each round, the queried were divided by the Range values

for the face, Medium, Slightly then Very; this is referred to as a cycle. After a cycle of queries was performed, subsequent cycles were repeated until the Range/Face value combinations were converged. During this process, the total number of queries was collected for each Range/Face pairing. The number of images is consistent across all rounds for each Range/Face value pairs; these values are provided in each table. Each table also provides the total number of images evaluated and the total number of queries per round.

The fuzzy values in the Face table were initialized using the applicable stored procedure. For each execution of the stored procedure, one round of querying were performed. Four stored procedures were executed, one for each initialization method, resulting in four rounds of querying. During this process, the total number of queries were collected for each Range/Face pairing. The number of images is consistent across all rounds for each Range/Face value pair; these values are provided in each table. Each table also provides the total number of images evaluated and the total number of queries per round. The resulting values for those rounds of querying are contained in Tables 28-31

Table 28. Random Initialization Query Results (Round 1)

Random	Totals	Range/Color Combinations		
		Medium Broad	Slightly Broad	Very Broad
Total # of Queries	69	12	43	14
Final # of Images	31	19	10	2

Table 29. Proportional Initialization Query Results (Round 2)

Proportional	Totals	Range/Color Combinations		
		Medium Broad	Slightly Broad	Very Broad
Total # of Queries	44	17	25	2
Final # of Images	31	19	10	2

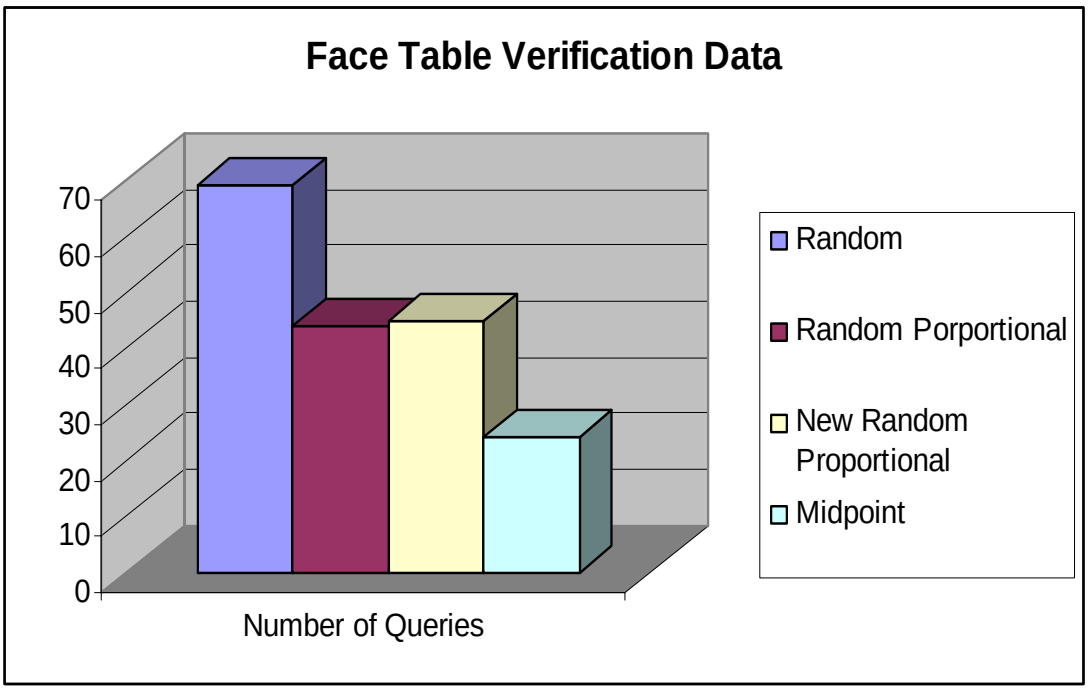
Table 30. Midpoint Fixed Initialization Query Results (Round 3)

Midpoint	Totals	Range/Color Combinations		
		Medium Broad	Slightly Broad	Very Broad
Total # of Queries	24	3	18	3
Final # of Images	31	19	10	2

Table 31. New Random Proportional Initialization Query Results (Round 4)

New Random Proportional	Totals	Range/Color Combinations		
		Medium Broad	Slightly Broad	Very Broad
Total # of Queries	45	18	25	2
Final # of Images	31	4	25	2

Figure 8. Verification Data Summary, Face Table



The results of the verification using the Face table were consistent with results reported in Figure 7 for the portion of the experiment that covered the Color table. This indicates that Midpoint initialization of the fuzzy membership function is the optimum solution to expedite convergence for this particular system. However, there are two main problems with that method of initialization for the current system:

1. In a community-defined membership value system, setting the functions to the midpoint initially makes random querying between the various ranges more difficult. If the user initiates a query in the "Very" range of any table, no results would be returned by the system. Until several users evaluate the midpoint range

values pushing them into either the “Very” or “Slightly” range, both range values would produce no resulting images for the user to evaluate.

2. Advanced Machine Learning techniques would need to be implemented so the system could learn when the rotation between ranges can produce results.. An automated system would need to be implemented to rotate the queries across the variable ranges based on the number of attributes whose values are in those ranges. Since the current prototype is not that advanced, this method was not investigated since further machine-learning algorithms need to be implemented to accomplish this task.

Currently, an investigation was completed to evaluate the determination of membership function values to optimize retrieval of the fuzzy data [29]. This research implemented a machine-learning algorithm to poll a small cross-section of the user community establishing the fuzzy values closer to an estimated community learned convergence values. This research agrees in theory that the values are converged more quickly using a machine-learning algorithm.

Random initialization methods for learning systems are suggested for two reasons:

1. Humans lack the knowledge to set the correct values for the domain of the attribute.
2. Weight of the attribute within the domain must adapt over time.

For the current prototype, both Random Proportional methods converged faster than the Random method. The prototype model attributes related to human physiology such as eye

color and the shape of the face. Each table contains three attributes that could be used to describe the object. Each attribute contains a fuzzy value in the function range; however the values are not totally independent. If a person's Eye color were totally blue then it would not be brown. Even though the values are using fuzzy principles to determine the degree of membership, each attribute is not totally independent of each other. The Random Proportional method attempts to quantify the dependency between the attributes without requiring them to be totally dependent. For instance, a person can have one blue eye and one green eye. If the initialization of the attributes for that person was set to Blue = 0.33, Brown = 0.33 and Green = 0.33, convergence of this person is limited to moving the brown function out of the visible range to say 0.05.

Since the eye color and facial attributes have some intrinsic relationship, the addition of the proportional restriction on the random initialization better models the attributes depicted within this prototype. The Midpoint method is the superior initialization to support fastest convergence. Both methods should be re-evaluated when a machine-learning algorithm is implemented.

CHAPTER 8 FUTURE WORK

Experiments should be conducted to evaluate when a fuzzy attribute value no longer provides value within the database. For instance, if a person has a very high membership value within the blue eyes function, how long should the database store a value for that image in the other membership functions?

Currently the system increments the weighted membership function by a single predefined increment. Regardless of the user feedback, that increment stays the same. A Machine Learning algorithm could be applied to evaluate the user response and alter the increment of the fuzzy membership function value based on the degree of the response or the range of the fuzzy value.

The current prototype allows the user to query based on only one facial feature. The current prototype only contains two facial attributes with fuzzy membership values. Expanding the system to capture more facial image attributes would be helpful if the community needs to classify images more robustly. Additionally, this would require expanding the query interface to handle multiple facial attributes within a single query.

Each fuzzy function had three range variables, Very, Medium and Slightly. Defining synonyms for those range variables that better apply to the facial features would make the system more intuitive.

For example, the Face table has three functions, 1) Broad, 2) Average and 3) Narrow. The range variables “Very” and “Slightly” seem to be intuitive. However “Medium” as it applies to the value Average seems difficult to understand.

Community convergence was not examined during this experiment. Community stability was also not addressed. Experimentation could be completed on community convergence with respect to the influx of new community members as it applies to the different initialization methods.

The Midpoint method was the superior initialization to support fastest convergence. This method should be re-evaluated when a machine-learning algorithm is implemented to sequence the queries across the ranges based on the number of attributes containing values within those ranges.

This experiment did not evaluate the initialization method with respect to the data set. Repeating the experiment with a different data set and compairing the results will allow the elimination of data set dependencies.

LIST OF REFERENCES

1) Aptronix Inc , Fuzzy Interference Development Environment (FIDE), "*What is Fuzzy Logic*", http://www.aptronix.com/fide/whatfuzzy.htm, (Accessed 04/10/2005)

2) Bedi, P, Kaur, H, & Malhotra, A. (2002). "*Fuzzy Dimension to Databases*." Published at the 37th National Convention of Computer Society of India, Bangalore, India. http://www.tsucorp.net/ankit/ankcsi02.doc

3) Bezdek, James C, "*Fuzzy models --- what are they, and why*?", IEEE Transactions on Fuzzy Systems, Volume 1, Number 1, pp. 1-6 (1993)

4) Bhootra, R. and Mehrotra, A., "*Overview of Natural Language Interfaces*", Directed Research Report, Department of Computer Science, VCU, December 2004

5) Bonissone, P, "*A Fuzzy Set Based Linguistic Approach: Theory and Applications*", in Proceedings of the 1980 Winter Simulation Conference, (Eds. J. I. Oren, C. M. Shub, P. F. Roth), pp.99--111, 1980.

6) Bonde, Allen (2000), "*Fuzzy Logic Basic*", http://www.austinlinks.com/Fuzzy/basics.html, (Accessed 02/09/2005)

7) Bosc, P. & Pivert, O. (1995). "*SQLf: A Relational Database Language for Fuzzy Querying*." IEEE Transactions on Fuzzy Systems, Volume 3, pp.1-17

8) Brule, J.(2000), "*Fuzzy Systems a Tutorial*", http://www.austinlinks.com/Fuzzy/tutorial.html , (Accessed 03/23/2005)

9) Carnegie Mellon University Computer Science Department Newsgroup FAQ (03/14/1997): "*Fuzzy Logic and Fuzzy Expert Systems—[2] What Is Fuzzy Logic?"*, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1 , (Accessed 04/14/2005)

10) Codd, E. F. (1970) "*A Relational Model for Large Shared Data Banks.*" Communications of the ACM, 13 (6), 377-187

11) Daly T., Smith W, Clickz (December 2004), "*Convergence: what is it anyway*?", www.clickz.com/experts/archives/tech/convergence/article.php/833801, (Accessed 04/01/2005)

12) Date, C. J. (2003). "*On Fuzzy Databases*" from Database Debunkings, http://www.dbdebunk.com, (Accessed 01/15/2005)

13) Edisbury, B, Fuzzy Application Library/Technical Applications/Aircraft Flight Path (May 1999), "*Fuzzy Logic predicts Air Flight Path*", http://www.fuzzytech.com/e/e_ap_afp.html, (Accessed 04/10/2005)

14) Dubios, D., Pardes, H., "*What does Fuzzy Logic Bring to AI*?", ACM Computing Surveys, Volume 27, Issue 3, pp. 328-330, September 1995

15) Gaines, B.R., "*Fuzzy Reasoning and Logics of Uncertainty*", Proceedings of the sixth international symposium on Multiple-valued Logic, pp. 178-188, 1976

16) Horstkotte, Erik (2000), "*Fuzzy Expert Systems*", http://www.austinlinks.com/Fuzzy/expert-systems.html, (Accessed 03/18/2005)

17) Joy, K., Dattatri, S.,"*Implementing a Fuzzy Relational Database and Querying System with Community Defined Membership Values*", Directed Research Report, Department of Computer Science, VCU, December 2004

18) Kantrowitz, M., Horstkotte, E., Joslyn, C.(March1997), "*Answers to Questions about Fuzzy Logic and Fuzzy Expert Systems*", http://www.cs.cmu.edu/Web/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html, (Accessed 04/12/2005)

19) Date, C. J. (2004). "*A Note on Relation-Valued Attributes*" in An Introduction to Database Systems (8_{th} ed.), Addison-Wesley, 2004, pp. 373-375

20) Fagin, R. (2002). "*Combining Fuzzy Information: An Overview*", SIGMOD Record, 31, pp. 109-118.

21) Kay, Russell, Computer World, Issue 8 (August 30, 2004), "*Fuzzy Logic*", http://www.computerworld.com/databasetopics/data/software/story/0,10801,95497,00.html (Accessed 03/23/2005)

22) Kay, Russell, Computer World, Issue 8 (August 30, 2004)," Sidebar: Seven Truths of Fuzzy Logic", http://www.computerworld.com/news/2004/story/0,11280,95499,00.html (Accessed 03/23/2005)

23) Krause, B., Pozybill, M., von Altrock, C., Fuzzy Application Library/Technical Applications/Traffic Control (1997), "*Data Analysis of Environmental Data for Traffic Control*", http://www.fuzzytech.com/e/e_a_tfc.html,IEEE International

Conference on Fuzzy Logic, (Accessed 04/10/2005)

24) Kwok, K. L. "*A network approach to probabilistic information retrieval*," ACM Transactions on Information Systems, Volume. 13, Number. 3, pp. 324-353, 1995

25) Mathworks (May 2004), "*Fuzzy Logic Toolbox 2.1, Design and Simulate Fuzzy Logic Systems*" https://tagteamdbserver.mathworks.com/ttserverroot/Download/20546_8281v05_FL.pdf, (Accessed 04/10/2005)

26) Mitchell, Tom M., Machine Learning, McGraw Hill 1997, Chapters 1, 2 and 7

27) Motro, A. (1988). "*VAGUE: A User Interface to Relational Databases the Permits Vague Queries.*" ACM Transactions on Office Information Systems, Volume 6, pp.187-214

28) Rigel Corporation (April 2002), "rFLASH User's Guide, Version 1.2", http://www.rigelcorp.com/__doc/8051/rFLASH.pdf, (Accessed 04/10/2005)

29) Sanghi, Shweta, "*Determining Membership Function Values to Optimize Retrieval in a Fuzzy Relational Database"*, Directed Research Report, Department of Computer Science, VCU, May 2005

30) Shafer, Glen, "A Mathematical Theory of Evidence", Princeton University Press, Princeton NJ, 1976

31) von Altrock, Constantin, "*Fuzzy Logic and NeuroFuzzy in Appliances*", Fuzzy Application Library/Technical Applications/Fuzzy in Appliances, http://www.fuzzytech.com/e/e_a_esa.html, Embedded Systems Conferences, 1996

32) Zadeh, L.A. "*Coping with the Imprecision of the Real World, An Interview with Lofti Zadeh*", Communications of the ACM, Volume 27, Issue 4, pp. 301 – 311, 1984

33) Zadeh, L.A., "*Semantic Inference from Fuzzy Premise*", Proceedings of the Sixth International Symposium on Multiple-valued Logic, pp. 217 - 218, 1976

APPENDIX A

RANDOM INITIALIZATION VALUES

Appendix A contains the initialization values that were created within the Color table using the `initialize_random` stored procedure. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Note: Some images of poor quality and eliminated from evaluation during this experiment. The inconsistent ID numbering reflects the images not processed.

Table 32. Random Initialization Values, Color Table (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.88	19	Brown	0.93
2	Brown	0.80	20	Brown	0.53
3	Brown	0.14	21	Brown	0.48
6	Brown	0.58	22	Brown	0.37
8	Brown	0.52	23	Brown	0.59
9	Brown	0.51	24	Brown	0.56
10	Brown	0.72	31	Brown	0.45
11	Brown	0.59	32	Brown	0.23
12	Brown	0.76	33	Brown	0.02
13	Brown	0.82	34	Brown	0.72
14	Brown	0.19	35	Brown	0.89
15	Brown	0.91	36	Brown	0.08
16	Brown	0.37	37	Brown	0.36
17	Brown	0.84	38	Brown	0.47
18	Brown	0.43	39	Brown	0.54
			40	Brown	0.43

Table 33. Random Initialization Values, Color Table (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.17	19	Blue	0.27
2	Blue	0.86	20	Blue	0.38
3	Blue	0.73	21	Blue	0.52
6	Blue	0.01	22	Blue	0.51
8	Blue	0.45	23	Blue	0.52
9	Blue	0.18	24	Blue	0.08
10	Blue	0.75	31	Blue	0.34
11	Blue	0.53	32	Blue	0.66
12	Blue	0.26	33	Blue	0.58
13	Blue	0.03	34	Blue	0.75
14	Blue	0.10	35	Blue	0.82
15	Blue	0.16	36	Blue	0.20
16	Blue	0.30	37	Blue	0.48
17	Blue	0.15	38	Blue	0.71
18	Blue	0.09	39	Blue	0.31
			40	Blue	0.17

Table 34. Random Initialization Values, Color Table (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.46	19	Green	0.09
2	Green	0.03	20	Green	0.39
3	Green	0.91	21	Green	0.81
6	Green	0.91	22	Green	0.98
8	Green	0.56	23	Green	0.82
9	Green	0.99	24	Green	0.26
10	Green	0.24	31	Green	0.87
11	Green	0.86	32	Green	0.39
12	Green	0.77	33	Green	0.44
13	Green	0.85	34	Green	0.20
14	Green	0.85	35	Green	0.66
15	Green	0.54	36	Green	0.30
16	Green	0.60	37	Green	0.04
17	Green	0.86	38	Green	0.15
18	Green	0.76	39	Green	0.82
			40	Green	0.52

Table 35. Random Initialization Values, Color Table (Round 4)

ID	Color	Wt	ID	Color	Wt
1	Brown	0.41	19	Brown	0.53
2	Brown	0.82	20	Brown	0.11
3	Brown	0.63	21	Brown	0.87
6	Brown	0.63	22	Brown	0.56
8	Brown	0.47	23	Brown	0.28
9	Brown	0.66	24	Brown	0.99
10	Brown	0.74	31	Brown	0.73
11	Brown	0.91	32	Brown	0.78
12	Brown	0.81	33	Brown	0.64
13	Brown	0.88	34	Brown	0.04
14	Brown	0.66	35	Brown	0.24
15	Brown	0.45	36	Brown	0.63
16	Brown	0.31	37	Brown	0.05
17	Brown	0.14	38	Brown	0.96
18	Brown	0.68	39	Brown	0.21
			40	Brown	0.10

Table 36. Random Initialization Values, Color Table (Round 5)

ID	Color	Wt	ID	Color	Wt
1	Blue	0.62	19	Blue	0.13
2	Blue	0.22	20	Blue	0.13
3	Blue	0.54	21	Blue	0.33
6	Blue	0.67	22	Blue	0.14
8	Blue	0.24	23	Blue	0.82
9	Blue	0.61	24	Blue	0.06
10	Blue	0.62	31	Blue	0.05
11	Blue	0.43	32	Blue	0.33
12	Blue	0.92	33	Blue	0.65
13	Blue	0.78	34	Blue	0.69
14	Blue	0.73	35	Blue	0.23
15	Blue	0.53	36	Blue	0.48
16	Blue	0.50	37	Blue	0.22
17	Blue	0.51	38	Blue	0.14
18	Blue	0.14	39	Blue	0.33
			40	Blue	0.80

Table 37. Random Initialization Values, Color Table (Round 6)

ID	Color	Wt	ID	Color	Wt
1	Green	0.17	19	Green	0.04
2	Green	0.04	20	Green	0.20
3	Green	0.48	21	Green	0.55
6	Green	0.05	22	Green	0.61
8	Green	0.86	23	Green	0.38
9	Green	0.96	24	Green	0.83
10	Green	0.71	31	Green	0.2
11	Green	0.46	32	Green	0.85
12	Green	0.37	33	Green	0.68
13	Green	0.03	34	Green	0.17
14	Green	0.71	35	Green	0.12
15	Green	0.63	36	Green	0.71
16	Green	0.25	37	Green	0.85
17	Green	0.74	38	Green	0.91
18	Green	0.81	39	Green	0.11
			40	Green	0.98

APPENDIX B

RANDOM INITIALIZATION CONVERGENCE VALUES

Appendix B contains the convergence values that resulted within the Color table after the six rounds of querying were completed. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 38. Random Convergence Values, Color Table (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.41	19	Brown	0.23
2	Brown	0.23	20	Brown	0.51
3	Brown	0.52	21	Brown	0.86
6	Brown	0.52	22	Brown	0.19
8	Brown	0.86	23	Brown	0.53
9	Brown	0.20	24	Brown	0.19
10	Brown	0.19	31	Brown	0.23
11	Brown	0.19	32	Brown	0.52
12	Brown	0.52	33	Brown	0.52
13	Brown	0.88	34	Brown	0.23
14	Brown	0.19	35	Brown	0.23
15	Brown	0.23	36	Brown	0.20
16	Brown	0.19	37	Brown	0.19
17	Brown	0.23	38	Brown	0.19
18	Brown	0.86	39	Brown	0.87
			40	Brown	0.53

Table 39. Random Convergence Values, Color Table (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.52	19	Blue	0.53
2	Blue	0.23	20	Blue	0.53
3	Blue	0.21	21	Blue	0.21
6	Blue	0.19	22	Blue	0.52
8	Blue	0.21	23	Blue	0.21
9	Blue	0.20	24	Blue	0.52
10	Blue	0.87	31	Blue	0.79
11	Blue	0.87	32	Blue	0.21
12	Blue	0.20	33	Blue	0.21
13	Blue	0.21	34	Blue	0.87
14	Blue	0.20	35	Blue	0.88
15	Blue	0.52	36	Blue	0.52
16	Blue	0.52	37	Blue	0.87
17	Blue	0.21	38	Blue	0.52
18	Blue	0.19	39	Blue	0.20
			40	Blue	0.19

Table 40. Random Convergence Values, Color Table (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.21	19	Green	0.19
2	Green	0.81	20	Green	0.53
3	Green	0.23	21	Green	0.23
6	Green	0.23	22	Green	0.23
8	Green	0.21	23	Green	0.23
9	Green	0.23	24	Green	0.20
10	Green	0.20	31	Green	0.23
11	Green	0.86	32	Green	0.21
12	Green	0.23	33	Green	0.21
13	Green	0.23	34	Green	0.20
14	Green	0.50	35	Green	0.21
15	Green	0.21	36	Green	0.21
16	Green	0.52	37	Green	0.20
17	Green	0.52	38	Green	0.21
18	Green	0.23	39	Green	0.23
			40	Green	0.21

Table 41. Random Convergence Values, Color Table (Round 4)

ID	Color	Wt	ID	Color	Wt
1	Brown	0.19	19	Brown	0.27
2	Brown	0.26	20	Brown	0.21
3	Brown	0.51	21	Brown	0.87
6	Brown	0.87	22	Brown	0.20
8	Brown	0.87	23	Brown	0.52
9	Brown	0.52	24	Brown	0.27
10	Brown	0.20	31	Brown	0.19
11	Brown	0.27	32	Brown	0.86
12	Brown	0.27	33	Brown	0.52
13	Brown	0.52	34	Brown	0.20
14	Brown	0.52	35	Brown	0.20
15	Brown	0.19	36	Brown	0.19
16	Brown	0.19	37	Brown	0.19
17	Brown	0.20	38	Brown	0.26
18	Brown	0.88	39	Brown	0.51
			40	Brown	0.52

Table 42. Random Convergence Values, Color Table Values (Round 5)

ID	Color	Wt	ID	Color	Wt
1	Blue	0.52	19	Blue	0.19
2	Blue	0.30	20	Blue	0.19
3	Blue	0.20	21	Blue	0.19
6	Blue	0.19	22	Blue	0.30
8	Blue	0.20	23	Blue	0.20
9	Blue	0.19	24	Blue	0.52
10	Blue	0.52	31	Blue	0.19
11	Blue	0.87	32	Blue	0.19
12	Blue	0.20	33	Blue	0.19
13	Blue	0.20	34	Blue	0.31
14	Blue	0.51	35	Blue	0.53
15	Blue	0.51	36	Blue	0.52
16	Blue	0.52	37	Blue	0.52
17	Blue	0.53	38	Blue	0.52
18	Blue	0.20	39	Blue	0.19
			40	Blue	0.20

Table 43. Random Convergence Values, Color Table (Round 6)

ID	Color	Wt	ID	Color	Wt
1	Green	0.21	19	Green	0.20
2	Green	0.78	20	Green	0.20
3	Green	0.20	21	Green	0.21
6	Green	0.21	22	Green	0.21
8	Green	0.26	23	Green	0.20
9	Green	0.26	24	Green	0.53
10	Green	0.21	31	Green	0.20
11	Green	0.88	32	Green	0.27
12	Green	0.21	33	Green	0.20
13	Green	0.19	34	Green	0.21
14	Green	0.21	35	Green	0.20
15	Green	0.21	36	Green	0.51
16	Green	0.51	37	Green	0.27
17	Green	0.52	38	Green	0.27
18	Green	0.27	39	Green	0.19
			40	Green	0.26

APPENDIX C

RANDOM PROPORTIONAL INITIALIZATION VALUES

Appendix C contains the initialization values that were created within the Color table using the `initialize_other` stored procedure. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 44. Random Proportional Initialization Values, Color Table (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.03	19	Brown	0.09
2	Brown	0.15	20	Brown	0.38
3	Brown	0.04	21	Brown	0.36
6	Brown	0.44	22	Brown	0.43
8	Brown	0.24	23	Brown	0.32
9	Brown	0.01	24	Brown	0.03
10	Brown	0.43	31	Brown	0.41
11	Brown	0.1	32	Brown	0.27
12	Brown	0.42	33	Brown	0.17
13	Brown	0.21	34	Brown	0.25
14	Brown	0.29	35	Brown	0.44
15	Brown	0.14	36	Brown	0.06
16	Brown	0.08	37	Brown	0.21
17	Brown	0.19	38	Brown	0.49
18	Brown	0.42	39	Brown	0.03
			40	Brown	0.15

Table 45. Random Proportional Initialization Values, Color Table (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.93	19	Blue	0.82
2	Blue	0.71	20	Blue	0.24
3	Blue	0.91	21	Blue	0.28
6	Blue	0.13	22	Blue	0.13
8	Blue	0.51	23	Blue	0.37
9	Blue	0.98	24	Blue	0.93
10	Blue	0.15	31	Blue	0.19
11	Blue	0.79	32	Blue	0.46
12	Blue	0.16	33	Blue	0.67
13	Blue	0.59	34	Blue	0.51
14	Blue	0.41	35	Blue	0.12
15	Blue	0.73	36	Blue	0.88
16	Blue	0.84	37	Blue	0.58
17	Blue	0.62	38	Blue	0.02
18	Blue	0.15	39	Blue	0.95
			40	Blue	0.71

Table 46. Random Proportional Initialization Values, Color Table (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.03	19	Green	0.09
2	Green	0.15	20	Green	0.38
3	Green	0.04	21	Green	0.36
6	Green	0.44	22	Green	0.43
8	Green	0.24	23	Green	0.32
9	Green	0.01	24	Green	0.03
10	Green	0.43	31	Green	0.41
11	Green	0.1	32	Green	0.27
12	Green	0.42	33	Green	0.17
13	Green	0.21	34	Green	0.25
14	Green	0.29	35	Green	0.44
15	Green	0.14	36	Green	0.06
16	Green	0.08	37	Green	0.21
17	Green	0.19	38	Green	0.49
18	Green	0.42	39	Green	0.03
			40	Green	0.15

Table 47. Random Proportional Initialization Values, Color Table (Round 4)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.24	19	Brown	0.24
2	Brown	0.22	20	Brown	0.49
3	Brown	0.35	21	Brown	0.19
6	Brown	0.37	22	Brown	0.38
8	Brown	0.37	23	Brown	0.11
9	Brown	0.18	24	Brown	0.31
10	Brown	0.29	31	Brown	0.21
11	Brown	0.23	32	Brown	0.10
12	Brown	0.44	33	Brown	0.34
13	Brown	0.04	34	Brown	0.45
14	Brown	0.28	35	Brown	0.42
15	Brown	0.10	36	Brown	0.50
16	Brown	0.10	37	Brown	0.30
17	Brown	0.14	38	Brown	0.44
18	Brown	0.06	39	Brown	0.03
			40	Brown	0.50

Table 48. Random Proportional Initialization Values, Color Table (Round 5)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.53	19	Blue	0.51
2	Blue	0.56	20	Blue	0.03
3	Blue	0.31	21	Blue	0.61
6	Blue	0.26	22	Blue	0.24
8	Blue	0.26	23	Blue	0.78
9	Blue	0.64	24	Blue	0.37
10	Blue	0.41	31	Blue	0.58
11	Blue	0.55	32	Blue	0.80
12	Blue	0.12	33	Blue	0.32
13	Blue	0.92	34	Blue	0.10
14	Blue	0.44	35	Blue	0.15
15	Blue	0.81	36	Blue	0.01
16	Blue	0.79	37	Blue	0.40
17	Blue	0.72	38	Blue	0.12
18	Blue	0.88	39	Blue	0.95
			40	Blue	0.01

Table 49. Random Proportional Initialization Values, Color Table (Round 6)

ID	Function	Wt	ID	Function	Wt
1	Green	0.24	19	Green	0.24
2	Green	0.22	20	Green	0.49
3	Green	0.35	21	Green	0.19
6	Green	0.37	22	Green	0.38
8	Green	0.37	23	Green	0.11
9	Green	0.18	24	Green	0.31
10	Green	0.29	31	Green	0.21
11	Green	0.23	32	Green	0.10
12	Green	0.44	33	Green	0.34
13	Green	0.04	34	Green	0.45
14	Green	0.28	35	Green	0.42
15	Green	0.10	36	Green	0.50
16	Green	0.10	37	Green	0.30
17	Green	0.14	38	Green	0.44
18	Green	0.06	39	Green	0.03
			40	Green	0.50

APPENDIX D

RANDOM PROPORTIONAL INITIALIZATION CONVERGENCE VALUES

Appendix D contains the convergence values that resulted within the Color table after the six rounds of querying were completed. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 50. Random Proportional Convergence, Color Values (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.19	19	Brown	0.21
2	Brown	0.19	20	Brown	0.19
3	Brown	0.51	21	Brown	0.79
6	Brown	0.52	22	Brown	0.19
8	Brown	0.79	23	Brown	0.52
9	Brown	0.21	24	Brown	0.19
10	Brown	0.19	31	Brown	0.52
11	Brown	0.20	32	Brown	0.77
12	Brown	0.53	33	Brown	0.52
13	Brown	0.73	34	Brown	0.21
14	Brown	0.20	35	Brown	0.19
15	Brown	0.20	36	Brown	0.20
16	Brown	0.20	37	Brown	0.21
17	Brown	0.52	38	Brown	0.19
18	Brown	0.81	39	Brown	0.78
			40	Brown	0.52

Table 51. Random Proportional Convergence, Color Values (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.51	19	Blue	0.25
2	Blue	0.25	20	Blue	0.20
3	Blue	0.25	21	Blue	0.20
6	Blue	0.21	22	Blue	0.53
8	Blue	0.19	23	Blue	0.19
9	Blue	0.52	24	Blue	0.51
10	Blue	0.52	31	Blue	0.53
11	Blue	0.87	32	Blue	0.19
12	Blue	0.20	33	Blue	0.25
13	Blue	0.19	34	Blue	0.53
14	Blue	0.51	35	Blue	0.52
15	Blue	0.51	36	Blue	0.52
16	Blue	0.52	37	Blue	0.52
17	Blue	0.25	38	Blue	0.52
18	Blue	0.19	39	Blue	0.25
			40	Blue	0.19

Table 52. Random Proportional Convergence, Color Values (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.19	19	Green	0.23
2	Green	0.77	20	Green	0.19
3	Green	0.52	21	Green	0.19
6	Green	0.19	22	Green	0.19
8	Green	0.20	23	Green	0.19
9	Green	0.52	24	Green	0.19
10	Green	0.19	31	Green	0.19
11	Green	0.27	32	Green	0.19
12	Green	0.27	33	Green	0.21
13	Green	0.21	34	Green	0.21
14	Green	0.52	35	Green	0.19
15	Green	0.20	36	Green	0.20
16	Green	0.52	37	Green	0.21
17	Green	0.77	38	Green	0.19
18	Green	0.19	39	Green	0.19
			40	Green	0.19

Table 53. Random Proportional Convergence, Color Values (Round 4)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.24	19	Brown	0.24
2	Brown	0.22	20	Brown	0.49
3	Brown	0.35	21	Brown	0.19
6	Brown	0.37	22	Brown	0.38
8	Brown	0.37	23	Brown	0.11
9	Brown	0.18	24	Brown	0.31
10	Brown	0.29	31	Brown	0.21
11	Brown	0.23	32	Brown	0.10
12	Brown	0.44	33	Brown	0.34
13	Brown	0.04	34	Brown	0.45
14	Brown	0.28	35	Brown	0.42
15	Brown	0.10	36	Brown	0.50
16	Brown	0.10	37	Brown	0.30
17	Brown	0.14	38	Brown	0.44
18	Brown	0.06	39	Brown	0.03
			40	Brown	0.50

Table 54. Random Proportional Convergence, Color Values (Round 5)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.53	19	Blue	0.51
2	Blue	0.56	20	Blue	0.03
3	Blue	0.31	21	Blue	0.61
6	Blue	0.26	22	Blue	0.24
8	Blue	0.26	23	Blue	0.78
9	Blue	0.64	24	Blue	0.37
10	Blue	0.41	31	Blue	0.58
11	Blue	0.55	32	Blue	0.80
12	Blue	0.12	33	Blue	0.32
13	Blue	0.92	34	Blue	0.10
14	Blue	0.44	35	Blue	0.15
15	Blue	0.81	36	Blue	0.01
16	Blue	0.79	37	Blue	0.40
17	Blue	0.72	38	Blue	0.12
18	Blue	0.28	39	Blue	0.95
			40	Blue	0.01

Table 55. Random Proportional Convergence, Color Values (Round 6)

ID	Function	Wt	ID	Function	Wt
1	Green	0.24	19	Green	0.24
2	Green	0.22	20	Green	0.49
3	Green	0.35	21	Green	0.19
6	Green	0.37	22	Green	0.38
8	Green	0.37	23	Green	0.11
9	Green	0.18	24	Green	0.31
10	Green	0.29	31	Green	0.21
11	Green	0.23	32	Green	0.10
12	Green	0.44	33	Green	0.34
13	Green	0.04	34	Green	0.45
14	Green	0.28	35	Green	0.42
15	Green	0.10	36	Green	0.50
16	Green	0.10	37	Green	0.30
17	Green	0.14	38	Green	0.44
18	Green	0.06	39	Green	0.03
			40	Green	0.50

APPENDIX E

MIDPOINT INITIALIZATION VALUES

Appendix E contains the initialization values that were created within the Color table using the `initialize_mids` stored procedure. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 56. Midpoint Initialization Values, Color Table (Round 1 & Round 4)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.50	19	Brown	0.50
2	Brown	0.50	20	Brown	0.50
3	Brown	0.50	21	Brown	0.50
6	Brown	0.50	22	Brown	0.50
8	Brown	0.50	23	Brown	0.50
9	Brown	0.50	24	Brown	0.50
10	Brown	0.50	31	Brown	0.50
11	Brown	0.50	32	Brown	0.50
12	Brown	0.50	33	Brown	0.50
13	Brown	0.50	34	Brown	0.50
14	Brown	0.50	35	Brown	0.50
15	Brown	0.50	36	Brown	0.50
16	Brown	0.50	37	Brown	0.50
17	Brown	0.50	38	Brown	0.50
18	Brown	0.50	39	Brown	0.50
			40	Brown	0.50

Table 57. Midpoint Initialization Values, Color Table (Round 2 & Round 5)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.50	19	Blue	0.50
2	Blue	0.50	20	Blue	0.50
3	Blue	0.50	21	Blue	0.50
6	Blue	0.50	22	Blue	0.50
8	Blue	0.50	23	Blue	0.50
9	Blue	0.50	24	Blue	0.50
10	Blue	0.50	31	Blue	0.50
11	Blue	0.50	32	Blue	0.50
12	Blue	0.50	33	Blue	0.50
13	Blue	0.50	34	Blue	0.50
14	Blue	0.50	35	Blue	0.50
15	Blue	0.50	36	Blue	0.50
16	Blue	0.50	37	Blue	0.50
17	Blue	0.50	38	Blue	0.50
18	Blue	0.50	39	Blue	0.50
			40	Blue	0.50

Table 58. Midpoint Initialization Values, Color Table (Round 3 & Round 6)

ID	Function	Wt	ID	Function	Wt
1	Green	0.50	19	Green	0.50
2	Green	0.50	20	Green	0.50
3	Green	0.50	21	Green	0.50
6	Green	0.50	22	Green	0.50
8	Green	0.50	23	Green	0.50
9	Green	0.50	24	Green	0.50
10	Green	0.50	31	Green	0.50
11	Green	0.50	32	Green	0.50
12	Green	0.50	33	Green	0.50
13	Green	0.50	34	Green	0.50
14	Green	0.50	35	Green	0.50
15	Green	0.50	36	Green	0.50
16	Green	0.50	37	Green	0.50
17	Green	0.50	38	Green	0.50
18	Green	0.50	39	Green	0.50
			40	Green	0.50

APPENDIX F

MIDPOINT INITIALIZATION CONVERGENCE VALUES

Appendix E contains the convergence values that resulted within the Color table after the six rounds of querying were completed. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 59. Midpoint Convergence Values, Color Table (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.23	19	Brown	0.23
2	Brown	0.23	20	Brown	0.52
3	Brown	0.52	21	Brown	0.79
6	Brown	0.51	22	Brown	0.23
8	Brown	0.79	23	Brown	0.52
9	Brown	0.52	24	Brown	0.23
10	Brown	0.23	31	Brown	0.23
11	Brown	0.23	32	Brown	0.79
12	Brown	0.53	33	Brown	0.52
13	Brown	0.79	34	Brown	0.23
14	Brown	0.23	35	Brown	0.23
15	Brown	0.23	36	Brown	0.23
16	Brown	0.23	37	Brown	0.23
17	Brown	0.23	38	Brown	0.23
18	Brown	0.79	39	Brown	0.51
			40	Brown	0.52

Table 60. Midpoint Convergence Values, Color Table (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.52	19	Blue	0.23
2	Blue	0.23	21	Blue	0.23
3	Blue	0.23	22	Blue	0.52
6	Blue	0.23	23	Blue	0.23
8	Blue	0.23	24	Blue	0.27
9	Blue	0.51	31	Blue	0.52
10	Blue	0.52	32	Blue	0.23
11	Blue	0.79	33	Blue	0.51
12	Blue	0.23	20	Blue	0.52
13	Blue	0.23	34	Blue	0.52
14	Blue	0.52	35	Blue	0.52
15	Blue	0.27	36	Blue	0.52
16	Blue	0.52	37	Blue	0.52
17	Blue	0.52	38	Blue	0.27
18	Blue	0.23	39	Blue	0.23
			40	Blue	0.23

Table 61. Midpoint Convergence Values, Color Table (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.21	19	Green	0.21
2	Green	0.83	20	Green	0.21
3	Green	0.34	21	Green	0.21
6	Green	0.21	22	Green	0.21
8	Green	0.21	23	Green	0.21
9	Green	0.21	24	Green	0.21
10	Green	0.21	31	Green	0.21
11	Green	0.21	32	Green	0.21
12	Green	0.21	33	Green	0.23
13	Green	0.21	34	Green	0.34
14	Green	0.51	35	Green	0.21
15	Green	0.21	36	Green	0.21
16	Green	0.33	37	Green	0.21
17	Green	0.81	38	Green	0.21
18	Green	0.21	39	Green	0.21
			40	Green	0.21

Table 62. Midpoint Convergence Values, Color Table (Round 4)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.26	19	Brown	0.26
2	Brown	0.26	20	Brown	0.26
3	Brown	0.52	21	Brown	0.82
6	Brown	0.52	22	Brown	0.26
8	Brown	0.82	23	Brown	0.52
9	Brown	0.52	24	Brown	0.26
10	Brown	0.26	31	Brown	0.26
11	Brown	0.26	32	Brown	0.82
12	Brown	0.52	33	Brown	0.52
13	Brown	0.82	34	Brown	0.26
14	Brown	0.32	35	Brown	0.26
15	Brown	0.26	36	Brown	0.26
16	Brown	0.26	37	Brown	0.26
17	Brown	0.26	38	Brown	0.26
18	Brown	0.52	39	Brown	0.80
			40	Brown	0.52

Table 63. Midpoint Convergence Values, Color Table (Round 5)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.52	19	Blue	0.26
2	Blue	0.56	20	Blue	0.40
3	Blue	0.26	21	Blue	0.26
6	Blue	0.26	22	Blue	0.52
8	Blue	0.26	23	Blue	0.26
9	Blue	0.26	24	Blue	0.52
10	Blue	0.52	31	Blue	0.52
11	Blue	0.78	32	Blue	0.26
12	Blue	0.26	33	Blue	0.26
13	Blue	0.26	34	Blue	0.52
14	Blue	0.26	35	Blue	0.52
15	Blue	0.52	36	Blue	0.52
16	Blue	0.52	37	Blue	0.52
17	Blue	0.26	38	Blue	0.52
18	Blue	0.26	39	Blue	0.26
			40	Blue	0.26

Table 64. Midpoint Convergence Values, Color Table (Round 6)

ID	Function	Wt	ID	Function	Wt
1	Green	0.52	19	Green	0.26
2	Green	0.78	20	Green	0.26
3	Green	0.26	21	Green	0.26
6	Green	0.26	22	Green	0.26
8	Green	0.26	23	Green	0.26
9	Green	0.26	24	Green	0.26
10	Green	0.26	31	Green	0.26
11	Green	0.78	32	Green	0.26
12	Green	0.26	33	Green	0.26
13	Green	0.26	34	Green	0.26
14	Green	0.52	35	Green	0.26
15	Green	0.26	36	Green	0.26
16	Green	0.52	37	Green	0.26
17	Green	0.52	38	Green	0.26
18	Green	0.26	39	Green	0.26
			40	Green	0.26

APPENDIX G

NEW RANDOM PROPORTIONAL INITIALIZATION VALUES

Appendix G contains the initialization values that were created within the Color table using the `initialize_random_new` stored procedure. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 65. New Random Proportional Initialization Values, Color Table (Round 1)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.36	19	Brown	0.40
2	Brown	0.48	20	Brown	0.01
3	Brown	0.47	21	Brown	0.38
6	Brown	0.38	22	Brown	0.02
8	Brown	0.44	23	Brown	0.06
9	Brown	0.44	24	Brown	0.05
10	Brown	0.33	31	Brown	0.02
11	Brown	0.38	32	Brown	0.22
12	Brown	0.27	33	Brown	0.03
13	Brown	0.40	34	Brown	0.34
14	Brown	0.03	35	Brown	0.37
15	Brown	0.24	36	Brown	0.27
16	Brown	0.32	37	Brown	0.32
17	Brown	0.42	38	Brown	0.48
18	Brown	0.32	39	Brown	0.48
			40	Brown	0.33

Table 66. New Random Proportional Initialization Values, Color Table (Round 2)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.29	19	Blue	0.20
2	Blue	0.04	20	Blue	0.97
3	Blue	0.05	21	Blue	0.24
6	Blue	0.25	22	Blue	0.96
8	Blue	0.13	23	Blue	0.87
9	Blue	0.12	24	Blue	0.90
10	Blue	0.34	31	Blue	0.96
11	Blue	0.23	32	Blue	0.56
12	Blue	0.45	33	Blue	0.94
13	Blue	0.20	34	Blue	0.31
14	Blue	0.95	35	Blue	0.27
15	Blue	0.52	36	Blue	0.45
16	Blue	0.37	37	Blue	0.36
17	Blue	0.15	38	Blue	0.04
18	Blue	0.37	39	Blue	0.04
			40	Blue	0.33

Table 67. New Random Proportional Initialization Values, Color Table (Round 3)

ID	Function	Wt	ID	Function	Wt
1	Green	0.36	19	Green	0.40
2	Green	0.48	20	Green	0.01
3	Green	0.47	21	Green	0.38
6	Green	0.38	22	Green	0.02
8	Green	0.44	23	Green	0.06
9	Green	0.44	24	Green	0.05
10	Green	0.33	31	Green	0.02
11	Green	0.38	32	Green	0.22
12	Green	0.27	33	Green	0.03
13	Green	0.40	34	Green	0.34
14	Green	0.03	35	Green	0.37
15	Green	0.24	36	Green	0.27
16	Green	0.32	37	Green	0.32
17	Green	0.42	38	Green	0.48
18	Green	0.32	39	Green	0.48
			40	Green	0.33

Table 68. New Random Proportional Initialization Values, Color Table (Round 4)

ID	Function	Wt	ID	Function	Wt
1	Brown	0.24	19	Brown	0.24
2	Brown	0.22	20	Brown	0.49
3	Brown	0.35	21	Brown	0.19
6	Brown	0.37	22	Brown	0.38
8	Brown	0.37	23	Brown	0.11
9	Brown	0.18	24	Brown	0.31
10	Brown	0.29	31	Brown	0.21
11	Brown	0.23	32	Brown	0.10
12	Brown	0.44	33	Brown	0.34
13	Brown	0.04	34	Brown	0.45
14	Brown	0.28	35	Brown	0.42
15	Brown	0.10	36	Brown	0.50
16	Brown	0.10	37	Brown	0.30
17	Brown	0.14	38	Brown	0.44
18	Brown	0.06	39	Brown	0.03
			40	Brown	0.50

Table 69. New Random Proportional Initialization Values, Color Table (Round 5)

ID	Function	Wt	ID	Function	Wt
1	Blue	0.53	19	Blue	0.51
2	Blue	0.56	20	Blue	0.03
3	Blue	0.31	21	Blue	0.61
6	Blue	0.26	22	Blue	0.24
8	Blue	0.26	23	Blue	0.78
9	Blue	0.64	24	Blue	0.37
10	Blue	0.41	31	Blue	0.58
11	Blue	0.55	32	Blue	0.80
12	Blue	0.12	33	Blue	0.32
13	Blue	0.92	34	Blue	0.10
14	Blue	0.44	35	Blue	0.15
15	Blue	0.81	36	Blue	0.01
16	Blue	0.79	37	Blue	0.40
17	Blue	0.72	38	Blue	0.12
18	Blue	0.88	39	Blue	0.95
			40	Blue	0.01

Table 70. New Random Proportional Initialization Values, Color Table (Round 6)

ID	Function	Wt	ID	Function	Wt
1	Green	0.24	19	Green	0.24
2	Green	0.22	20	Green	0.49
3	Green	0.35	21	Green	0.19
6	Green	0.37	22	Green	0.38
8	Green	0.37	23	Green	0.11
9	Green	0.18	24	Green	0.31
10	Green	0.29	31	Green	0.21
11	Green	0.23	32	Green	0.10
12	Green	0.44	33	Green	0.34
13	Green	0.04	34	Green	0.45
14	Green	0.28	35	Green	0.42
15	Green	0.10	36	Green	0.50
16	Green	0.10	37	Green	0.30
17	Green	0.14	38	Green	0.44
18	Green	0.06	39	Green	0.03
			40	Green	0.50

APPENDIX H

NEW RANDOM PROPORTIONAL INITIALIZATION CONVERGENCE VALUES

Appendix H contains the convergence values that resulted within the Color table after the six rounds of querying were completed. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 71. New Random Proportional Convergence Values, Color Table (Round 1)

ID	Color	Wt	ID	Color	Wt
1	Brown	0.20	19	Brown	0.52
2	Brown	0.20	20	Brown	0.21
3	Brown	0.53	21	Brown	0.77
6	Brown	0.79	22	Brown	0.20
8	Brown	0.79	23	Brown	0.51
9	Brown	0.52	24	Brown	0.21
10	Brown	0.21	31	Brown	0.21
11	Brown	0.19	32	Brown	0.78
12	Brown	0.52	33	Brown	0.52
13	Brown	0.52	34	Brown	0.21
14	Brown	0.20	35	Brown	0.20
15	Brown	0.20	36	Brown	0.20
16	Brown	0.20	37	Brown	0.20
17	Brown	0.20	38	Brown	0.20
18	Brown	0.52	39	Brown	0.77
			40	Brown	0.52

Table 72. New Random Proportional Convergence Values, Color Table (Round 2)

ID	Color	Wt	ID	Color	Wt
1	Blue	0.53	19	Blue	0.21
2	Blue	0.52	20	Blue	0.19
3	Blue	0.21	21	Blue	0.21
6	Blue	0.20	22	Blue	0.52
8	Blue	0.20	23	Blue	0.28
9	Blue	0.20	24	Blue	0.53
10	Blue	0.53	31	Blue	0.52
11	Blue	0.87	32	Blue	0.28
12	Blue	0.20	33	Blue	0.20
13	Blue	0.28	34	Blue	0.52
14	Blue	0.52	35	Blue	0.53
15	Blue	0.51	36	Blue	0.53
16	Blue	0.51	37	Blue	0.52
17	Blue	0.28	38	Blue	0.52
18	Blue	0.76	39	Blue	0.29
			40	Blue	0.21

Table 73. New Random Proportional Convergence Values, Color Table (Round 3)

ID	Color	Wt	ID	Color	Wt
1	Green	0.20	19	Green	0.20
2	Green	0.78	20	Green	0.53
3	Green	0.19	21	Green	0.21
6	Green	0.19	22	Green	0.20
8	Green	0.19	23	Green	0.21
9	Green	0.20	24	Green	0.19
10	Green	0.19	31	Green	0.19
11	Green	0.77	32	Green	0.20
12	Green	0.20	33	Green	0.20
13	Green	0.20	34	Green	0.53
14	Green	0.20	35	Green	0.20
15	Green	0.20	36	Green	0.52
16	Green	0.52	37	Green	0.20
17	Green	0.20	38	Green	0.20
18	Green	0.20	39	Green	0.21
			40	Green	0.20

Table 74. New Random Proportional Convergence Values, Color Table (Round 4)

ID	Color	Wt	ID	Color	Wt
1	Brown	0.20	19	Brown	0.52
2	Brown	0.22	20	Brown	0.21
3	Brown	0.56	21	Brown	0.77
6	Brown	0.82	22	Brown	0.20
8	Brown	0.84	23	Brown	0.54
9	Brown	0.50	24	Brown	0.21
10	Brown	0.10	31	Brown	0.21
11	Brown	0.16	32	Brown	0.76
12	Brown	0.52	33	Brown	0.56
13	Brown	0.52	34	Brown	0.18
14	Brown	0.20	35	Brown	0.20
15	Brown	0.20	36	Brown	0.20
16	Brown	0.20	37	Brown	0.18
17	Brown	0.20	38	Brown	0.18
18	Brown	0.52	39	Brown	0.78
			40	Brown	0.56

Table 75. New Random Proportional Convergence Values, Color Table (Round 5)

ID	Color	Wt	ID	Color	Wt
1	Blue	0.62	19	Blue	0.12
2	Blue	0.52	20	Blue	0.17
3	Blue	0.18	21	Blue	0.21
6	Blue	0.22	22	Blue	0.54
8	Blue	0.22	23	Blue	0.26
9	Blue	0.22	24	Blue	0.53
10	Blue	0.56	31	Blue	0.52
11	Blue	0.88	32	Blue	0.24
12	Blue	0.22	33	Blue	0.20
13	Blue	0.28	34	Blue	0.50
14	Blue	0.50	35	Blue	0.50
15	Blue	0.50	36	Blue	0.51
16	Blue	0.50	37	Blue	0.52
17	Blue	0.22	38	Blue	0.52
18	Blue	0.76	39	Blue	0.29
			40	Blue	0.20

Table 76. New Random Proportional Convergence Values, Color Table (Round 6)

ID	Color	Wt	ID	Color	Wt
1	Green	0.20	19	Green	0.20
2	Green	0.84	20	Green	0.52
3	Green	0.19	21	Green	0.21
6	Green	0.19	22	Green	0.20
8	Green	0.19	23	Green	0.21
9	Green	0.20	24	Green	0.19
10	Green	0.19	31	Green	0.19
11	Green	0.80	32	Green	0.20
12	Green	0.20	33	Green	0.20
13	Green	0.20	34	Green	0.50
14	Green	0.20	35	Green	0.20
15	Green	0.20	36	Green	0.50
16	Green	0.52	37	Green	0.20
17	Green	0.20	38	Green	0.20
18	Green	0.20	39	Green	0.21
			40	Green	0.20

APPENDIX I

VERIFICATION DATA USING FACE TABLE

Appendix I contains the initialization values that were created within the Face table using the `each` stored procedure as well as the convergence values that resulted within the Face table after the six rounds of querying were completed. This appendix contains one table per round of queries (six total) each relating to a Range/Color pairing.

Table 77. Random Initialization Verification Data for the Face Table

Random Initialization Values						Random Convergence Values					
ID	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**
1	Broad	0.48	19	Broad	0.40	1	Broad	0.19	19	Broad	0.19
2	Broad	0.90	20	Broad	0.46	2	Broad	0.52	20	Broad	0.19
3	Broad	0.02	21	Broad	0.28	3	Broad	0.52	21	Broad	0.76
6	Broad	0.19	22	Broad	0.31	6	Broad	0.19	22	Broad	0.20
8	Broad	0.20	23	Broad	0.03	8	Broad	0.20	23	Broad	0.52
9	Broad	0.24	24	Broad	0.02	9	Broad	0.20	24	Broad	0.20
10	Broad	0.60	31	Broad	0.40	10	Broad	0.19	31	Broad	0.52
11	Broad	0.04	32	Broad	0.48	11	Broad	0.20	32	Broad	0.19
12	Broad	0.63	33	Broad	0.43	12	Broad	0.53	33	Broad	0.19
13	Broad	0.66	34	Broad	0.67	13	Broad	0.52	34	Broad	0.19
14	Broad	0.49	35	Broad	0.26	14	Broad	0.19	35	Broad	0.20
15	Broad	0.90	36	Broad	0.83	15	Broad	0.88	36	Broad	0.53
16	Broad	0.40	37	Broad	0.60	16	Broad	0.52	37	Broad	0.19
17	Broad	0.34	38	Broad	0.28	17	Broad	0.19	38	Broad	0.20
18	Broad	0.84	39	Broad	0.45	18	Broad	0.25	39	Broad	0.51
			40	Broad	0.97				40	Broad	0.52

Table 78. Random Proportional Initialization Verification Data for the Face Table

Random Proportional Initialization Values						Random Proportional Convergence Values					
ID	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**
1	Broad	0.38	19	Broad	0.38	1	Broad	0.19	19	Broad	0.19
2	Broad	0.32	20	Broad	0.12	2	Broad	0.19	20	Broad	0.20
3	Broad	0.44	21	Broad	0.25	3	Broad	0.51	21	Broad	0.78
6	Broad	0.24	22	Broad	0.04	6	Broad	0.20	22	Broad	0.52
8	Broad	0.05	23	Broad	0.23	8	Broad	0.21	23	Broad	0.52
9	Broad	0.03	24	Broad	0.23	9	Broad	0.19	24	Broad	0.19
10	Broad	0.46	31	Broad	0.06	10	Broad	0.52	31	Broad	0.52
11	Broad	0.45	32	Broad	0.21	11	Broad	0.19	32	Broad	0.21
12	Broad	0.38	33	Broad	0.26	12	Broad	0.52	33	Broad	0.20
13	Broad	0.39	34	Broad	0.42	13	Broad	0.51	34	Broad	0.19
14	Broad	0.32	35	Broad	0.16	14	Broad	0.19	35	Broad	0.20
15	Broad	0.46	36	Broad	0.10	15	Broad	0.79	36	Broad	0.34
16	Broad	0.01	37	Broad	0.11	16	Broad	0.52	37	Broad	0.19
17	Broad	0.05	38	Broad	0.37	17	Broad	0.21	38	Broad	0.19
18	Broad	0.11	39	Broad	0.50	18	Broad	0.19	39	Broad	0.52
			40	Broad	0.16				40	Broad	0.20

Table 79. Midpoint Initialization Verification Data for the Face Table

Midpoint Initialization Values						Midpoint Convergence Values					
ID	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**
1	Broad	0.50	19	Broad	0.50	1	Broad	0.23	19	Broad	0.23
2	Broad	0.50	20	Broad	0.50	2	Broad	0.23	20	Broad	0.23
3	Broad	0.50	21	Broad	0.50	3	Broad	0.52	21	Broad	0.79
6	Broad	0.50	22	Broad	0.50	6	Broad	0.23	22	Broad	0.52
8	Broad	0.50	23	Broad	0.50	8	Broad	0.23	23	Broad	0.52
9	Broad	0.50	24	Broad	0.50	9	Broad	0.23	24	Broad	0.23
10	Broad	0.50	31	Broad	0.50	10	Broad	0.47	31	Broad	0.52
11	Broad	0.50	32	Broad	0.50	11	Broad	0.23	32	Broad	0.23
12	Broad	0.50	33	Broad	0.50	12	Broad	0.53	33	Broad	0.23
13	Broad	0.50	34	Broad	0.50	13	Broad	0.52	34	Broad	0.23
14	Broad	0.50	35	Broad	0.50	14	Broad	0.23	35	Broad	0.23
15	Broad	0.50	36	Broad	0.50	15	Broad	0.79	36	Broad	0.46
16	Broad	0.50	37	Broad	0.50	16	Broad	0.52	37	Broad	0.23
17	Broad	0.50	38	Broad	0.50	17	Broad	0.23	38	Broad	0.23
18	Broad	0.50	39	Broad	0.50	18	Broad	0.23	39	Broad	0.52
			40	Broad	0.50				40	Broad	0.23

Table 80. New Random Initialization Verification Data for the Face Table

New Random Proportional Initialization						New Random Proportional Convergence					
ID	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**	**ID**	**Face**	**Wt**
1	Broad	0.36	19	Broad	0.09	1	Broad	0.20	19	Broad	0.19
2	Broad	0.48	20	Broad	0.30	2	Broad	0.20	20	Broad	0.20
3	Broad	0.28	21	Broad	0.38	3	Broad	0.52	21	Broad	0.80
6	Broad	0.08	22	Broad	0.34	6	Broad	0.20	22	Broad	0.52
8	Broad	0.22	23	Broad	0.12	8	Broad	0.20	23	Broad	0.52
9	Broad	0.46	24	Broad	0.47	9	Broad	0.20	24	Broad	0.19
10	Broad	0.01	31	Broad	0.32	10	Broad	0.19	31	Broad	0.52
11	Broad	0.29	32	Broad	0.18	11	Broad	0.19	32	Broad	0.20
12	Broad	0.29	33	Broad	0.08	12	Broad	0.51	33	Broad	0.20
13	Broad	0.05	34	Broad	0.31	13	Broad	0.51	34	Broad	0.51
14	Broad	0.35	35	Broad	0.06	14	Broad	0.19	35	Broad	0.24
15	Broad	0.37	36	Broad	0.47	15	Broad	0.53	36	Broad	0.51
16	Broad	0.45	37	Broad	0.06	16	Broad	0.77	37	Broad	0.20
17	Broad	0.37	38	Broad	0.08	17	Broad	0.19	38	Broad	0.20
18	Broad	0.48	39	Broad	0.13	18	Broad	0.52	39	Broad	0.19
			40	Broad	0.50				40	Broad	0.20

APPENDIX J

STORED PROCEDURE TO INITIALIZE RANDOM WEIGHTS

Appendix J contains the SQL Server stored procedure used to initialize the Color and Face table values randomly. The procedure is executed in the SQL Server Enterprise Manager window prior to executing queries using the system.

```
CREATE  PROCEDURE [initialize_weights]
as

declare @id as int, @color as varchar(20)
declare @face as varchar(20)

DECLARE color_weight CURSOR
for
select ID,color from color
 OPEN color_weight

FETCH NEXT FROM color_weight
into @ID,@color

WHILE @@FETCH_STATUS = 0
BEGIN
     print 'ID'
     print @id
     print 'color'
     print @color
     update color
     set weight=rand()
     where ID=@id and color=@color
     FETCH NEXT FROM color_weight
     into @ID,@color
END

CLOSE color_weight
DEALLOCATE color_weight
```

```
DECLARE face_weight CURSOR
for
select ID,face from face
 OPEN face_weight

FETCH NEXT FROM face_weight
into @ID, @face

WHILE @@FETCH_STATUS = 0
BEGIN
     print 'ID+Face'
     print @id
     print 'Face'
     print @face
     update Face
     set weight=rand()
     where ID=@id and Face=@face

     FETCH NEXT FROM face_weight
     into @ID,@face
END

CLOSE face_weight
DEALLOCATE face_weight

GO
```

APPENDIX K

STORED PROCEDURE TO INITIALIZE RANDOM PROPORTIONAL WEIGHTS

Appendix J contains the SQL Server stored procedure used to initialize the Color and Face table values for the random proportional method. The procedure is executed in the SQL Server Enterprise Manager window prior to executing queries using the system.

```
CREATE  PROCEDURE [initialize_other]
as

declare @id as int,@color as varchar(20)
declare @face as varchar(20), @counter as int
declare @blue as float, @brown as float, @green as float
declare @average as float, @broad as float
declare @narrow as float

DECLARE color_weight CURSOR
For

select ID,color from color
 OPEN color_weight

FETCH NEXT FROM color_weight
into @ID,@color

     set @counter = 2
     set @brown= rand()
     set @blue = (1.0 - @brown) / 2.0
     set @green = (1.0 - @brown) / 2.0

WHILE @@FETCH_STATUS = 0
BEGIN

      IF @counter = 2
      BEGIN
         print 'ID'
         print @id
```

```
        print 'color'
        print @color
        update color
        set weight=@brown
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
    END

    IF @counter = 1
    BEGIN
        print 'ID'
        print @id
        print 'color'
        print @color
        update color
        set weight=@blue
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
    END

    IF @counter = 0
    BEGIN
        print 'ID'
        print @id
        print 'color'
        print @color
        update color
        set weight=@blue
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
    END

     set @counter = @counter - 1
     IF @counter < 0
     BEGIN
     set @counter = 2
     set @brown= rand()
     set @blue = (1.0 - @brown)/2.0
     set @green = (1.0 - @brown)/2.0
     END
END
```

```
CLOSE color_weight
DEALLOCATE color_weight

DECLARE face_weight CURSOR
For

select ID,face from face
 OPEN face_weight

FETCH NEXT FROM face_weight
into @ID,@face

     set @counter = 2
     set @average= rand()
     set @broad = (1.0 - @average) / 2.0
     set @narrow = (1.0 - @average) / 2.0

WHILE @@FETCH_STATUS = 0
BEGIN

      IF @counter = 2
      BEGIN
         print 'ID'
         print @id
         print 'Face'
         print @face
         update Face
         set weight=@average
         where ID=@id and Face=@face
         FETCH NEXT FROM face_weight
         into @ID,@face
     END

     IF @counter = 1
     BEGIN
         print 'ID'
         print @id
         print 'Face'
         print @face
         update Face
         set weight=@narrow
         where ID=@id and Face=@face
         FETCH NEXT FROM face_weight
```

```
        into @ID,@face
    END


     IF @counter = 0
     BEGIN
        print 'ID'
        print @id
        print 'Face'
        print @face
        update Face
        set weight=@broad
        where ID=@id and Face=@face
        FETCH NEXT FROM face_weight
        into @ID,@face
     END

      set @counter = @counter - 1

      IF @counter < 0
      BEGIN
      set @counter = 2
      set @average= rand()
      set @broad = (1.0 - @average)/2.0
      set @narrow = (1.0 - @average)/2.0
      END
END
CLOSE face_weight
DEALLOCATE face_weight
GO
```

APPENDIX L

STORED PROCEDURE TO INITIALIZE MIDPOINT WEIGHTS

Appendix L contains the SQL Server stored procedure used to initialize the Color and Face table values to the midpoint value. The procedure is executed in the SQL Server Enterprise Manager window prior to executing queries using the system.

```
CREATE  PROCEDURE [initialize_mids]
as

declare @id as int,@color as varchar(20)
declare @face as varchar(20)

DECLARE color_weight CURSOR
For

select ID,color from color
 OPEN color_weight

FETCH NEXT FROM color_weight
into @ID,@color

WHILE @@FETCH_STATUS = 0
BEGIN
     print 'ID'
     print @id
     print 'color'
     print @color
     update color
     set weight=0.50
     where ID=@id and color=@color
     FETCH NEXT FROM color_weight
     into @ID,@color
END

CLOSE color_weight
DEALLOCATE color_weight
```

```
DECLARE face_weight CURSOR
For

select ID,face from face
 OPEN face_weight

FETCH NEXT FROM face_weight
into @ID,@face

WHILE @@FETCH_STATUS = 0
BEGIN
     print 'ID+Face'
     print @id
     print 'Face'
     print @face
     update Face
     set weight=0.50
     where ID=@id and Face=@face
     FETCH NEXT FROM face_weight
     into @ID,@face
END

CLOSE face_weight
DEALLOCATE face_weight
GO
```

APPENDIX M

STORED PROCEDURE TO INITIALIZE NEW RANDOM PROPORTIONAL WEIGHTS

Appendix J contains the SQL Server stored procedure used to initialize the Color and Face table values for the new random proportional method. The procedure is executed in the SQL Server Enterprise Manager window prior to executing queries using the system.

```
CREATE  PROCEDURE [initialize_other_random]
as

declare @id as int,@color as varchar(20),@face as varchar(20)
declare @vala as float, @valb as float, @counter as int, @looper as int
declare @average as float, @broad as float, @narrow as float

DECLARE color_weight CURSOR
for
select ID,color from color
 OPEN color_weight

FETCH NEXT FROM color_weight
into @ID,@color

    set @looper = 0
    set @counter = 2
    set @vala= rand()
    set @valb= ( 1.0 - @vala) / 2.0

WHILE @@FETCH_STATUS = 0
BEGIN

IF @looper = 0
BEGIN
WHILE @@FETCH_STATUS = 0
BEGIN

     IF @counter = 2
     BEGIN
          print 'ID'
           print @id
          print 'color'
          print @color
```

```
        update color
        set weight=@vala
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
    END

    IF @counter = 1
    BEGIN
        print 'ID'
        print @id
        print 'color'
        print @color
        update color
        set weight= @valb
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
     END

   IF @counter = 0
    BEGIN
        print 'ID'
        print @id
        print 'color'
        print @color
        update color
        set weight=@valb
        where ID=@id and color=@color
        FETCH NEXT FROM color_weight
        into @ID,@color
     END

     set @counter = @counter - 1
    IF @counter < 0
    BEGIN
                set @counter = 2
        set @vala= rand()
        set @valb= (1.0 - @vala) / 2.0
        set @looper = @looper + 1

    END
END
END
IF @looper = 1
BEGIN
WHILE @@FETCH_STATUS = 0
BEGIN

    IF @counter = 2
    BEGIN
        print 'ID'
```

```
            print @id
            print 'color'
            print @color
            update color
            set weight=@valb
            where ID=@id and color=@color
            FETCH NEXT FROM color_weight
            into @ID,@color
        END

        IF @counter = 1
        BEGIN
            print 'ID'
            print @id
            print 'color'
            print @color
            update color
            set weight= @vala
            where ID=@id and color=@color
            FETCH NEXT FROM color_weight
            into @ID,@color
         END

       IF @counter = 0
        BEGIN
            print 'ID'
            print @id
            print 'color'
            print @color
            update color
            set weight=@valb
            where ID=@id and color=@color
            FETCH NEXT FROM color_weight
            into @ID,@color
         END

         set @counter = @counter - 1
        IF @counter < 0
        BEGIN
                    set @counter = 2
            set @valb= rand()
            set @vala= (1.0 - @valb) / 2.0
            set @looper = @looper + 1

        END
    END
    END

    IF @looper = 2
    BEGIN
    WHILE @@FETCH_STATUS = 0
    BEGIN
```

```
 IF @counter = 2
 BEGIN
       print 'ID'
        print @id
       print 'color'
       print @color
       update color
       set weight=@valb
       where ID=@id and color=@color
       FETCH NEXT FROM color_weight
       into @ID,@color
END

IF @counter = 1
BEGIN
       print 'ID'
       print @id
       print 'color'
       print @color
       update color
       set weight= @valb
       where ID=@id and color=@color
       FETCH NEXT FROM color_weight
       into @ID,@color
 END

IF @counter = 0
 BEGIN
       print 'ID'
       print @id
       print 'color'
       print @color
       update color
       set weight=@vala
       where ID=@id and color=@color
       FETCH NEXT FROM color_weight
       into @ID,@color
  END

  set @counter = @counter - 1
 IF @counter < 0
 BEGIN
                set @counter = 2
       set @vala= rand()
       set @valb= (1.0 - @vala) / 2.0
       set @looper = @looper + 1

 END
END
END
IF @looper = 3
```

```
BEGIN
        Set @looper = 0
        set @vala= rand()
        set @valb= (1.0 - @vala) / 2.0
END
END

CLOSE color_weight
DEALLOCATE color_weight
GO
```

APPENDIX N

STORED PROCEDURE TO UPDATE WEIGHTS

Appendix N contains the SQL Server stored procedure used to update the image weight after the query evaluation was returned with the user feedback. The procedure is executed in the SQL Server Enterprise Manager window prior to executing queries using the system.

```
CREATE PROCEDURE [Update_Data]
@ID as int,
@color as  varchar(50),
@more_less as varchar(50),
@modifier as varchar(50)

AS

SET ANSI_NULLS ON
declare @threshold as float
declare @existing_weight as float
declare @new_weight as float

 IF @color = 'Blue' or
    @color= 'Green' or
    @color= 'Brown'
 BEGIN
     select @threshold = threshold from Range where
     modifier=@modifier

     select @existing_weight=Weight from Color where ID=@ID
     and Color=@color

     set @new_weight=@existing_weight

  IF @more_less = 'meets'
  BEGIN

     IF @existing_weight < @threshold
     BEGIN
```

```
          set @new_weight = @existing_weight + .02
     END

     IF @existing_weight > @threshold
     BEGIN
          set @new_weight = @existing_weight - .02
     END
END

ELSE
BEGIN
   IF @more_less = 'more'
   BEGIN
      IF @modifier = 'very'
      BEGIN

     IF @existing_weight > @threshold
     BEGIN
        SET @new_weight = @existing_weight - .02
     END

     IF @existing_weight < @threshold
     BEGIN
        SET @new_weight = @existing_weight + .02
     END
   END
   ELSE
   BEGIN
               SET @new_weight = @existing_weight + .02
    END
END
ELSE
BEGIN
   IF @modifier='slightly'
   BEGIN
      IF @existing_weight > @threshold
      BEGIN
        SET @new_weight = @existing_weight - .02
      END

      IF @existing_weight < @threshold
      BEGIN
         SET @new_weight = @existing_weight + .02
      END
```

```
   END
   ELSE
   BEGIN
      SET @new_weight = @existing_weight - .02
   END
 END
END
     UPDATE Color SET weight=@new_weight where ID=@ID and
     color=@color

END
IF @color = 'Broad' or
    @color = 'Average' or
    @color='Narrow'
BEGIN
   SELECT @threshold = threshold from Range where
   modifier = @modifier

   SELECT @existing_weight = Weight from Face where
   ID = @ID and Face=@color

   SET @new_weight=@existing_weight

   IF @more_less = 'meets'
   BEGIN
     IF @existing_weight < @threshold
     BEGIN
          SET @new_weight = @existing_weight + .02
     END

     IF @existing_weight > @threshold
     BEGIN
          SET @new_weight = @existing_weight - .02
     END
END

ELSE
BEGIN
   IF @more_less = 'more'
   BEGIN
      IF @modifier = 'very'
      BEGIN
      IF @existing_weight > @threshold
         BEGIN
```

```
         SET @new_weight = @existing_weight - .02
      END

     IF @existing_weight < @threshold
      BEGIN
         SET @new_weight = @existing_weight + .02
      END
      END

      ELSE
      BEGIN
      SET @new_weight = @existing_weight + .02
      END
   END

   ELSE
   BEGIN
      IF @modifier='slightly'
      BEGIN
      IF @existing_weight > @threshold
      BEGIN
         SET @new_weight = @existing_weight - .02
      END

      IF @existing_weight < @threshold
      BEGIN
         SET @new_weight = @existing_weight + .02
      END
      END
      ELSE
      BEGIN
         SET @new_weight = @existing_weight - .02
      END
   END
END

UPDATE Face SET weight = @new_weight where ID = @ID and
Face = @color

END
GO
```

APPENDIX O

FUZZY DATABASE PROTOTYPE CODE

Appendix O contains all the source code used to run and create the Graphical User Interface for the prototype. The source code contained below is the code executed for this evaluation. Debra Duke updated portions of the code original code prior to this evaluation. The original code is referenced in [17].

```
'Database Research Summer 2004
'Karen Joy and Smita Dattatri
'This form invokes the stored procedure "Fetch Data" which
interprets the Fuzzy query and
'returns the set of rows to be displayed on the User
Interface.This form also invokes a
'stored procedure called Update_data which updates the
database depending on the feedback
'given by the user.

'Spring, 2005
'Debra Duke
'Updated to add multiple page viewing functionality.

Imports System.IO
Imports System.Data.SqlClient


Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
```

```
        InitializeComponent()

        'Add any initialization after the
InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal
disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the
Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Panel1 As System.Windows.Forms.Panel
    Friend WithEvents Button2 As System.Windows.Forms.Button
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Button3 As System.Windows.Forms.Button
    Friend WithEvents Panel2 As System.Windows.Forms.Panel
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.Panel1 = New System.Windows.Forms.Panel
        Me.Button3 = New System.Windows.Forms.Button
        Me.Button1 = New System.Windows.Forms.Button
        Me.Button2 = New System.Windows.Forms.Button
        Me.Label1 = New System.Windows.Forms.Label
        Me.Panel2 = New System.Windows.Forms.Panel
        Me.Panel1.SuspendLayout()
        Me.SuspendLayout()
        '
        'Panel1
        '
```

```
        Me.Panel1.Controls.Add(Me.Button3)
        Me.Panel1.Controls.Add(Me.Button1)
        Me.Panel1.Controls.Add(Me.Button2)
        Me.Panel1.Dock =
System.Windows.Forms.DockStyle.Bottom
        Me.Panel1.Location = New System.Drawing.Point(0,
637)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New System.Drawing.Size(872, 32)
        Me.Panel1.TabIndex = 2
        '
        'Button3
        '
        Me.Button3.Anchor =
System.Windows.Forms.AnchorStyles.Bottom
        Me.Button3.Location = New System.Drawing.Point(216,
0)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New System.Drawing.Size(104, 24)
        Me.Button3.TabIndex = 4
        Me.Button3.Text = "More"
        '
        'Button1
        '
        Me.Button1.Anchor =
System.Windows.Forms.AnchorStyles.Bottom
        Me.Button1.BackColor =
System.Drawing.Color.LightGray
        Me.Button1.Font = New System.Drawing.Font("Verdana",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button1.Location = New System.Drawing.Point(384,
0)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(104, 24)
        Me.Button1.TabIndex = 1
        Me.Button1.Text = "Update"
        '
        'Button2
        '
        Me.Button2.Anchor =
System.Windows.Forms.AnchorStyles.Bottom
        Me.Button2.BackColor =
System.Drawing.Color.LightGray
```

```
        Me.Button2.Font = New System.Drawing.Font("Verdana",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button2.Location = New System.Drawing.Point(552,
0)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New System.Drawing.Size(104, 24)
        Me.Button2.TabIndex = 3
        Me.Button2.Text = "Cancel"
        '
        'Label1
        '
        Me.Label1.Font = New System.Drawing.Font("Verdana",
12.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.Location = New System.Drawing.Point(280,
0)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(552, 24)
        Me.Label1.TabIndex = 4
        Me.Label1.Text = "Label1"
        '
        'Panel2
        '
        Me.Panel2.Location = New System.Drawing.Point(0, 32)
        Me.Panel2.Name = "Panel2"
        Me.Panel2.Size = New System.Drawing.Size(864, 600)
        Me.Panel2.TabIndex = 5
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5,
13)
        Me.ClientSize = New System.Drawing.Size(872, 669)
        Me.Controls.Add(Me.Panel2)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.Panel1)
        Me.Name = "Form1"
        Me.Text = "Query Results"
        Me.WindowState =
System.Windows.Forms.FormWindowState.Maximized
        Me.Panel1.ResumeLayout(False)
        Me.ResumeLayout(False)
```

```
    End Sub

#End Region
    Dim Query As String
    Dim string2 As String
    Dim modifier As String
    Dim color_face As String
    Dim ds As New DataSet
    Dim al As New ArrayList
    Dim nextPix As Integer
    Dim remainingPix As Integer

    'Connection String used to connect to the database.
    'CSC-LAB=Name Of the Computer
    'DBResearch=Name of the database

'2/16/05:  Name of the Computer changed to reflect working
on my local machine (DEBS), Debra
    Dim con As New System.Data.SqlClient.SqlConnection("data
source=DEBS;initial catalog=DBResearch;user
id=dbresearch;password=dbresearch")


    'Event handling procedure for the 'Update' button

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim I As Integer
        Dim more_less As String
        'con.Open()
        'This block of code checks which check boxes were
clicked by the user  and for which
        'person in order to update the corresponding rows in
the database.For each person the
        'stored procedure Update_Data is invoked.
        For I = 0 To al.Count - 1
            If (al(I).meets_criteria.Checked() = True) Then
                more_less = "meets"
            ElseIf (al(I).more.Checked() = True) Then
                more_less = "more"
            ElseIf (al(I).less.Checked() = True) Then
                more_less = "less"
            End If
```

```
            Dim daptr = New SqlDataAdapter
            daptr.SelectCommand = New SqlCommand
            daptr.SelectCommand.CommandType =
CommandType.StoredProcedure
            daptr.SelectCommand.CommandText =
"dbo.Update_Data"
            daptr.SelectCommand.Connection = con
            daptr.SelectCommand.Parameters.Add("@ID",
al(I).ID)
            daptr.SelectCommand.Parameters.Add("@color",
color_face)
            daptr.SelectCommand.Parameters.Add("@more_less",
more_less)
            daptr.SelectCommand.Parameters.Add("@modifier",
modifier)
            con.Open()
            Try
                daptr.SelectCommand.ExecuteNonQuery()
            Catch ex As Exception
                MessageBox.Show("Failed to execute query")
            End Try

            con.Close()

        Next
        Dim message As Message
        message = New Message
        Me.Dispose()
        message.Show()

    End Sub


    'This function is invoked when the form is loaded.The
stored procedure "Fetch_Data" is
    'invoked and the fuzzy modifiers are passed as
parameters to this stored procedure.
    'This stored procedure interprets the fuzzy modifiers
and returns the result set that
    ' matches the criteria.

    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim I As Integer
```

```
        Dim Count As Integer
        Dim cmd As New
System.Data.SqlClient.SqlCommand("Fetch_Data", con)

        cmd.CommandType = CommandType.StoredProcedure

        cmd.Parameters.Add(New SqlParameter("@query",
SqlDbType.VarChar, 100))
        cmd.Parameters("@query").Value = Query
        cmd.Parameters.Add(New SqlParameter("@s1",
SqlDbType.VarChar, 100))
        cmd.Parameters("@s1").Value = modifier
        cmd.Parameters.Add(New SqlParameter("@s2",
SqlDbType.VarChar, 100))
        cmd.Parameters("@s2").Value = color_face
        Try
            Dim da As New
System.Data.SqlClient.SqlDataAdapter(cmd)
            da.Fill(ds)
        Catch ex As Exception
            MessageBox.Show(ex.ToString)
        End Try

        'displays the number of rows returned as a result of
query (use for debugging)
        'MessageBox.Show(ds.Tables(0).Rows().Count)

        If (ds.Tables(0).Rows().Count) <> 0 Then
            Count = ds.Tables(0).Compute("COUNT(ID)", "")

            'The following block of code creates a new
object of ImageSet for each
            'record in the which result set which contains
the image and the Check Boxes
            'associated with it.  It also reads the image in
binary format from the database
            'and displays it appropriately.

            For I = 0 To Count - 1
                Dim S As New ImageSet(color_face)
                Dim bits As Byte() =
CType(ds.Tables(0).Rows(I).Item(2), Byte())
                Dim memorybits As New MemoryStream(bits)
                Dim bitmap As New Bitmap(memorybits)
```

```
                S.picture.Image = bitmap
                al.Add(S)
                al(I).ID = ds.Tables(0).Rows(I).Item(0)
            Next I

            'variable indicating the first picture to be
displayed on a page
            nextPix = 0
            'variable indicating the remaining pictures to
be displayed
            remainingPix = al.Count

            placeComponents(nextPix)

            updateButtons()

            'Changes the labels on the Panel depending on
the query.
            If color_face = "Green" Or color_face = "Blue"
Or color_face = "Brown" Or color_face = "green" Or
color_face = "blue" Or color_face = "brown" Then
                Label1.Text() = "People with " + modifier +
" " + color_face + " Eyes"
            Else
                Label1.Text() = "People With " + modifier +
" " + color_face + " Faces"
            End If
        Else
            MessageBox.Show("No images match your criteria")
        End If
    End Sub


    'The following block of code places the components on
the Group Box and
    'then places the group box on the panel to be displayed.
    Private Sub placeComponents(ByVal Position1 As Integer)

        Dim X As Integer, Y As Integer
        Y = 8
        Dim I As Integer

        'limit the display to 6 images
```

```
        Dim lastPix As Integer

        'determing the last picture for the current page
        If remainingPix <= 6 Then
            lastPix = Position1 + remainingPix - 1
        Else : lastPix = Position1 + 5
        End If

        For I = Position1 To lastPix
            Dim G As New GroupBox
            G.Location() = New Point(8 + X, Y)
            G.Size() = New Size(268, 295) 'was (288,350)

            al(I).picture.Location = New Point(50, 24)
            al(I).picture.size = New Size(200, 200) 'was
(224,120)

            al(I).meets_criteria.location = New Point(32,
225) 'was (32, 250)
            al(I).meets_criteria.size = New Size(176, 24)

            al(I).more.size = New Size(176, 24)
            al(I).more.location = New Point(32, 250)  'was
(32, 275)

            al(I).less.size = New Size(176, 24)
            al(I).less.location = New Point(32, 275)  'was
(32,300)

            G.Controls.Add(al(I).picture)
            G.Controls.Add(al(I).meets_criteria)
            G.Controls.Add(al(I).more)
            G.Controls.Add(al(I).less)
            Panel2.Controls.Add(G)

            'Allows to display multiple rows of records.
            X = 300 + X
            If ((I Mod 6) = 2) Then
                Y = 295 + 5 + Y
                X = 0
            End If
        Next

        'update the number of picutes left to be displayed
```

```
        If remainingPix > 6 Then
            remainingPix = remainingPix - 6
        Else
            remainingPix = 0
        End If

        'update first picture on next page
        nextPix = nextPix + 6
    End Sub


    'set up buttons depending on number of images in query
result
    Private Sub updateButtons()
        If al.Count <= 6 Or remainingPix = 0 Then
            Me.Button3.Enabled = False
            Me.Button1.Enabled = True
        Else
            Me.Button3.Enabled = True
            Me.Button1.Enabled = False
        End If
    End Sub


    Public Sub initialize(ByVal s1 As String, ByVal s2 As
String, ByVal s3 As String)
        Dim strarry() As String
        Query = s1
        modifier = s2
        color_face = s3
    End Sub


    'Event handling for 'Cancel' button
    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        con.Close()
        Me.Dispose()
    End Sub


    'Event handling for 'More' button
    Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click
```

```
        'clear existing pictures
        Panel2.Controls.Clear()

        'build and display next page
        placeComponents(nextPix)
        Panel2.Show()

        updateButtons()
    End Sub

    '-------------------------------------------------------
---------
    'The class ImagesSet is a user defined class which has
the image
    'and the check boxes as its components.

    Public Class ImageSet
        Public picture As PictureBox
        Public meets_criteria As RadioButton
        Public more As RadioButton
        Public Less As RadioButton
        Public ID As Integer


        Public Sub New(ByVal color_face As String)
            picture = New PictureBox
            meets_criteria = New RadioButton
            more = New RadioButton
            Less = New RadioButton
            meets_criteria.Text = "Meets The Criteria"
            meets_criteria.Checked = True
            'Changes the labels on the Check Boxes depending
on the query.
            If color_face = "Green" Or color_face = "green"
Then
                more.Text = "More Green"
                Less.Text = "Less Green"
            End If
            If color_face = "Brown" Or color_face = "brown"
Then
                more.Text = "More Brown"
                Less.Text = "Less Brown"
            End If
```

```
            If color_face = "Blue" Or color_face = "blue"
Then
                more.Text = "More Blue"
                Less.Text = "Less Blue"
            End If
            If color_face = "Broad" Or color_face = "broad"
Then
                more.Text = "More Broad"
                Less.Text = "Less Broad"
            End If
            If color_face = "Narrow" Or color_face =
"narrow" Then
                more.Text = "More Narrow"
                Less.Text = "Less Narrow"
            End If
            ID = 0

        End Sub

    End Class
End Class

Public Class Form2
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the
InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal
disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
```

```
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the
Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents ImageList1 As
System.Windows.Forms.ImageList
    Friend WithEvents ImageList2 As
System.Windows.Forms.ImageList
    Friend WithEvents ImageList3 As
System.Windows.Forms.ImageList
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Me.ImageList1 = New
System.Windows.Forms.ImageList(Me.components)
        Me.ImageList2 = New
System.Windows.Forms.ImageList(Me.components)
        Me.ImageList3 = New
System.Windows.Forms.ImageList(Me.components)
        '
        'ImageList1
        '
        Me.ImageList1.ImageSize = New
System.Drawing.Size(16, 16)
        Me.ImageList1.TransparentColor =
System.Drawing.Color.Transparent
        '
        'ImageList2
        '
        Me.ImageList2.ImageSize = New
System.Drawing.Size(16, 16)
        Me.ImageList2.TransparentColor =
System.Drawing.Color.Transparent
        '
        'ImageList3
        '
```

```
        Me.ImageList3.ImageSize = New
System.Drawing.Size(16, 16)
        Me.ImageList3.TransparentColor =
System.Drawing.Color.Transparent
        '
        'Form2
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5,
13)
        Me.ClientSize = New System.Drawing.Size(292, 266)
        Me.Name = "Form2"
        Me.Text = "Form2"

    End Sub

#End Region

    Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub Panel1_Paint(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.PaintEventArgs)

    End Sub
End Class

'Database Research Summer 2004
'Karen Joy and Smita Dattatri
'This form informs the user that the update opearation was
successful and asks the user
' whether he/she would like to run another query.
Public Class Message
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()
```

```
        'Add any initialization after the
InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal
disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the
Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents Button2 As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label
        Me.Label2 = New System.Windows.Forms.Label
        Me.Button1 = New System.Windows.Forms.Button
        Me.Button2 = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.Font = New System.Drawing.Font("Verdana",
9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.Location = New System.Drawing.Point(136,
8)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(200, 24)
```

```
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Updated Successfully!!"
        '
        'Label2
        '
        Me.Label2.Font = New System.Drawing.Font("Verdana",
12.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label2.Location = New System.Drawing.Point(80,
48)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(320, 24)
        Me.Label2.TabIndex = 1
        Me.Label2.Text = "Would you like to run another
query?"
        '
        'Button1
        '
        Me.Button1.Font = New System.Drawing.Font("Verdana",
9.75!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button1.Location = New System.Drawing.Point(80,
96)
        Me.Button1.Name = "Button1"
        Me.Button1.TabIndex = 2
        Me.Button1.Text = "Yes"
        '
        'Button2
        '
        Me.Button2.Font = New System.Drawing.Font("Verdana",
9.75!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button2.Location = New System.Drawing.Point(256,
96)
        Me.Button2.Name = "Button2"
        Me.Button2.TabIndex = 3
        Me.Button2.Text = "No"
        '
        'Message
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5,
13)
        Me.ClientSize = New System.Drawing.Size(424, 149)
        Me.Controls.Add(Me.Button2)
```

```
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Location = New System.Drawing.Point(5000, 7000)
        Me.Name = "Message"
        Me.Text = "Form1"
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub Label1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Label1.Click

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim f As QueryForm
        f = New QueryForm
        f.Show()
        Me.Dispose()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        Me.Dispose()
    End Sub

    Private Sub Message_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub
End Class

'Database Research Summer 2004
'Karen Joy and Smita Dattatri
'This form reads the file which contains the Fuzzy Query and
displays that on the screen
'giving the user a chance to change the query.The user can
then click on the "Submit" button which submits the
'fuzzy query to the intermediate layer(stored procedures)in
order to be processed.
```

```
Imports System.io
Public Class QueryForm
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the
InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal
disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the
Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents TextBox1 As
System.Windows.Forms.TextBox
    Friend WithEvents TextBox2 As
System.Windows.Forms.TextBox
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents Button2 As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.TextBox1 = New System.Windows.Forms.TextBox
```

```
        Me.TextBox2 = New System.Windows.Forms.TextBox
        Me.Button1 = New System.Windows.Forms.Button
        Me.Button2 = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'TextBox1
        '
        Me.TextBox1.Font = New
System.Drawing.Font("Verdana", 9.0!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox1.Location = New System.Drawing.Point(40,
40)
        Me.TextBox1.Multiline = True
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.ScrollBars =
System.Windows.Forms.ScrollBars.Both
        Me.TextBox1.Size = New System.Drawing.Size(512, 50)
        Me.TextBox1.TabIndex = 0
        Me.TextBox1.Text = "TextBox1"
        '
        'TextBox2
        '
        Me.TextBox2.Font = New
System.Drawing.Font("Verdana", 9.0!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox2.Location = New System.Drawing.Point(40,
136)
        Me.TextBox2.Multiline = True
        Me.TextBox2.Name = "TextBox2"
        Me.TextBox2.ScrollBars =
System.Windows.Forms.ScrollBars.Both
        Me.TextBox2.Size = New System.Drawing.Size(512, 50)
        Me.TextBox2.TabIndex = 1
        Me.TextBox2.Text = "TextBox2"
        '
        'Button1
        '
        Me.Button1.Font = New System.Drawing.Font("Verdana",
9.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button1.Location = New System.Drawing.Point(104,
256)
```

```
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(136, 40)
        Me.Button1.TabIndex = 1
        Me.Button1.Text = "Submit Query"
        '
        'Button2
        '
        Me.Button2.Font = New System.Drawing.Font("Verdana",
9.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Button2.Location = New System.Drawing.Point(304,
256)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New System.Drawing.Size(136, 40)
        Me.Button2.TabIndex = 3
        Me.Button2.Text = "Cancel"
        '
        'QueryForm
        '
        Me.AcceptButton = Me.Button1
        Me.AutoScaleBaseSize = New System.Drawing.Size(5,
13)
        Me.BackColor = System.Drawing.SystemColors.Control
        Me.ClientSize = New System.Drawing.Size(592, 366)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.TextBox2)
        Me.Controls.Add(Me.TextBox1)
        Me.Name = "QueryForm"
        Me.Text = "Query"
        Me.ResumeLayout(False)

    End Sub

#End Region
    Dim modifier As String
    Dim color_face As String
    Dim strarry(20) As String
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        'Reads the file which contains the Fuzzy Query
        Dim myfileStream As New
IO.FileStream("C:\input.txt", IO.FileMode.Open,
IO.FileAccess.Read)
```

```
        Dim myreader As New IO.StreamReader(myfileStream)

        Try
            TextBox1.Clear()
            Dim line2 As New String("")
            Do
                line2 = line2 + myreader.ReadLine + vbCrLf

            Loop Until myreader.Peek = -1
            TextBox1.AppendText(line2)
        Catch ex As Exception
            TextBox1.AppendText("File is empty")
        Finally
            myreader.Close()
        End Try
        Dim myfileStream1 As New
IO.FileStream("C:\input1.txt", IO.FileMode.Open,
IO.FileAccess.Read)
        Dim myreader1 As New IO.StreamReader(myfileStream1)
        Try
            TextBox2.Clear()
            Dim line1 As New String("")
            Dim i As Integer
            i = 0
            Do
                'line1 = line1 + myreader1.ReadLine + vbCrLf
                strarry(i) = New
String(myreader1.ReadLine())
                i = i + 1
            Loop Until myreader1.Peek = -1
            TextBox2.AppendText(strarry(0) + vbCrLf +
strarry(1))
        Catch ex As Exception
            ' TextBox2.AppendText("File is empty")
            TextBox2.AppendText(ex.ToString)
        Finally
            myreader1.Close()
        End Try
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim f As Form1
        f = New Form1
```

```
        Dim s1 As String = TextBox2.Lines(0)
        Dim s2 As String = TextBox2.Lines(1)
        f.initialize(TextBox1.Text, s1, s2)
        Me.Hide()
        f.Show()


    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        Me.Dispose()
        Me.Close()


    End Sub
End Class

Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled
through the following
' set of attributes. Change these attribute values to modify
the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>

'The following GUID is for the ID of the typelib if this
project is exposed to COM
<Assembly: Guid("A9C1C1EF-DF0B-4521-BFC1-FB36E5E37A25")>

' Version information for an assembly consists of the
following four values:
```

```
'
'      Major Version
'      Minor Version
'      Build Number
'      Revision
'
' You can specify all the values or you can default the
Build and Revision Numbers
' by using the '*' as shown below:

<Assembly: AssemblyVersion("1.0.*")>
```