FUZZY GENETIC ALGORITHM FOR TELEMEDICINE

A Dissertation submitted
in partial fulfillment of the requirement for the award of the degree of
Master of Engineering
in Controls & Instrumentations

To
Faculty of Technology, University of Delhi

Under the guidance of
PROF. PARMOD KUMAR
Session: 2002-2005

Submitted by
ROMA RAINA
08/C&I/2002
(University Roll No. 4161)

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI COLLEGE OF ENGINEERING
UNIVERSITY OF DELHI
DELHI-110042

CERTIFICATE

Fuzzy Genetic Algorithm for Telemedicine is a dissertation work submitted in partial fulfillment of the requirement for the award of Degree of master of Engineering in Control and Instrumentation of University of Delhi, By Roma Raina. She has worked for this dissertation under my guidelines and the work has not been submitted else where for a degree.

Prof. Pramod Kumar

Head
Department of Electrical Engineering
Delhi College of Engineering
Delhi

ACKNOWLEDGEMENT

It has been pleasure for me to take the major project on the new and recent area (Fuzzy Genetic Algorithm for telemedicine) as a part of M.E (Controls and Instrumentation) curriculum under the supervision and guidance of Prof. Pramod Kumar at Delhi College of Engineering, Delhi.

I take the opportunity to thank DCE authorities and the faculty members of the electrical department of the college for extending all the help and assistance.

I would like to thank Prof. Pramod Kumar (H.O.D, Electrical Engg. Dept.) who was my project guide and the source of inspiration throughout the course of the project. I express my sincere gratitude towards him for his valuable and timely advice and guidance without which the project would not have been successful.

I would also like to thank Prof. Swarn Ahuja (H.O.D, ECE Dept, of Institute of Technology and Management, Gurgaon), The college Director Prof.H.P.Garg, Management of I.T.M, Mr. Ashish Raina, and Mr. Lalit Ahuja for their inspiration and continual help in this project.

Roma Raina
(University Roll No. 4161)

ABSTRACT

Uncertainties are always present in the medical diagnosis. In this thesis the uncertainty and vagueness in the Blood Pressure monitoring have been handled by processing them with fuzzy- genetic algorithm. The initial population for Genetic Algorithm applied is generated with assumed fuzzy functions. Genetic algorithm can yield global diagnostic by tuning the fuzzy functions, whatever the weights of the symptoms are considered or fuzzy functions assumed. These algorithms are based on certain concepts of biology and represent the coded set of symptoms, called population. These functions are optimized using theory of reproduction, crossover and mutation. The result shows that fuzzy genetic algorithm shows better results than other methods.

A good diagnosis based on the symptoms that a patient display is quite important. The symptom and complaints may be of various degrees and weights. It therefore makes sense to study these problems of diagnosis in the context of fuzzy sets, which can take into account the ambiguity and vagueness. The genetic algorithms are used to optimize the fuzzy functions of global diagnostic.

Chapter I – Introduction to Fuzzy Genetic Algorithm for Telemedicine

Telemedicine is the information exchange between a patient and the doctor and the consultancy provided by the doctor to the patient at remote locations. The patient communicates the symptoms in his own language regarding the problem he is facing to the doctor. The doctor processes this information and extracts the actual information of the symptoms to diagnose the possible disease the patient may have and accordingly he prescribes the medicine to the patient. Thus there is a vagueness and uncertainty involved in the two way communication. The processing of information at doctors end by conventional techniques such as Fourier series, wavelet do not meet the requirements related to uncertainties and vagueness in the raw data provided by the patient. It is therefore proposed to use Fuzzy Logic technology to extract the information from uncertain and vague data. In order to tune the fuzzy functions and get the solution globally, genetic algorithm has been applied to fuzzy logic technology. The integration of fuzzy with genetic has been defined as geno fuzzy hybrid system in the work present in the dissertation.

GenoFuzzy hybrid systems in telemedicine's combine the advantages of fuzzy systems, and genetic algorithms, which provides a good way to adjust the expert's knowledge and automatically generate additional fuzzy rules, to meet certain specifications and reduce design time and costs. This genofuzzy combination can give more accurate results of any disease to which this technique is applied and hence a patient can be cured accordingly by doctors.

Introduction to Telemedicine

Telemedicine has been defined as the use of telecommunications to provide medical information and services. It may be as simple as two health professionals discussing a case over the telephone, or as sophisticated as using satellite technology to broadcast a consultation between providers at facilities in two countries, using videoconferencing equipment. The first is used daily by most health professionals, and the latter is used by the military and some large medical centers. This transfer of medical data may utilize a variety of telecommunications technology, including, telephone lines, ISDN, the Internet, intranets, and satellites.

Types of Technology

Two different kinds of technology make up most of the telemedicine applications in use today. The first, called store and forward, is used for transferring digital images from one location to another. A digital image is taken using a digital camera, ('stored') and then sent ('forwarded') to another location. This is typically used for non-emergent situations, when a diagnosis or consultation may be made in the next 24 - 48 hours and sent back.

The image may be transferred within a building, between two buildings in the same city, or from one location to another anywhere in the world. Teleradiology, the sending of x-rays, CT scans, or MRIs (store-and-forward images) is the most common application of telemedicine in use today. There are hundreds of medical centers, clinics, and individual physicians who use some form of teleradiology. Many radiologists are installing appropriate computer technology in their homes, so they can have images sent directly to them for diagnosis, instead of making an off-hours trip to a hospital or clinic.

Telepathology is another common use of this technology. Images of pathology slides may be sent from one location to another for diagnostic consultation. Dermatology is also a natural for store and forward technology (although practitioners are increasingly using interactive technology for dermatological exams). Digital images may be taken of skin conditions, and sent to a dermatologist for diagnosis.

Videoconferencing equipment at both locations allows a 'real-time' consultation to take place. The technology has decreased in price and complexity over the past five years, and many programs now use desktop videoconferencing systems. There are many configurations of an interactive consultation, but most typically it is from an urban-to-rural location. It means that the patient does not have to travel to an urban area to see a specialist, and in many cases, provides access to specialty care when none has been available previously. Almost all specialties of medicine have been found to be conducive to this kind of consultation, including psychiatry, internal medicine, rehabilitation, cardiology, pediatrics, obstetrics and gynecology and neurology. There are also many peripheral devices, which can be attached to computers, which can aid in an interactive examination. For instance, an otoscope allows a physician to 'see' inside a patient's ear; a stethoscope allows the consulting physician to hear the patient's heartbeat.

1.2 Advantages of Telemedicine

Imagine this scenario, an elderly man, age 84, lives in a rural community in central Gujarat. He has a severe case of hypertension and his blood pressure and heart rate must be monitored daily. However, the nearest hospital is one and one- half hours away, and transporting him to the center for a 5-minute check- up would be inefficient, both in cost and time. Additionally, because of his old age, traveling that far wears him out and nurses are not justified to travel that far of a distance. Unfortunately, this incident has been reoccurring, especially in underserved areas. Doctors are unable to provide as high a quality of care as in urban areas due to the inadequate number of specialists and a decreased level of technology. To answer many of these problems, telemedicine has been developed.

People living in rural and remote areas struggle to access timely, quality specialty medical care. Residents of these areas have substandard or no access to specialty health cares, primarily because specialist physicians are more likely to be located in areas of concentrated population. Because of innovations in computing and telecommunications technology, many elements of medical practice can be accomplished when the patient and health care provider are geographically separated. Particularly in rural areas, telemedicine offers the potential of both improved access to care and improved quality of care. But in no way does it mean that it is a substitution for health care. It helps in the improvement and renovation of our present health care system

1.3 Features of fuzzy Genetic Algorithm:

- Optimization

- Uncertainty due to vagueness
- Membership function
- If then rules
- Natural Evolution
- Fitness function
- Reproduction
- Crossover
- Mutation

1.4 Objective of work:

Objective of my work is to optimize the blood pressure monitoring by using fuzzy genetic algorithm and find out the range for a BP for any patient so that accordingly Doctor can provide the consultancy at far distance through telemedicine technology..

Following were the steps followed during my project work:

- Making fuzzy form of blood pressure.
- Applying genetic to the string formed by fuzzy representation.
- Applying mamdani to the final genetic chain obtained.
- Defuzzifying in the last to get the final output value.

1.5 Dissection of Dissertation

Following para lists down the chapters covered in this report.

Chapter 2 describes the evolutionary Genetic Algorithms, operation of Genetic algorithm, crossover, mutation, fitness function, efficiency, applications involved.

Chapter 3 contains the designing and developing of fuzzy logic. It describes the details of the techniques of Fuzzy logic system, membership functions, its applications, advantages in the growing field.

Chapter 4 will describes the details of the genofuzzy computing system

Chapter 5 contains the development of fuzzy genetic algorithm and its application in blood pressure monitoring. It gives the detail of the logic used based on Fuzzy implementation , then how the crossover and mutations have been done, and finally calculating the result based on the final genetic string by applying mamdani rules on it.

Chapter 6 describes the results and conclusion of this new technology it also explains the further scope of this tool in various fields.

Chapter II
Genetic Algorithm

Introduction

A **genetic algorithm** (**GA**) is a heuristic used in computer science_to find approximate solutions to combinatorial optimization problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover).

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called *chromosomes*) of candidate solutions (called *individuals*) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

Many Genetic Algorithms have been produced by different researchers, and all of them are very different from each other. They all, however, display the characteristics of the genetic algorithm, which follow these basic steps:

1. Step one it to randomly create a population of individuals.
2. Step two is to evaluate the population to see which individuals will contribute to the next generation.

3. Step three is to alter the new generation of individuals once they have been paired off.
4. The final step is to discard the old population and perform step two on the new population.

Once step three, above, has been completed, the algorithm jumps back to step two. The loop will only stop when one of the individuals has been evaluated and is said to be either very close to the solution, or it has found the solution.

2.2 Operation of a GA

The problem to be solved is represented by a list of parameters which can be used to drive an evaluation procedure, called chromosomes or genomes. Chromosomes are typically represented as simple strings of data and instructions, although a wide variety of other data structures for storing chromosomes have also been tested, with varying degrees of success in different problem domains.

Initially several such parameter lists or chromosomes are generated. This may be totally random, or the programmer may seed the gene pool with "hints" to form an initial pool of possible solutions. This is called the first generation pool.

During each successive generation, each organism (or individual) is evaluated, and a value of goodness or fitness is returned by a fitness function. The pool is sorted, with those having better fitness (representing better solutions to the problem) ranked at the top. Notice that "better" in this context is relative, as initial solutions are all likely to be rather poor.

The next step is to generate a second generation pool of organisms, which is done using any or all of the genetic operators: selection, crossover (or recombination), and mutation. A pair of organisms is selected for breeding. Selection is biased towards elements of the initial generation which have better fitness, though it is usually not so biased that poorer elements have no chance to participate, in order to prevent the solution set from converging too early to a sub-optimal or local solution

Following selection, the crossover (or recombination) operation is performed upon the selected chromosomes. Most genetic algorithms will have a single tweak able probability of crossover (P_c), typically between 0.6 and 1.0, which encodes the probability that two selected organisms will actually breed. A random number between 0 and 1 is generated, and if it falls below the crossover threshold, the organisms are mated; otherwise, they are propagated into the next generation unchanged. Crossover results in two new child chromosomes, which are added to the second generation pool. The chromosomes of the parents are mixed in some way during crossover, typically by simply swapping a portion of the underlying data structure (although other, more complex merging mechanisms have proved useful for certain types of problems.) This process is repeated with different parent organisms until there are an appropriate number of candidate solutions in the second generation pool.

The next step is to mutate the newly created offspring. Typical genetic algorithms have a fixed, very small probability of mutation (P_m) of perhaps 0.01 or less. A random number between 0 and 1 is generated; if it falls within the P_m range, the new child organism's chromosome is randomly mutated in some way, typically by simply randomly altering bits in the chromosome data structure.

These processes ultimately result in a second generation pool of chromosomes that is different from the initial generation. Generally the average degree of fitness will have increased by this procedure for the second generation pool, since only the best organisms from the first generation are selected for breeding. The entire process is repeated for this second generation: each organism in the second generation pool is then evaluated, the fitness value for each organism is obtained, pairs are selected for breeding, a third generation pool is generated, etc. The process is repeated until an organism is produced which gives a solution that is "good enough".

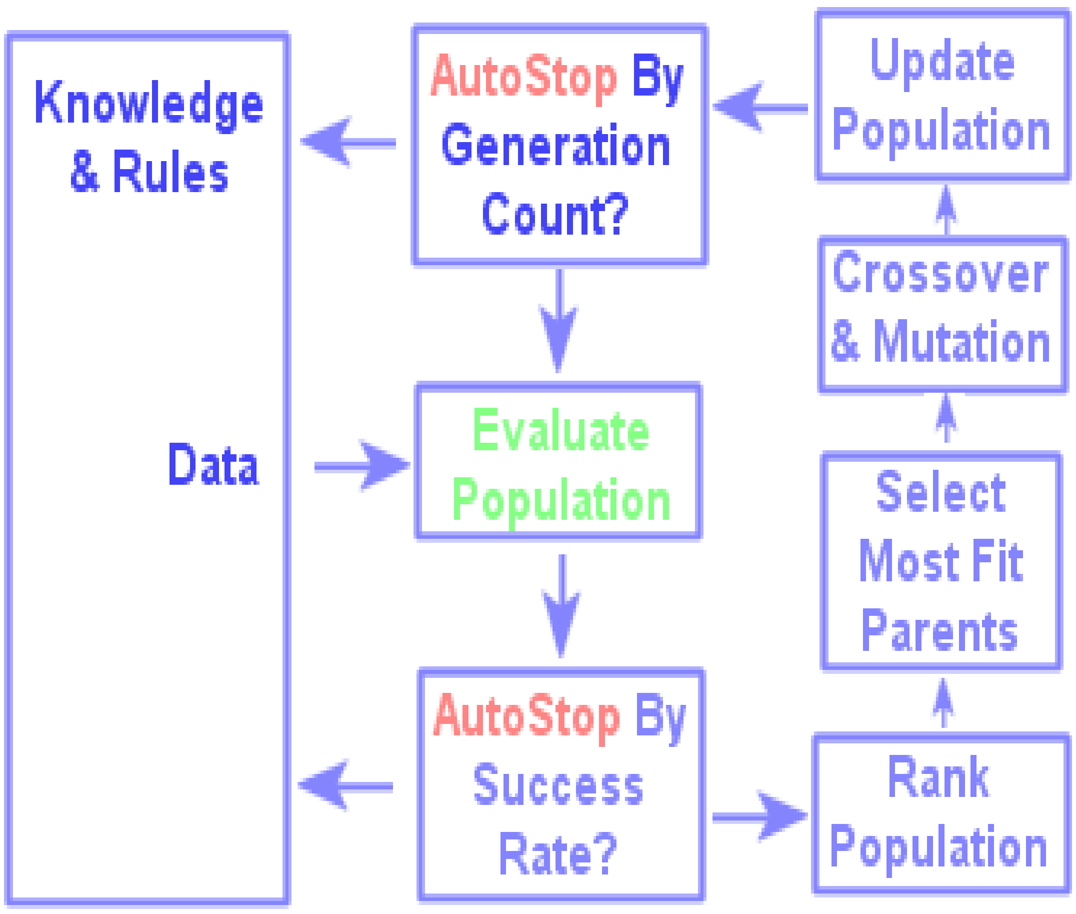


Figure 1 - The various stages of GA.

- **Fitness Functions**

Putative solutions to the target problem are evaluated using "Fitness functions", otherwise known as "Objective functions". Based upon the result of such functions, evolutionary pressures may be applied to a set of solutions. The objective function will obviously be problem specific. An advantage of GAs over many search or optimization algorithms is that derivatives of this function are not required. This fact ensures that GAs may be readily applied on fitness landscapes (or potential surfaces), which contain discontinuities or singularities without any special treatments.

- **Selection**

GA selection operators perform the equivalent role to natural selection. The overall effect is to bias the gene set in following generations to those genes which belong to the most fit individuals in the current generation.

There are numerous selection schemes described in the literature; "Roulette wheel" selection, tournament selection, random selection, stochastic sampling. These, in essence, mimic the processes involved in natural selection

- **Crossover**

There is also a multitude of methods to create a coherent set of genes from two parent sets. If following the schema formalism, crossover is the process by which good schema get combined. The most commonly used forms are shown in the figures below. Another alternative is chromosome mixing, where intact chromosomes are randomly swapped, which is highly advantageous in some applications.

- **Mutation**

The exact purpose of the mutation operations depends upon who you talk to. Mutations enable the GA to maintain diversity whilst also introducing some random search behavior. As for crossover, many types of mutation operator may be conceived depending upon the details of the problem and the chromosomal representation of solutions to that problem.

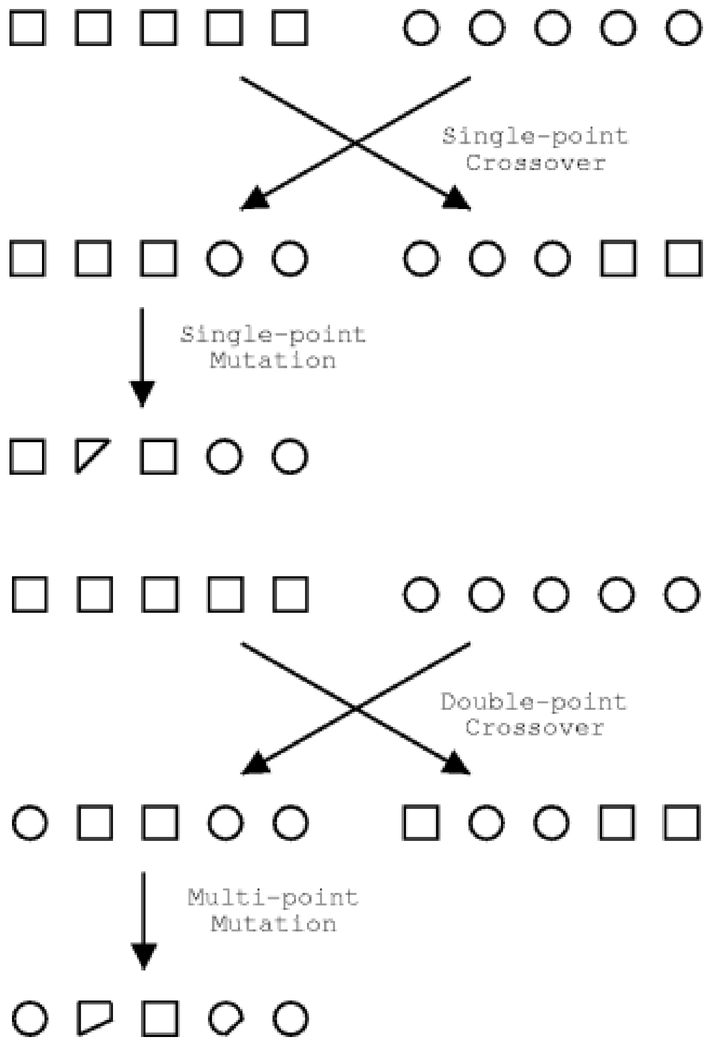


Figure -2 Crossovers and Mutation

2.3 Efficiency

Genetic algorithms are known to produce good results for some problem instances. Their major disadvantage is that they are relatively slow, being very computationally intensive compared to other heuristics, such as random optimization.

Recent speed improvements have focused on speciation, where crossover can only occur if individuals are closely-enough related. Genetic algorithms are extremely easy to adapt to parallel computing and clustering environments. One method simply treats each node as a parallel population. Organisms are then migrated from one pool to another according to various propagation techniques

2.4 Advantage of GA

One major advantage of genetic algorithms that other searches do not cover, is that of a false solution. A problem may have several solutions, some are better than others. If another type of search finds that one path is worth pursuing, and discards the others, it may be leading to a solution which is not the best, or even not a solution at all. Genetic algorithms have several individuals in its population, each one of which could be a potential solution to a problem. Each one of those individuals would be following a path to a possible solution, which means that it is even possible for the search to find more than one solution.

2.5 Applications of GA

- Automated design, including research on composite material design and multi-objective design of automotive components for crashworthiness, weight savings, and other characteristics.

- Automated design of mechatronic systems using bond graphs and genetic programming (NSF).
- Calculation of Bound States and Local Density Approximations.
- Configuration applications, particularly physics applications of optimal molecule configurations for particular systems like C60 (buckyballs).
- Container loading optimization.
- Distributed computer network topologies.
- Electronic circuit design, known as Evolvable hardware.
- File allocation for a distributed system.
- Parallelization of GAs/GPs including use of hierarchical decomposition of problem domains and design spaces nesting of irregular shapes using feature matching and GAs.
- Game Theory Equilibrium Resolution.
- Learning Robot behavior using Genetic Algorithms.
- Learning fuzzy rule base using genetic algorithms.
- Mobile communications infrastructure optimization.
- Molecular Structure Optimization (Chemistry).
- Multiple population topologies and interchange methodologies.
- Protein folding and protein/ligand docking.
- Plant floor layout.
- Scheduling applications, including job-shop scheduling. The objective being to schedule jobs in a sequence dependent or non-sequence dependent setup environment for a minimal total tardiness.
- Solving the machine-component grouping problem required for cellular manufacturing systems.

2.6 Genetic Algorithms for search and optimization

Genetic algorithms are now widely applied in science and engineering as adaptive algorithms for solving practical problems. Certain classes of problem are particularly suited to being tackled using a GA based approach.

The general acceptance is that GAs are particularly suited to multidimensional global search problems where the search space potentially contains multiple local minima. Unlike other search methods, correlation between the search variables is not generally a problem. The basic GA does not require extensive knowledge of the search space, such as likely solution bounds or functional derivatives. A task for which simple GAs are not suited is rapid local optimization; however, coupling the GA with other search techniques to overcome this problem is trivial. Whenever multidimensional systematic searching would be technique of choice, except that the large number of comparisons makes that approach intractable, a GA should be considered for the reasons outlined in the sections below.

Modern GAs deviate greatly from the original form proposed by Holland, but their linage is clear. There is no single firm definition for a genetic algorithm, and the computational system is highly simplified compared to the actual situation in nature. Therefore, we must first define a few terms and show how they relate between modern GAs and more traditional evolutionary theory.

Chapter III
Fuzzy Logic

Introduction

Natural language abounds with vague and imprecise concepts, such as "Sally is tall," or "It is very hot today." Such statements are difficult to translate into more precise language without losing some of their semantic value: for example, the statement "Sally's height is 152 cm." does not explicitly state that she is tall, and the statement "Sally's height is 1.2 standard deviations about the mean height for women of her age in her culture" is fraught with difficulties: would a woman 1.1999999 standard deviations above the mean be tall? Which culture does Sally belong to, and how is membership in it defined?

While it might be argued that such vagueness is an obstacle to clarity of meaning, only the most staunch traditionalists would hold that there is no loss of richness of meaning when statements such as "Sally is tall" are discarded from a language. Yet this is just what happens when one tries to translate human language into classic logic. Such a loss is not noticed in the development of a payroll program, perhaps, but when one wants to allow for natural language queries, or "knowledge representation" in expert systems, the meanings lost are often those being searched for.

For example, when one is designing an expert system to mimic the diagnostic powers of a physician, one of the major tasks is to codify the physician's decision-making process. The designer soon learns that the physician's view of the world, despite her dependence upon precise, scientific tests and measurements, incorporates evaluations of symptoms, and relationships between them, in a "fuzzy," intuitive manner: deciding how much of a particular medication to administer will have as much to do with the physician's sense of the relative "strength" of the patient's symptoms as it will their height/weight ratio. While some of the decisions and calculations could be done using traditional logic, we will see how fuzzy systems affords a broader, richer field of data and the manipulation of that data than do more traditional methods.

A fuzzy expert system is an expert system that uses a collection of Fuzzy membership functions and rules, instead of Boolean logic, to reason about data. The rules in a fuzzy expert system are usually of a form similar to the following:

if x is low and y is high then z = medium

Where x and y are input variables (names for know data values), z is an Output variable (a name for a data value to be computed), low is a membership function (fuzzy subset) defined on x, high is a membership function defined on y, and medium is a membership function defined on z.

The antecedent (the rule's premise) describes to what degree the rule applies, while the conclusion (the rule's consequent) assigns a membership functions to each of one or more output variables. Most tools for working with fuzzy expert systems allow more than one conclusion per rule. The set of rules in a fuzzy expert system is known as the rulebase or knowledge base.

Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth. Whereas classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces boolean truth values with degrees of truth.

Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition. To illustrate the difference, consider this scenario: Bob is in a house with two adjacent rooms: the kitchen and the dining room. In many cases, Bob's status within the set of things "in the kitchen" is completely plain: he's either "in the kitchen" or "not in the kitchen". What about when Bob stands in the doorway? He may be considered "partially in the kitchen". Quantifying this partial state yields a fuzzy set membership. With only his little toe in the dining room, we might say Bob is 0.99 "in the kitchen", for instance. No event (like a coin toss) will resolve Bob to being completely "in the kitchen" or "not in the kitchen", as long as he's standing in that doorway. Fuzzy sets are based on vague definitions of sets, not randomness.

Fuzzy logic allows for set membership values between and including 0 and 1, shades of gray as well as black and white, and in its linguistic form, imprecise concepts like "slightly", "quite" and "very". Specifically, it allows partial membership in a set. It is related to fuzzy sets and possibility theory. It was introduced in 1965 by Dr. Lotfi Zadeh of Berkeley.

The general inference process proceeds in three (or four) steps.

1. Under FUZZIFICATION, the membership functions defined on the Input variables are applied to their actual values, to determine the degree of truth for each rule premise.

2. Under INFERENCE, the truth value for the premise of each rule is computed, and applied to the conclusion part of each rule. These results in one fuzzy subset to be assigned to each output variable for each rule.

Usually only MIN or PRODUCT are used as inference rules. In MIN inferencing, the output membership function is clipped off at a height corresponding to the rule premise's computed degree of truth (fuzzy logic AND). In PRODUCT inferencing, the output membership function is scaled by the rule premise's computed degree of truth.

3. Under COMPOSITION, all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. Again, usually MAX or SUM are used. In MAX composition, the combined output fuzzy subset is constructed by taking the point wise maximum over all of the fuzzy subsets assigned to variable by the inference rule (fuzzy logic OR). In SUM composition, the combined output fuzzy subset is constructed by taking the point wise sum over all of the fuzzy subsets assigned to the output variable by the inference rule.

4. Finally is the (optional) DEFUZZIFICATION, which is used when it is useful to convert the fuzzy output set to a crisp number. There are more defuzzification methods than you can shake a stick at (at least 30). Two of the more common techniques are the CENTROID and MAXIMUM methods. In the CENTROID method, the crisp value of the output variable is computed by finding the variable value of the center of gravity of the membership function for the fuzzy value. In the MAXIMUM method, one of the variable values at which the fuzzy subset has its maximum truth value is chosen as the crisp value for the output variable.

3.2 Fuzzy Logic vs. conventional control methods

FL incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. For example, rather than dealing with temperature control in terms such as "SP =500F", "T <1000F", or "210C <TEMP <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Consider what you do in the shower if the temperature is too cold: you will make the water comfortable very quickly with little trouble. FL is capable of mimicking this type of behavior but at very high rate.

3.3 Features of Fuzzy Logic

FL offers several unique features that make it a particularly good choice for many control problems.

1) It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations.

2) Since the FL controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. New sensors can easily be incorporated into the system simply by generating appropriate governing rules.

3) FL is not limited to a few feedback inputs and one or two control outputs, nor is it necessary to measure or compute rate-of-change parameters in order for it to be implemented. Any sensor data that provides

some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system cost and complexity low.

4) Because of the rule-based operation, any reasonable number of inputs can be processed (1-8 or more) and numerous outputs (1-4 or more) generated, although defining the rule base quickly becomes complex if too many inputs and outputs are chosen for a single implementation since rules defining their interrelations must also be defined. It would be better to break the control system into smaller chunks and use several smaller FL controllers distributed on the system, each with more limited responsibilities.

5) FL can control nonlinear systems that would be difficult or impossible to model mathematically. This opens doors for control systems that would normally be deemed unfeasible for automation.

3.4 Membership functions

The rule matrix was introduced and used. The next logical question is how to apply the rules. This leads into the next concept, the membership function. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. There are different membership functions associated with each input and output response. Some features to note are:

SHAPE - triangular is common, but bell, trapezoidal, haversine and, exponential have been used. More complex functions are possible but require greater computing overhead to implement. HEIGHT or magnitude (usually normalized to 1) WIDTH (of the base of function)

SHOULDERING (locks height at maximum if an outer function. Shouldered functions evaluate as 1.0 past their center) CENTER points (center of the member function shape) OVERLAP (N&Z, Z&P, typically about 50% of width but can be less).

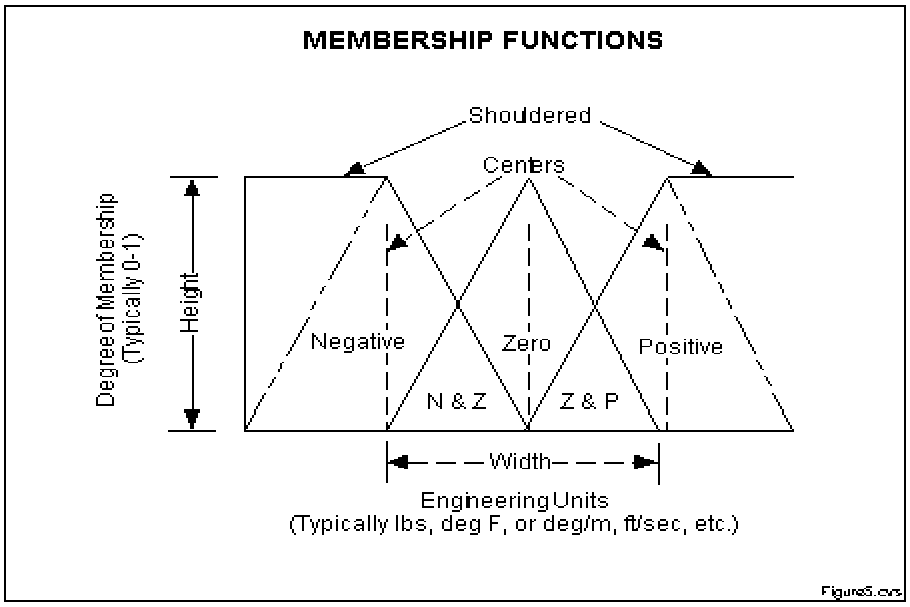


Figure 3 - The features of a membership function

Figure 3 illustrates the features of the triangular membership function which is used in this example because of its mathematical simplicity. Other shapes can be used but the triangular shape lends itself to this illustration.

The degree of membership (DOM) is determined by plugging the selected input parameter (error or error-dot) into the horizontal axis and projecting vertically to the upper boundary of the membership function(s).

3.5 Applications

Fuzzy logic can be used to control household appliances such as washing machines (which sense load size and detergent concentration and adjust their wash cycles accordingly) and refrigerators.

A basic application might characterize sub ranges of a continuous variable. For instance, a temperature measurement for anti-lock brakes might have several separate membership functions defining particular temperature ranges needed to control the brakes properly. Each function maps the same temperature value to a truth value in the 0 to 1 range. These truth values can then be used to determine how the brakes should be controlled.

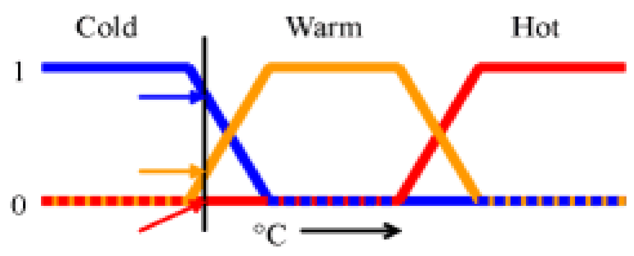


Figure 4

In this image, cold, warm, and hot are functions mapping a temperature scale. A point on that scale has three "truth values" — one for each of the three functions. For the particular temperature shown, the three truth values could be interpreted as describing the temperature as, say, "fairly cold", "slightly warm", and "not at all hot".

A more sophisticated practical example is the use of fuzzy logic in high-performance error correction to improve information reception over a limited-bandwidth communication link affected by data-corrupting noise using turbo codes. The front-end of a decoder produces a likelihood measure for the value intended by the sender (0 or 1) for each bit in the data stream. The likelihood measures might use a scale of 256 values between extremes of "certainly 0" and "certainly 1". Two decoders may analyse the data in parallel, arriving at different likelihood results for the values intended by the sender. Each can then use as additional data the other's likelihood results, and repeats the process to improve the results until consensus is reached as to the most likely values.

Areas in which fuzzy logic has been successfully applied are often quite concrete. The first major commercial application was in the area of cement kiln control, an operation which requires that an operator monitor four internal states of the kiln, control four sets of operations, and dynamically manage 40 or 50 "rules of thumb" about their interrelationships, all with the goal of controlling a highly complex set of chemical interactions. One such rule is "If the oxygen percentage is rather high and the free-lime and kiln-drive torque rate is normal, decrease the flow of gas and slightly reduce the fuel rate".

Chapter IV
Genofuzzy Computing

Introduction

Fuzzy logic has been applied very successfully in many areas where conventional model based approaches are difficult or not cost-effective to implement. However, as system complexity increases, reliable fuzzy rules and membership functions used to describe the system behavior are difficult to determine. Furthermore, due to the dynamic nature of economic and financial applications, rules and membership functions must be adaptive to the changing environment in order to continue to be useful.

The main advantage of using genetic algorithms (GAs) in economic or financial modeling is that they can be synthesized without making use of the detailed, explicit knowledge of the underlying process. However, limited or noisy training data may result in inconsistent, meaningless output. This has been known to be a severe problem of genetic algorithms.

Because of their complementary nature, these two technologies can be integrated in a number of ways to overcome the drawbacks of each other. A GenoFuzzy hybrid system is described in this chapter for the purpose of explanation.

4.2 A GenoFuzzy Hybrid System

GenoFuzzy hybrid systems combine the advantages of fuzzy systems, which deal with explicit knowledge that can be explained and understood, and genetic algorithms which deal with implicit knowledge that can be acquired by genetic search, which provides a good way to adjust the expert's knowledge and automatically generate additional fuzzy rules, to meet certain specifications and reduce design time and costs. On the other hand, fuzzy logic enhances the generalization capability of a genetic system by providing more reliable output when extrapolation is needed beyond the limits of the training data.

4.3 The GenoFuzzy Architecture

The GenoFuzzy system consists of the various components of a traditional fuzzy system, except that each stage is performed by layers of rude nodes, and a genetic search capability is provided to acquire and enhance the system knowledge.

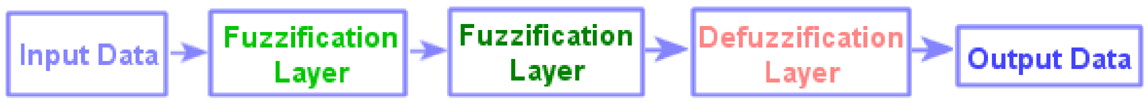


Fig. 5. The schematic of a GenoFuzzy System architecture.

The Fuzzification Layer - Each neuron in this layer represents an input membership function of the antecedent of a fuzzy rule. One common method to implement this layer is to express membership functions as discrete points. Thus for a fuzzy rule "IF X1 is A! and X2 is A2 ... THEN Y is B", A's characterize the possibility distribution of the antecedent clause "X is A". Each of the hidden nodes is defined as a fuzzy reference point in the input space. This method can approximate many continuous functions and the degree of error depends very much on the number of discrete points used. Another much better approach is to use a combination of one to two sigmoidal functions and a linear function to represent each membership function in the Fuzzication and Defuzzification layers. The parameters of these neurons can be trained to fine tune the final shape and location of the membership functions. In most designs, the number of neurons in this layer is fixed, but it is possible to add or remove these neurons during training, according to the outputs produced on the training samples.

In Fig.3, the membership grade indicating the certainty of "X1 is High" is 0.6, "X1 is Medium" is 0.4, and "X1 is Low" is 0.0. The output of these membership function neurons are connected to the Fuzzy Rule Layer as specified by the fuzzy rules, using links with fixed weights of unity.

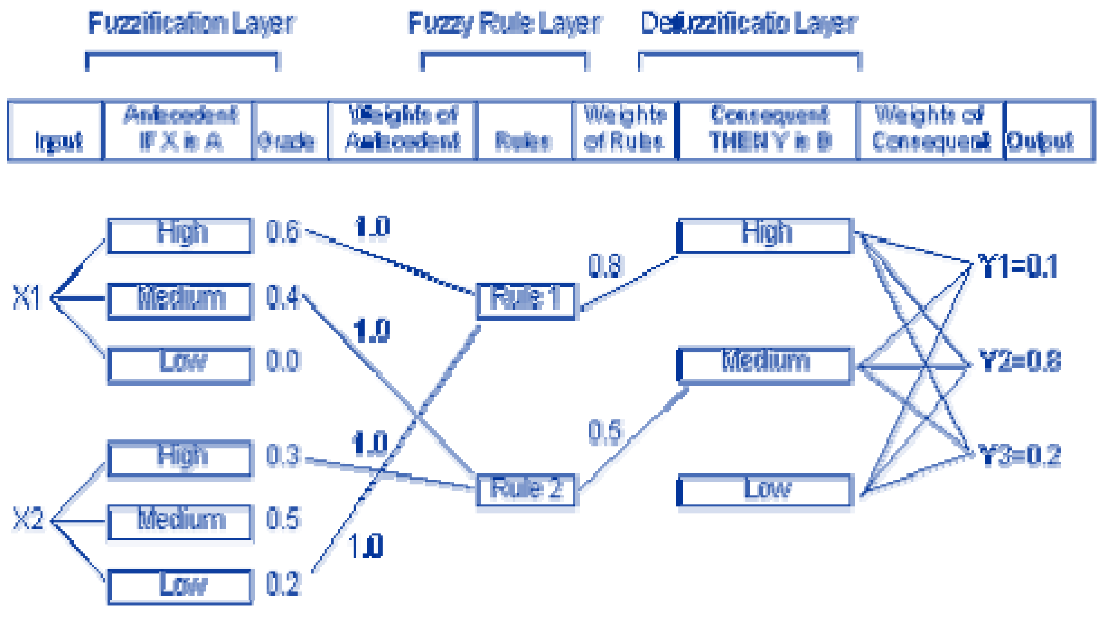


Fig. 6. Implementation of fuzzy rules.

The Fuzzy Rule Layer - It represents the fuzzy rule base and its function is to perform the fuzzy logical operations. Each neuron represents a fuzzy rule such as "IF X1 is A1 and X2 is A2 ... THEN Y is B" and it calculates the certainty of each compound proposition "IF X1 is A1 and X2 is A2 ... " which indicates the goodness of fit, that is, how well the prerequisites of each fuzzy rule are satisfied. The neurons have a linear function, and their output are connected to the Defuzzification Layer by weighted links. The weights of these links represent the relative significance of the rules associated with the neurons. Their values can be preset according to the expert or initialized to be 1.0, and then trained to reflect their actual importance to the output membership functions contained in the Defuzzification Layer.

The Defuzzification Layer - The function of this layer is for rule evaluation. Each neuron in this layer represents a consequent proposition "THEN Y is B" and its membership function can be implemented by combining one or two sigmoidal functions and linear functions. The certainty of each consequent proposition is calculated, and is regarded as the goodness of fit of those fuzzy rules which have the same consequent proposition. The weight of each output link from these nodes represents the center of gravity of each output membership function of the consequent, and is trainable. The final output value is then calculated using the center of gravity method.

4.4 Fuzzy Rule Implementation

In the GenoFuzzy system illustrated in Fig. 3, can be programmed easily. A more general format is given below:

Rule 1: IF X1 is High and X2 is Low THEN Y is High 0.8

Rule 2: IF X1 is Medium and X2 is High THEN Y is Medium .0.5

The value at the end of each rule represents the initial weight of the rule, and will be adjusted to its appropriate level at the end of training. If all the rules have the same subject "Y" for the consequent propositions, then only 1 output node is needed. Fig. 3 illustrates a system with three subjects "Y1", "Y2" and "Y3". The output values could be Dollars transacted, credit worthiness, valuation of real estates, market indices, buy/sell signals and etc.

4.5 Training the GenoFuzzy System

The structure in Fig. 3 can be configured with initial values specified by human experts, and then further tuned by using a genetic search algorithm as follows:

Step 1: Present an input data sample, compute the corresponding output

Step 2: Compute the error between the output(s) and the actual target(s)

Step 3: The connection weights and membership functions are adjusted

Step 4: At a fixed number of epochs, delete useless rule and membership function nodes, and add in new ones

Step 5: IF Success Rate >= Tolerance THEN go to Step 1 ELSE stop.

When the error level drops to below the user-specified tolerance, the final interconnection weights reflect the changes in the initial fuzzy rules and membership functions. If the resulting weight of a rule is close to zero, the rule can be safely removed from the rule base, since it is insignificant compared to others. Also, the shape and position of the membership functions in the Fuzzification and Defuzzification Layers can be fine tuned by adjusting the parameters of the nodes in these layers, during the training process.

Chapter V
GenoFuzzy Application To Telemedicine

Introduction

In the previous chapters we studied the basic concepts of Fuzzy, Genetic, Telemedicine and the advantages of applying Fuzzy Genetic in various fields. Fuzzy Genetic Algorithm can also be applied to different diseases in telemedicine area so that a doctor sitting at some other location can prescribe a required medicine to the patient. This work is on Blood Pressure monitoring in which after obtaining the data of BP, the same can be sent to distant location where the doctor is available for consultation. By using Fuzzy Genetic technique I have tried to find out the severity of disease so that doctor can prescribe the medicine on the basis of the fact that in what range the Blood pressure actually lies in.

Before explaining the logic behind the whole work its important to know that what are the measuring areas to monitor Blood Pressure.

Normal Blood Pressure for an adult is 120 **/ 80**.

Where: **120** is called as **Systolic Pressure**

: **80** is called as **Diastolic Pressure**.

If the B.P of a person does not lie around this standard value then it can cause a serious problem. So the aim of my project is to find the precise value of range of disease in which the Blood Pressure lies.

The main problem in fuzzy function selection is difficulty in defining the host parameters such as the number and shape of fuzzy sets for symptoms and inference mechanism. The choice of parameters has considerable influence on diagnosing of disease. In this project I have considered three triangular fuzzy functions as shown in the below given fig. The three functions X_1, X_2, and X_3 uniquely define the fuzzy set. Fuzzy set representation for the following is shown:

1. Systolic Pressure

2. Diastolic Pressure

3. Disease.

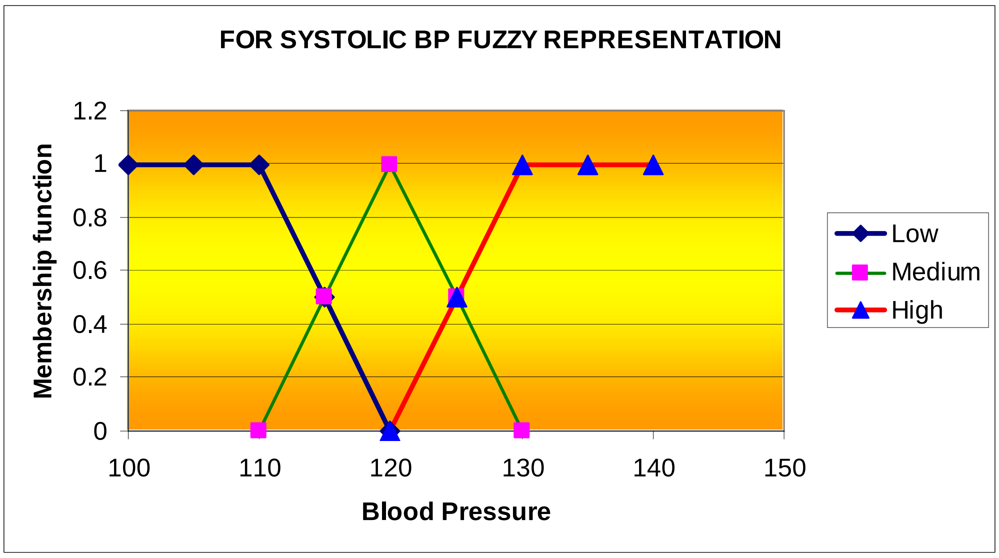


Figure 7

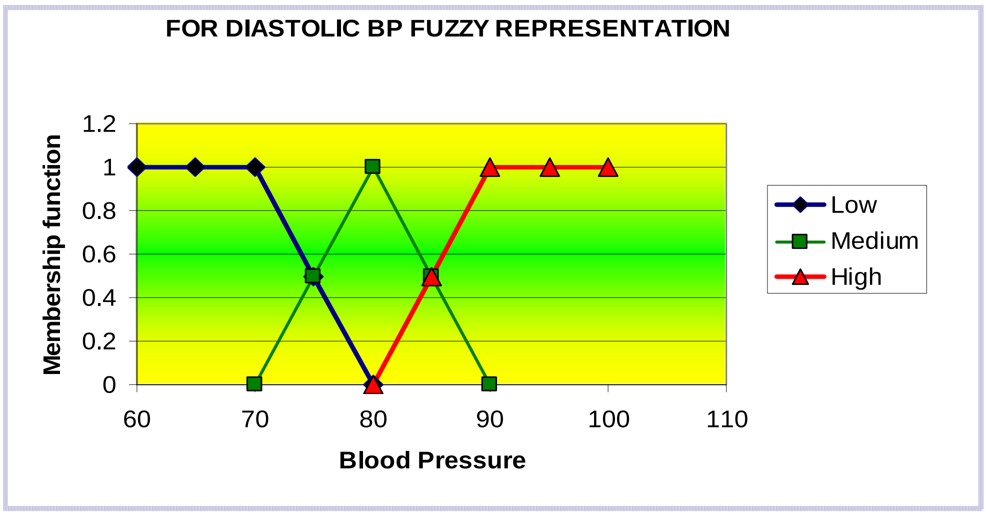


Figure 8

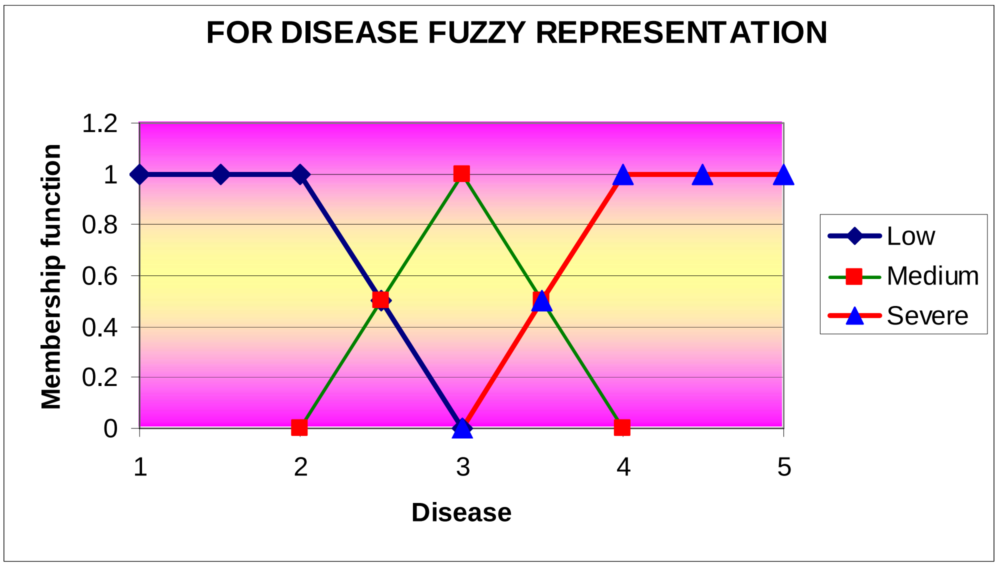


Figure 9

The Fuzzy set μ(L), μ (M) and μ(H) can be represented by the equations:

For Diastolic Pressure, equations are:

μ (L) = 1 , BP$\leq$ 70

= (80-BP)/10 ,70$\leq$BP$\leq$80

= 0 , otherwise.

μ (M) = (80-BP)/10 , 70$\leq$BP$\leq$80

=1 , BP =80

= (BP-90)/10 ,80$\leq$BP$\leq$90

= 0 , otherwise

μ (H) = (BP-90)/10 ,80$\leq$BP$\leq$90

= 1 , BP$\geq$90

For Systolic Pressure, equations are:

μ (L) =1 , BP$\leq$ 110

= (120-BP)/10 ,110$\leq$BP$\leq$120

= 0 , otherwise.

μ (M) = (120-BP)/10 , $110 \le BP \le 120$

=1 , BP =120

= (BP-130)/10 ,$120 \le BP \le 130$

= 0 , otherwise

μ (H) = (BP-130)/10 ,$120 \le BP \le 130$

= 1 , $BP \ge 130$

For Disease, equations are:

μ (L) =1 , $D \le 2$

= (3-D)/1 ,$2 \le D \le 3$

= 0 , otherwise.

μ (M) = (3-D)/1 ,$2 \le D \le 3$

=1 , D =3

= (D-4)/1 ,$3 \le D \le 4$

= 0 , otherwise

μ (H) = (D-4)/1 ,$3 \le D \le 4$

= 1 , $D \ge 4$

Fuzzy function for Systolic Pressure is shown in Figure 7 and can be written in fuzzy form as given below:

1.0/100 + 1.0/105 + 1.0/110 + 0.5/115 + 0/120 + 0/125 + 0/130 + 0/135 + 0/140 **(low)**

0/100 + 0/105 + 0/110 + 0.5/115 + 1.0/120 + 0.5/125 + 0/130 + 0/135 + 0/140**(Medium)**

0/100 + 0/105 + 0/110 + 0/115 + 0/120 + 0.5/125 + 1.0/130 + 1.0/135 + 1.0/140 **(high)**

Fuzzy function for Diastolic Pressure is shown in Figure 8 and can be written in fuzzy form as given below:

1.0/60 + 1.0/65 + 1.0/70 + 0.5/75 + 0/80 + 0/85 + 0/90 + 0/95 + 0/100 **(low)**

0/60 + 0/65 + 0/70 + 0.5/75 + 1.0/80 + 0.5/85 + 0/90 + 0/95 + 0/100 **(Medium)**

0/60 + 0/65 + 0/70 + 0/75 + 0/80 + 0.5/85 + 1.0/90 +1.0/95 + 1.0/100 **(high)**

Fuzzy function for Disease is shown in figure 9 and can be written in fuzzy form as given below:

1.0/1 + 1.0/1.5 + 1.0/2 + 0.5/2.5 + 0/3 + 0/3.5 + 0/4 **(low)**

0/1 + 0/1.5 + 0/2 + 0.5/2.5 + 1/3 + 0.5/3.5 + 0/4 **(Medium)**

0/1 + 0/1.5 + 0/2 + 0/2.5 + 0/3 + 0.5/3.5 + 1/4**(high)**

The three strings for the functions of Systolic pressure can be written as:

1010 1010 1010 0101 0000 0000 0000 0000 0000

0000 0000 0000 0101 1010 0101 0000 0000 0000

0000 0000 0000 0000 0000 0101 1010 1010 1010

The three strings for the functions of Diastolic pressure can be written as:

1010 1010 1010 0101 0000 0000 0000 0000 0000

0000 0000 0000 0101 1010 0101 0000 0000 0000

0000 0000 0000 0000 0000 0101 1010 1010 1010

Now the crossover and mutation operators are applied at different locations to shape the input fuzzy function according to genetic process. The genetic is first applied to the low range and then to the high range.

5.2 For Low Range (Diastolic Pressure)

Diastolic Binary Form is : $\underbrace{1010}_{x_1}\ \underbrace{1010}_{x_2}\ \underbrace{1010}_{x_3}\ \underbrace{0101}_{x_4}$

Where x_1, x_2 , x_3 , x_4 represents the decimal values of the corresponding binary nibbles.

1010 1010 1010 0101 String1

0000 0000 0000 0000 String 2 **(any random string)**

Two-point crossover is done and the two output strings are again obtained. Mutation operation is performed such that always the nibble x_2 should be1010 i.e a bitwise operation of converting a zero to one or one to zero. Now the output strings obtained after mutation are checked to get the fittest value by applying the fitness function formula.

Let the final string be :

$$\underbrace{a_1\, a_2\, a_3\, a_4}_{x_1}\quad \underbrace{a_5\, a_6\, a_7\, a_8}_{x_2}\quad \underbrace{a_9\, a_{10}\, a_{11}\, a_{12}}_{x_3}\quad \underbrace{a_{13}\, a_{14}\, a_{15}\, a_{16}}_{x_4}$$

Formula is:

$$D = \frac{(x_1 * 60) + (x_2 * 65) + (x_3 * 70) + (x_4 * 75)}{x_1 + x_2 + x_3 + x_4}$$

$= 65$ (approx.)

Now if we get the fittest value of the string i.e near 65 then we select the string as the final string otherwise we go for more iterations of crossover and mutations till we get the required value.

After getting the final string it can be represented in the fuzzy form as follows to plot the graph:

$$\mathbf{x_1/60 + x_2/65 + x_3/70 + x_4/75} \qquad \mathbf{(1)}$$

For Low Range (Systolic Pressure)

Systolic Binary Form is: $\underbrace{1010}_{y1}\ \underbrace{1010}_{y_2}\ \underbrace{1010}_{y_3}\ \underbrace{0101}_{y_4}$

Where y_1, y_2, y_3, y_4 represents the decimal values of the corresponding binary nibbles

1010 1010 1010 0101 String1

0000 0000 0000 0000 String 2 **(any random string**)

Two-point crossover is done and the two output strings are again obtained. Mutation operation is performed such that always the nibble y_2 should be1010 i.e a bitwise operation of converting a zero to one or one to zero. Now the output strings obtained after mutation are checked to get the fittest value by applying the fitness function formula.

Let the final string be:

$$\underbrace{a_1\,a_2\,a_3\,a_4}_{y_1}\quad \underbrace{a_5\,a_6\,a_7\,a_8}_{y_2}\quad \underbrace{a_9\,a_{10}\,a_{11}\,a_{12}}_{y_3}\quad \underbrace{a_{13}\,a_{14}\,a_{15}\,a_{16}}_{y_4}$$

Formula is:

$$D = \frac{\mathbf{(Y_1 * 60) + (y_2 * 65) + (Y_3 * 70) + (y_4 * 75)}}{\mathbf{Y_1 + y_2 + y_3 + y_4}}$$

$= 105$ (approx.)

Now if we get the fittest value of the string i.e. near 105 then we select the string as the final string otherwise we go for more iterations of crossover and mutations till we get the required value.

After getting the final string it can be represented in the fuzzy form as follows to plot the graph:

$\mathbf{y_1/100 + y_2/105 + y_3/110 + y_4/115}$ **(2)**

Low Range Disease in fuzzy form

$\mathbf{1.0/1 + 1.0/1.5 + 1.0/2 + 0.5/2.5}$ **(3)**

Now we are having three equations i.e. (1), (2) ,(3) showing the fuzzy final form after applying genetic algorithm to the low range of diastolic and systolic pressure. Next step is to find relation matrix i.e.

If systolic BP is Linguistic value, and diastolic BP is Linguistic Value then disease is a linguistic value.

Note: Linguistic value can be anything i.e. low, high, and medium.

Hence we apply mamdani rule to find a relation matrix with the help of which we can find where the disease lies.

Cylindrical Extension of Diastolic BP. TABLE 1

XY/Z	1	1.5	2	2.5
60/100	X1	X1	X1	X1
60/105	X1	X1	X1	X1
60/110	X1	X1	X1	X1
60/115	X1	X1	X1	X1
65/100	X2	X2	X2	X2
65/105	X2	X2	X2	X2
65/110	X2	X2	X2	X2
65/115	X2	X2	X2	X2
70/100	X3	X3	X3	X3
70/105	X3	X3	X3	X3
70/110	X3	X3	X3	X3
70/115	X3	X3	X3	X3

XY/Z	1	1.5	2	2.5
75/100	X4	X4	X4	X4
75/105	X4	X4	X4	X4
75/110	X4	X4	X4	X4
75/115	X4	X4	X4	X4

Cylindrical Extension of Systolic BP. TABLE 2

XY/Z	1	1.5	2	2.5
60/100	Y1	Y1	Y1	Y1
60/105	Y2	Y2	Y2	Y2
60/110	Y3	Y3	Y3	Y3
60/115	Y4	Y4	Y4	Y4
65/100	Y1	Y1	Y1	Y1
65/105	Y2	Y2	Y2	Y2
65/110	Y3	Y3	Y3	Y3
65/115	Y4	Y4	Y4	Y4
70/100	Y1	Y1	Y1	Y1
70/105	Y2	Y2	Y2	Y2
70/110	Y3	Y3	Y3	Y3
70/115	Y4	Y4	Y4	Y4
75/100	Y1	Y1	Y1	Y1
75/105	Y2	Y2	Y2	Y2
75/110	Y3	Y3	Y3	Y3
75/115	Y4	Y4	Y4	Y4

Cylindrical Extension of Disease. TABLE 3

XY/Z	1	1.5	2	2.5
60/100	1	1	1	0.5
60/105	1	1	1	0.5
60/110	1	1	1	0.5
60/115	1	1	1	0.5
65/100	1	1	1	0.5
65/105	1	1	1	0.5
65/110	1	1	1	0.5
65/115	1	1	1	0.5
70/100	1	1	1	0.5
70/105	1	1	1	0.5

XY/Z	1	1.5	2	2.5
70/110	1	1	1	0.5
70/115	1	1	1	0.5
75/100	1	1	1	0.5
75/105	1	1	1	0.5
75/110	1	1	1	0.5
75/115	1	1	1	0.5

Minimization of Tables 1,2,3 gives a relation matrix R

XY/Z	1	1.5	2	2.5
60/100	A1	A2	A3	A4
60/105	A5	A6	A7	A8
60/110	A9	A10	A11	A12
60/115	A13	A14	A15	A16
65/100	A17	A18	A19	A20
65/105	A21	A22	A23	A24
65/110	A25	A26	A217	A28
65/115	A29	A30	A31	A32
70/100	A33	A34	A35	A36
70/105	A37	A38	A39	A40
70/110	A41	A42	A43	A44
70/115	A45	A46	A47	A48
75/100	A49	A50	A51	A52
75/105	A53	A54	A55	A56
75/110	A57	A58	A59	A60
75/115	A61	A62	A63	A64

Now let us take one example if suppose BP is 70/100 then we have

Matrix S

XY/Z	1	1.5	2	2.5
60/100	0	0	0	0
60/105	0	0	0	0
60/110	0	0	0	0
60/115	0	0	0	0
65/100	0	0	0	0
65/105	0	0	0	0
65/110	0	0	0	0

XY/Z	1	1.5	2	2.5
65/115	0	0	0	0
70/100	1	1	1	1
70/105	0	0	0	0
70/110	0	0	0	0
70/115	0	0	0	0
75/100	0	0	0	0
75/105	0	0	0	0
75/110	0	0	0	0
75/115	0	0	0	0

Minimization of R and S gives

XY/Z	1	1.5	2	2.5
60/100	0	0	0	0
60/105	0	0	0	0
60/110	0	0	0	0
60/115	0	0	0	0
65/100	0	0	0	0
65/105	0	0	0	0
65/110	0	0	0	0
65/115	0	0	0	0
70/100	a	b	c	d
70/105	0	0	0	0
70/110	0	0	0	0
70/115	0	0	0	0
75/100	0	0	0	0
75/105	0	0	0	0
75/110	0	0	0	0
75/115	0	0	0	0

Projection on disease axis is a b c d

Fuzzy set = a/1 + b/1.5 + c/2 + d/2.5

Defuzzified value of disease

$$Df = \frac{(a*1) + (b*1.5) + (c*2) + (d*2.5)}{a+b+c+d}$$

= let value is B

This value should lie in the low range of disease i.e. 1 to 3.

5.3 For High Range (Diastolic Pressure)

Diastolic Binary Form is : $\underbrace{0101}_{x_1}\ \underbrace{1010}_{x_2}\ \underbrace{1010}_{x_3}\ \underbrace{1010}_{x_4}$

Where x_1, x_2, x_3, x_4 represents the decimal values of the corresponding binary nibbles.

0101 1010 1010 1010 String1

0000 0000 0000 0000 String 2 **(any random string)**

Two-point crossover is done and the two output strings are again obtained. Mutation operation is performed such that always the nibble x_3 should be1010 i.e. a bitwise operation of converting a zero to one or one to zero. Now the output strings obtained after mutation are checked to get the fittest value by applying the fitness function formula .

Let the final string be :

$\underbrace{a_1\, a_2\, a_3\, a_4}_{x_1}\quad \underbrace{a_5\, a_6\, a_7\, a_8}_{x_2}\quad \underbrace{a_9\, a_{10}\, a_{11}\, a_{12}}_{x_3}\quad \underbrace{a_{13}\, a_{14}\, a_{15}\, a_{16}}_{x_4}$

Formula is:

$$D = \frac{(x_1 * 85) + (x_2 * 90) + (x_3 * 95) + (x_4 * 100)}{x_1 + x_2 + x_3 + x_4}$$

= 95 (approx.)

Now if we get the fittest value of the string i.e. near 95 then we select the string as the final string otherwise we go for more iterations of crossover and mutations till we get the required value.

After getting the final string it can be represented in the fuzzy form as follows to plot the graph:

$$\mathbf{x_1/85 + x_2/90 + x_3/95 + x_4/100} \qquad \mathbf{(1)}$$

For High Range (Systolic Pressure)

Systolic Binary Form is: $\underbrace{0101}_{Y_1}\ \underbrace{1010}_{y_2}\ \underbrace{1010}_{y_3}\ \underbrace{1010}_{y_4}$

Where y_1, y_2, y_3, y_4 represents the decimal values of the corresponding binary nibbles

0101 1010 1010 1010 String1

0000 0000 0000 0000 String 2 **(any random string**)

Two-point crossover is done and the two output strings are again obtained. Mutation operation is performed such that always the nibble y_3 should be1010 i.e. a bitwise operation of converting a zero to one or one to zero. Now the output strings obtained after mutation are checked to get the fittest value by applying the fitness function formula.

Let the final string be:

$\underbrace{a_1\, a_2\, a_3\, a_4}_{Y_1}\quad \underbrace{a_5\, a_6\, a_7\, a_8}_{y_2}\quad \underbrace{a_9\, a_{10}\, a_{11}\, a_{12}}_{y_3}\quad \underbrace{a_{13}\, a_{14}\, a_{15}\, a_{16}}_{y_4}$

Formula is:

$$\mathbf{D = \frac{(y_1 * 125) + (y_2 * 130) + (Y_3 * 135) + (y_4 * 140)}{y_1 + y_2 + y_3 + y_4}}$$

= 135 (approx.)

Now if we get the fittest value of the string i.e. near 135 then we select the string as the final string otherwise we go for more iterations of crossover and mutations till we get the required value.

After getting the final string it can be represented in the fuzzy form as follows to plot the graph:

$$\mathbf{y_1/125 + y_2/130 + y_3/135 + y_4/140} \qquad \mathbf{(2)}$$

High Range Disease in fuzzy form

$$\mathbf{0/3 + 0.5/3.5 + 1.0/4 + 1/4.5} \qquad \mathbf{(3)}$$

Now we are having three equations i.e. (1), (2) ,(3) showing the fuzzy final form after applying genetic algorithm to the low range of diastolic and systolic pressure. Next step is to find relation matrix i.e.

If systolic BP is Linguistic value

And diastolic BP is Linguistic Value

Then disease is a linguistic value.

Note: Linguistic value can be anything i.e. low, high, and medium.

Hence we apply mamdani rule to find a relation matrix with the help of which we can find where the disease lies.

Cylindrical Extension of Diastolic BP. TABLE 1

XY/Z	3	3.5	4	4.5
85/125	X1	X1	X1	X1
85/130	X1	X1	X1	X1
85/135	X1	X1	X1	X1
85/140	X1	X1	X1	X1
90/125	X2	X2	X2	X2
90/130	X2	X2	X2	X2
90/135	X2	X2	X2	X2
90/140	X2	X2	X2	X2
95/125	X3	X3	X3	X3
95/130	X3	X3	X3	X3
95/135	X3	X3	X3	X3
95/140	X3	X3	X3	X3
100/125	X4	X4	X4	X4
100/130	X4	X4	X4	X4
100/135	X4	X4	X4	X4
100/140	X4	X4	X4	X4

Cylindrical Extension of Systolic BP. TABLE 2

XY/Z	3	3.5	4	4.5
85/125	Y1	Y1	Y1	Y1
85/130	Y2	Y2	Y2	Y2
85/135	Y3	Y3	Y3	Y3
85/140	Y4	Y4	Y4	Y4
90/125	Y1	Y1	Y1	Y1
90/130	Y2	Y2	Y2	Y2
90/135	Y3	Y3	Y3	Y3
90/140	Y4	Y4	Y4	Y4
95/125	Y1	Y1	Y1	Y1
95/130	Y2	Y2	Y2	Y2
95/135	Y3	Y3	Y3	Y3
95/140	Y4	Y4	Y4	Y4
100/125	Y1	Y1	Y1	Y1
100/130	Y2	Y2	Y2	Y2
100/135	Y3	Y3	Y3	Y3
100/140	Y4	Y4	Y4	Y4

Cylindrical Extension of Disease. TABLE 3

XY/Z	3	3.5	4	4.5
85/125	0	0.5	1	1
85/130	0	0.5	1	1
85/135	0	0.5	1	1
85/140	0	0.5	1	1
90/125	0	0.5	1	1
90/130	0	0.5	1	1
90/135	0	0.5	1	1
90/140	0	0.5	1	1
95/125	0	0.5	1	1
95/130	0	0.5	1	1
95/135	0	0.5	1	1
95/140	0	0.5	1	1
100/125	0	0.5	1	1
100/130	0	0.5	1	1
100/135	0	0.5	1	1
100/140	0	0.5	1	1

Minimization of Tables 1,2,3 gives a relation matrix R

XY/Z	3	3.5	4	4.5
85/125	A1	A2	A3	A4
85/130	A5	A6	A7	A8
85/135	A9	A10	A11	A12
85/140	A13	A14	A15	A16
90/125	A17	A18	A19	A20
90/130	A21	A22	A23	A24
90/135	A25	A26	A217	A28
90/140	A29	A30	A31	A32
95/125	A33	A34	A35	A36
95/130	A37	A38	A39	A40
95/135	A41	A42	A43	A44
95/140	A45	A46	A47	A48
100/125	A49	A50	A51	A52
100/130	A53	A54	A55	A56
100/135	A57	A58	A59	A60
100/140	A61	A62	A63	A64

Now let us take one example if suppose BP is 95/125 then we have Matrix S

XY/Z	3	3.5	4	4.5
85/125	0	0	0	0
85/130	0	0	0	0
85/135	0	0	0	0
85/140	0	0	0	0
90/125	0	0	0	0
90/130	0	0	0	0
90/135	0	0	0	0
90/140	0	0	0	0
95/125	1	1	1	1
95/130	0	0	0	0
95/135	0	0	0	0
95/140	0	0	0	0
100/125	0	0	0	0

XY/Z	3	3.5	4	4.5
100/130	0	0	0	0
100/135	0	0	0	0
100/140	0	0	0	0

Minimization of R and S gives

XY/Z	3	3.5	4	4.5
85/125	0	0	0	0
85/130	0	0	0	0
85/135	0	0	0	0
85/140	0	0	0	0
90/125	0	0	0	0
90/130	0	0	0	0
90/135	0	0	0	0
90/140	0	0	0	0
95/125	a	b	c	d
95/130	0	0	0	0
95/135	0	0	0	0
95/140	0	0	0	0
100/125	0	0	0	0
100/130	0	0	0	0
100/135	0	0	0	0
100/140	0	0	0	0

Projection on disease axis is a b c d

Fuzzy set = a/3 + b/3.5 + c/4 + d/4.5

Defuzzified value of disease

$$Df = \frac{(a*3) + (b*3.5) + (c*4) + (d*4.5)}{a+b+c+d}$$

= let value is B

This value should lie in the high range of disease i.e. 3 or above

5.4 For Medium range

No genetic is applied to the medium range fuzzy function and as already explained that the fuzzy function for medium range is :

For Diastolic

0/60 + 0/65 + 0/70 + 0.5/75 + 1.0/80 + 0.5/85 + 0/90 + 0/95 + 0/100 **(Medium)** **(1)**

For Systolic

0/100 + 0/105 + 0/110 + 0.5/115 + 1.0/120 + 0.5/125 + 0/130 + 0/135 + 0/140 **(Medium)** **(2)**

For Disease

0/1 + 0/1.5 + 0/2 + 0.5/2.5 + 1/3 + 0.5/3.5 + 0/4**(Medium)** **(3)**

Now we are having three equations i.e. (1), (2) ,(3) showing the fuzzy final form after applying genetic algorithm to the low range of diastolic and systolic pressure. Next step is to find relation matrix. Hence we apply mamdani rule to find a relation matrix with the help of which we can find where the disease lies.

Cylindrical Extension of Diastolic BP. TABLE 1

XY/Z	2	2.5	3	3.5
70/110	0	0	0	0
70/115	0	0	0	0
70/120	0	0	0	0
70/125	0	0	0	0
75/110	0.5	0.5	0.5	0.5
75/115	0.5	0.5	0.5	0.5
75/120	0.5	0.5	0.5	0.5
75/125	0.5	0.5	0.5	0.5
80/110	1	1	1	1
80/115	1	1	1	1
80/120	1	1	1	1
80/125	1	1	1	1
85/110	0.5	0.5	0.5	0.5

85/115	0.5	0.5	0.5	0.5
85/120	0.5	0.5	0.5	0.5
85/125	0.5	0.5	0.5	0.5

Cylindrical Extension of Systolic BP. TABLE 2

XY/Z	2	2.5	3	3.5
70/110	0	0	0	0
70/115	0.5	0.5	0.5	0.5
70/120	1	1	1	1
70/125	0.5	0.5	0.5	0.5
75/110	0	0	0	0
75/115	0.5	0.5	0.5	0.5
75/120	1	1	1	1
75/125	0.5	0.5	0.5	0.5
80/110	0	0	0	0
80/115	0.5	0.5	0.5	0.5
80/120	1	1	1	1
80/125	0.5	0.5	0.5	0.5
85/110	0	0	0	0
85/115	0.5	0.5	0.5	0.5
85/120	1	1	1	1
85/125	0.5	0.5	0.5	0.5

Cylindrical Extension of Disease TABLE 3

XY/Z	2	2.5	3	3.5
70/110	0	0.5	1	0.5
70/115	0	0.5	1	0.5
70/120	0	0.5	1	0.5
70/125	0	0.5	1	0.5
75/110	0	0.5	1	0.5
75/115	0	0.5	1	0.5
75/120	0	0.5	1	0.5
75/125	0	0.5	1	0.5
80/110	0	0.5	1	0.5
80/115	0	0.5	1	0.5
80/120	0	0.5	1	0.5
80/125	0	0.5	1	0.5

XY/Z	2	2.5	3	3.5
85/110	0	0.5	1	0.5
85/115	0	0.5	1	0.5
85/120	0	0.5	1	0.5
85/125	0	0.5	1	0.5

Minimization of Tables 1,2,3 gives a relation matrix R

XY/Z	2	2.5	3	3.5
70/110	A1	A2	A3	A4
70/115	A5	A6	A7	A8
70/120	A9	A10	A11	A12
70/125	A13	A14	A15	A16
75/110	A17	A18	A19	A20
75/115	A21	A22	A23	A24
75/120	A25	A26	A217	A28
75/125	A29	A30	A31	A32
80/110	A33	A34	A35	A36
80/115	A37	A38	A39	A40
80/120	A41	A42	A43	A44
80/125	A45	A46	A47	A48
85/110	A49	A50	A51	A52
85/115	A53	A54	A55	A56
85/120	A57	A58	A59	A60
85/125	A61	A62	A63	A64

Now let us take one example if suppose BP is 80/110 then we have

Matrix S

XY/Z	2	2.5	3	3.5
70/110	0	0	0	0
70/115	0	0	0	0
70/120	0	0	0	0
70/125	0	0	0	0
75/110	0	0	0	0
75/115	0	0	0	0
75/120	0	0	0	0
75/125	0	0	0	0
80/110	1	1	1	1

80/115	0	0	0	0
80/120	0	0	0	0
80/125	0	0	0	0
85/110	0	0	0	0
85/115	0	0	0	0
85/120	0	0	0	0
85/125	0	0	0	0

Minimization of R and S gives

XY/Z	2	2.5	3	3.5
70/110	0	0	0	0
70/115	0	0	0	0
70/120	0	0	0	0
70/125	0	0	0	0
75/110	0	0	0	0
75/115	0	0	0	0
75/120	0	0	0	0
75/125	0	0	0	0
80/110	A33	A34	A35	A36
80/115	0	0	0	0
80/120	0	0	0	0
80/125	0	0	0	0
85/110	0	0	0	0
85/115	0	0	0	0
85/120	0	0	0	0
85/125	0	0	0	0

Projection on disease axis is A33 A34 A35 A36

Fuzzy set = A33/2 + A34/2.5 + A35/3 + A36/3.5

Defuzzified value of disease

$$Df = \frac{(A33*2) + (A34*2.5) + (A35*3) + (A36*3.5)}{A33+A34+A35+A36}$$

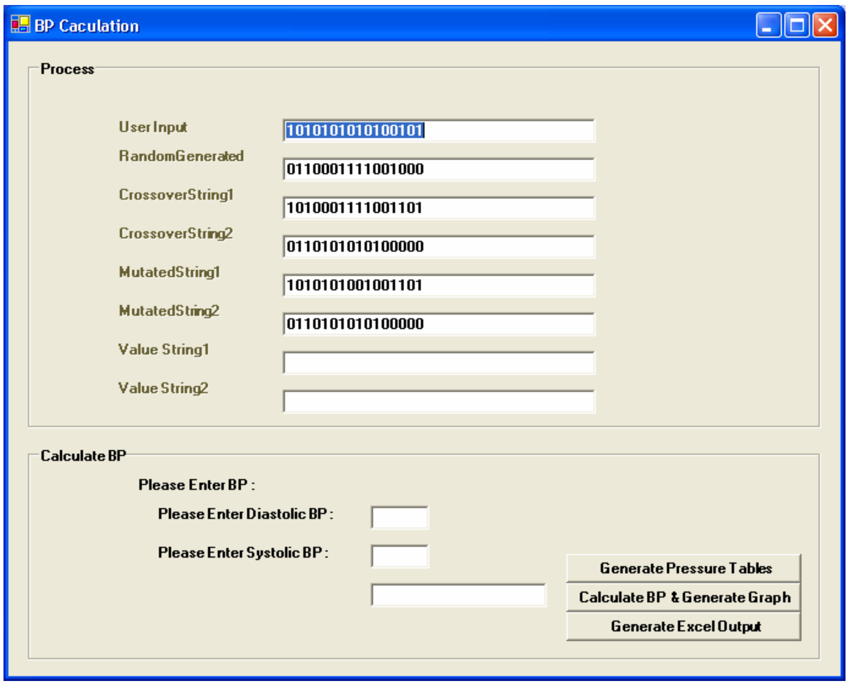
= let value is B

This value should lie in the medium range of disease i.e. 2 to 4

Chapter VI
Result & Conclusion

6.1 Result

The result shows following output graphs, which shows the range of disease i.e. whether the disease is High, Medium or Low depending upon the input values of BP given.


BP Caculation

Process

User Input	1010101010100101
RandomGenerated	0110001111001000
CrossoverString1	1010001111001101
CrossoverString2	0110101010100000
MutatedString1	1010101001001101
MutatedString2	0110101010100000
Value String1	
Value String2	

Calculate BP

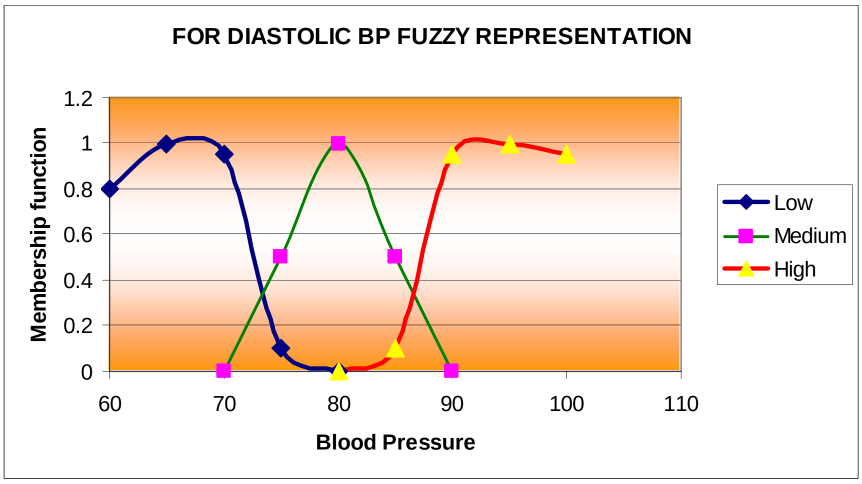
Please Enter BP :

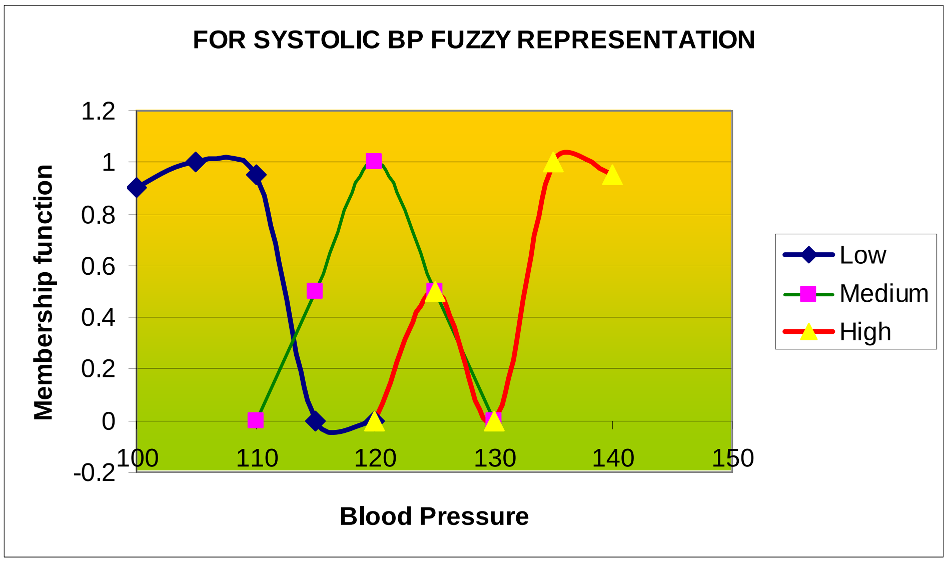
Please Enter Diastolic BP :

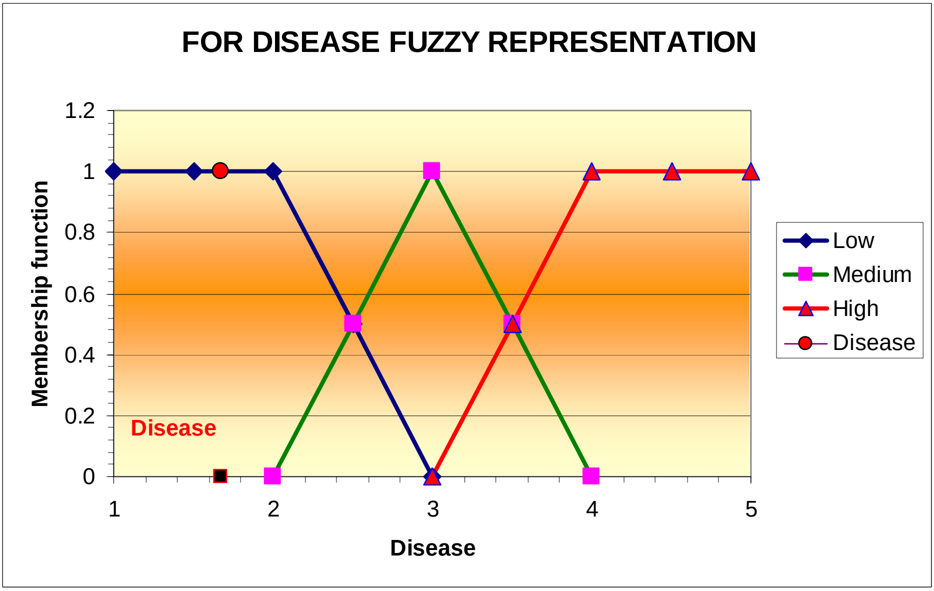
Please Enter Systolic BP :

Input data needed to be fed in above dialog box.

Output graphs







6.2 Conclusion

This project has presented a preliminary study on principal and potential application of Fuzzy Geno systems in the field of medical sciences. Here we have monitored the range of blood pressure fed as an input data to find that whether the BP is low, high or medium. The main aim of monitoring BP is to get an optimized value and the same value can be transferred through telemedicine technique to a doctor sitting at a far place so that he can prescribe a medicine according to the range of the disease a patient is having. In this project we see that the output value gives only the range mentioned as Low, Medium, and High but the value of the disease obtained is not so highly precise. For that we are having a further scope to exactly find the degree of disease either in low, medium or high groups by using a different fuzzy membership representation. By increasing the triangular membership functions or by using other shapes of membership functions we can modify the output obtained.

Appendix
Program Involved

```
<VisualStudioProject>
    <VisualBasic
        ProjectType = "Local"
        ProductVersion = "7.10.3077"
        SchemaVersion = "2.0"
        ProjectGuid = "{92DF8663-8628-4E34-86A6-71646290572B}"
    >
        <Build>
            <Settings
                ApplicationIcon = ""
                AssemblyKeyContainerName = ""
                AssemblyName = "BinaryBP"
                AssemblyOriginatorKeyFile = ""
                AssemblyOriginatorKeyMode = "None"
                DefaultClientScript = "JScript"
                DefaultHTMLPageLayout = "Grid"
                DefaultTargetSchema = "IE50"
                DelaySign = "false"
                OutputType = "WinExe"
                OptionCompare = "Binary"
                OptionExplicit = "On"
                OptionStrict = "Off"
                RootNamespace = "BinaryBP"
                StartupObject = "BinaryBP.Form1"
            >
                <Config
                    Name = "Debug"
                    BaseAddress = "285212672"
                    ConfigurationOverrideFile = ""
                    DefineConstants = ""
                    DefineDebug = "true"
                    DefineTrace = "true"
                    DebugSymbols = "true"
                    IncrementalBuild = "true"
                    Optimize = "false"
                    OutputPath = "bin\"
                    RegisterForComInterop = "false"
                    RemoveIntegerChecks = "false"
                    TreatWarningsAsErrors = "false"
                    WarningLevel = "1"
                />
                <Config
                    Name = "Release"
                    BaseAddress = "285212672"
                    ConfigurationOverrideFile = ""
                    DefineConstants = ""
                    DefineDebug = "false"
                    DefineTrace = "true"
                    DebugSymbols = "false"
                    IncrementalBuild = "false"
                    Optimize = "true"
                    OutputPath = "bin\"
                    RegisterForComInterop = "false"
                    RemoveIntegerChecks = "false"
                    TreatWarningsAsErrors = "false"
                    WarningLevel = "1"
                />
            </Settings>
            <References>
                <Reference
                    Name = "System"
                    AssemblyName = "System"
                />
                <Reference
                    Name = "System.Data"
                    AssemblyName = "System.Data"
                />
                <Reference
                    Name = "System.Drawing"
                    AssemblyName = "System.Drawing"
                />
                <Reference
                    Name = "System.Windows.Forms"
                    AssemblyName = "System.Windows.Forms"
                />
                <Reference
                    Name = "System.XML"
                    AssemblyName = "System.Xml"
                />
                <Reference
                    Name = "VBIDE"
                    Guid = "{0002E157-0000-0000-C000-000000000046}"
                    VersionMajor = "5"
                    VersionMinor = "3"
                    Lcid = "0"
                    WrapperTool = "tlbimp"
                />
                <Reference
                    Name = "Microsoft.Office.Core"
                    Guid = "{2DF8D04C-5BFA-101B-BDE5-00AA0044DE52}"
                    VersionMajor = "2"
                    VersionMinor = "2"
                    Lcid = "0"
                    WrapperTool = "tlbimp"
                />
                <Reference
                    Name = "stdole"
                    Guid = "{00020430-0000-0000-C000-000000000046}"
                    VersionMajor = "2"
```

```
                VersionMinor = "0"
                Lcid = "0"
                WrapperTool = "primary"
            />
            <Reference
                Name = "Excel"
                Guid = "{00020813-0000-0000-C000-000000000046}"
                VersionMajor = "1"
                VersionMinor = "4"
                Lcid = "0"
                WrapperTool = "tlbimp"
            />
        </References>
        <Imports>
            <Import Namespace = "Microsoft.VisualBasic" />
            <Import Namespace = "System" />
            <Import Namespace = "System.Collections" />
            <Import Namespace = "System.Data" />
            <Import Namespace = "System.Drawing" />
            <Import Namespace = "System.Diagnostics" />
            <Import Namespace = "System.Windows.Forms" />
        </Imports>
    </Build>
    <Files>
        <Include>
            <File
                RelPath = "AssemblyInfo.vb"
                SubType = "Code"
                BuildAction = "Compile"
            />
            <File
                RelPath = "Form1.vb"
                SubType = "Form"
                BuildAction = "Compile"
            />
            <File
                RelPath = "Form1.resx"
                DependentUpon = "Form1.vb"
                BuildAction = "EmbeddedResource"
            />
        </Include>
    </Files>
  </VisualBasic>
</VisualStudioProject>
Imports Excel
Imports System.IO
Imports System.Runtime.InteropServices

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 TextBox1 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents TextBox4 As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents Label6 As System.Windows.Forms.Label
    Friend WithEvents TextBox5 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox6 As System.Windows.Forms.TextBox
    Friend WithEvents Label7 As System.Windows.Forms.Label
    Friend WithEvents Label8 As System.Windows.Forms.Label
    Friend WithEvents TextBox7 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox8 As System.Windows.Forms.TextBox
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
    Friend WithEvents btnBP As System.Windows.Forms.Button
    Friend WithEvents txtUpper As System.Windows.Forms.TextBox
    Friend WithEvents txtLower As System.Windows.Forms.TextBox
    Friend WithEvents txtBP As System.Windows.Forms.TextBox
    Friend WithEvents Label9 As System.Windows.Forms.Label
    Friend WithEvents Label10 As System.Windows.Forms.Label
    Friend WithEvents Label11 As System.Windows.Forms.Label
    Friend WithEvents btnOutput As System.Windows.Forms.Button
    Friend WithEvents Button1 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.TextBox3 = New System.Windows.Forms.TextBox
        Me.Label1 = New System.Windows.Forms.Label
        Me.TextBox4 = New System.Windows.Forms.TextBox
        Me.Label2 = New System.Windows.Forms.Label
        Me.Label3 = New System.Windows.Forms.Label
        Me.Label4 = New System.Windows.Forms.Label
        Me.Label5 = New System.Windows.Forms.Label
        Me.Label6 = New System.Windows.Forms.Label
        Me.TextBox5 = New System.Windows.Forms.TextBox
        Me.TextBox6 = New System.Windows.Forms.TextBox
        Me.Label7 = New System.Windows.Forms.Label
        Me.Label8 = New System.Windows.Forms.Label
        Me.TextBox7 = New System.Windows.Forms.TextBox
        Me.TextBox8 = New System.Windows.Forms.TextBox
        Me.GroupBox1 = New System.Windows.Forms.GroupBox
        Me.GroupBox2 = New System.Windows.Forms.GroupBox
        Me.Button1 = New System.Windows.Forms.Button
        Me.btnOutput = New System.Windows.Forms.Button
        Me.Label11 = New System.Windows.Forms.Label
        Me.Label10 = New System.Windows.Forms.Label
        Me.btnBP = New System.Windows.Forms.Button
        Me.txtUpper = New System.Windows.Forms.TextBox
        Me.txtLower = New System.Windows.Forms.TextBox
        Me.txtBP = New System.Windows.Forms.TextBox
        Me.Label9 = New System.Windows.Forms.Label
        Me.GroupBox1.SuspendLayout()
        Me.GroupBox2.SuspendLayout()
        Me.SuspendLayout()
        '
        'TextBox1
        '
        Me.TextBox1.Location = New System.Drawing.Point(208, 48)
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.Size = New System.Drawing.Size(256, 20)
        Me.TextBox1.TabIndex = 0
        Me.TextBox1.Text = "1010101010100101"
        '
        'TextBox2
        '
        Me.TextBox2.Location = New System.Drawing.Point(208, 80)
        Me.TextBox2.Name = "TextBox2"
        Me.TextBox2.Size = New System.Drawing.Size(256, 20)
        Me.TextBox2.TabIndex = 1
        Me.TextBox2.Text = ""
        '
        'TextBox3
        '
        Me.TextBox3.Location = New System.Drawing.Point(208, 112)
        Me.TextBox3.Name = "TextBox3"
        Me.TextBox3.Size = New System.Drawing.Size(256, 20)
        Me.TextBox3.TabIndex = 3
        Me.TextBox3.Text = ""
        '
        'Label1
        '
        Me.Label1.Enabled = False
        Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
    System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.Location = New System.Drawing.Point(72, 48)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(64, 16)
        Me.Label1.TabIndex = 5
        Me.Label1.Text = "User Input"
        '
        'TextBox4
        '
        Me.TextBox4.Location = New System.Drawing.Point(208, 144)
        Me.TextBox4.Name = "TextBox4"
        Me.TextBox4.Size = New System.Drawing.Size(256, 20)
        Me.TextBox4.TabIndex = 7
        Me.TextBox4.Text = ""
        '
        'Label2
        '
        Me.Label2.Enabled = False
        Me.Label2.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
    System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label2.Location = New System.Drawing.Point(72, 72)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(112, 16)
        Me.Label2.TabIndex = 8
        Me.Label2.Text = "RandomGenerated"
        '
        'Label3
        '
        Me.Label3.Enabled = False
        Me.Label3.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
    System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label3.Location = New System.Drawing.Point(72, 104)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(112, 16)
        Me.Label3.TabIndex = 9
        Me.Label3.Text = "CrossoverString1"
        '
        'Label4
        '
        Me.Label4.Enabled = False
```

```
        Me.Label4.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label4.Location = New System.Drawing.Point(72, 136)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(112, 16)
        Me.Label4.TabIndex = 10
        Me.Label4.Text = "CrossoverString2"
        '
        'Label5
        '
        Me.Label5.Enabled = False
        Me.Label5.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label5.Location = New System.Drawing.Point(72, 200)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(112, 16)
        Me.Label5.TabIndex = 14
        Me.Label5.Text = "MutatedString2"
        '
        'Label6
        '
        Me.Label6.Enabled = False
        Me.Label6.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label6.Location = New System.Drawing.Point(72, 168)
        Me.Label6.Name = "Label6"
        Me.Label6.Size = New System.Drawing.Size(112, 16)
        Me.Label6.TabIndex = 13
        Me.Label6.Text = "MutatedString1"
        '
        'TextBox5
        '
        Me.TextBox5.Location = New System.Drawing.Point(208, 208)
        Me.TextBox5.Name = "TextBox5"
        Me.TextBox5.Size = New System.Drawing.Size(256, 20)
        Me.TextBox5.TabIndex = 12
        Me.TextBox5.Text = ""
        '
        'TextBox6
        '
        Me.TextBox6.Location = New System.Drawing.Point(208, 176)
        Me.TextBox6.Name = "TextBox6"
        Me.TextBox6.Size = New System.Drawing.Size(256, 20)
        Me.TextBox6.TabIndex = 11
        Me.TextBox6.Text = ""
        '
        'Label7
        '
        Me.Label7.Enabled = False
        Me.Label7.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label7.Location = New System.Drawing.Point(72, 264)
        Me.Label7.Name = "Label7"
        Me.Label7.Size = New System.Drawing.Size(112, 16)
        Me.Label7.TabIndex = 19
        Me.Label7.Text = "Value String2"
        '
        'Label8
        '
        Me.Label8.Enabled = False
        Me.Label8.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label8.Location = New System.Drawing.Point(72, 232)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New System.Drawing.Size(112, 16)
        Me.Label8.TabIndex = 18
        Me.Label8.Text = "Value String1"
        '
        'TextBox7
        '
        Me.TextBox7.Location = New System.Drawing.Point(208, 272)
        Me.TextBox7.Name = "TextBox7"
        Me.TextBox7.Size = New System.Drawing.Size(256, 20)
        Me.TextBox7.TabIndex = 17
        Me.TextBox7.Text = ""
        '
        'TextBox8
        '
        Me.TextBox8.Location = New System.Drawing.Point(208, 240)
        Me.TextBox8.Name = "TextBox8"
        Me.TextBox8.Size = New System.Drawing.Size(256, 20)
        Me.TextBox8.TabIndex = 16
        Me.TextBox8.Text = ""
        '
        'GroupBox1
        '
        Me.GroupBox1.Controls.Add(Me.TextBox2)
        Me.GroupBox1.Controls.Add(Me.TextBox3)
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Controls.Add(Me.TextBox4)
        Me.GroupBox1.Controls.Add(Me.Label2)
        Me.GroupBox1.Controls.Add(Me.Label3)
        Me.GroupBox1.Controls.Add(Me.Label4)
        Me.GroupBox1.Controls.Add(Me.Label5)
        Me.GroupBox1.Controls.Add(Me.Label6)
        Me.GroupBox1.Controls.Add(Me.TextBox5)
        Me.GroupBox1.Controls.Add(Me.TextBox6)
        Me.GroupBox1.Controls.Add(Me.Label7)
        Me.GroupBox1.Controls.Add(Me.Label8)
        Me.GroupBox1.Controls.Add(Me.TextBox7)
        Me.GroupBox1.Controls.Add(Me.TextBox8)
        Me.GroupBox1.Controls.Add(Me.TextBox1)
        Me.GroupBox1.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
        Me.GroupBox1.Location = New System.Drawing.Point(16, 16)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(648, 304)
        Me.GroupBox1.TabIndex = 26
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Process"
        '
        'GroupBox2
        '
        Me.GroupBox2.Controls.Add(Me.Button1)
        Me.GroupBox2.Controls.Add(Me.btnOutput)
        Me.GroupBox2.Controls.Add(Me.Label11)
        Me.GroupBox2.Controls.Add(Me.Label10)
        Me.GroupBox2.Controls.Add(Me.btnBP)
        Me.GroupBox2.Controls.Add(Me.txtUpper)
        Me.GroupBox2.Controls.Add(Me.txtLower)
        Me.GroupBox2.Controls.Add(Me.txtBP)
        Me.GroupBox2.Controls.Add(Me.Label9)
        Me.GroupBox2.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.GroupBox2.Location = New System.Drawing.Point(16, 336)
        Me.GroupBox2.Name = "GroupBox2"
        Me.GroupBox2.Size = New System.Drawing.Size(648, 176)
        Me.GroupBox2.TabIndex = 28
        Me.GroupBox2.TabStop = False
        Me.GroupBox2.Text = "Calculate BP"
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(440, 88)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(192, 24)
        Me.Button1.TabIndex = 36
        Me.Button1.Text = "Generate Pressure Tables"
        '
        'btnOutput
        '
        Me.btnOutput.Location = New System.Drawing.Point(440, 136)
        Me.btnOutput.Name = "btnOutput"
        Me.btnOutput.Size = New System.Drawing.Size(192, 24)
        Me.btnOutput.TabIndex = 35
        Me.btnOutput.Text = "Generate Excel Output"
        '
        'Label11
        '
        Me.Label11.Location = New System.Drawing.Point(104, 80)
        Me.Label11.Name = "Label11"
        Me.Label11.Size = New System.Drawing.Size(144, 16)
        Me.Label11.TabIndex = 34
        Me.Label11.Text = "Please Enter Systolic BP :"
        '
        'Label10
```

```
'
Me.Label10.Location = New System.Drawing.Point(104, 48)
Me.Label10.Name = "Label10"
Me.Label10.Size = New System.Drawing.Size(152, 16)
Me.Label10.TabIndex = 33
Me.Label10.Text = "Please Enter Diastolic BP :"
'
'btnBP
'
Me.btnBP.Location = New System.Drawing.Point(440, 112)
Me.btnBP.Name = "btnBP"
Me.btnBP.Size = New System.Drawing.Size(192, 24)
Me.btnBP.TabIndex = 32
Me.btnBP.Text = "Calculate BP  && Generate Graph"
'
'txtUpper
'
Me.txtUpper.Location = New System.Drawing.Point(280, 80)
Me.txtUpper.MaxLength = 3
Me.txtUpper.Name = "txtUpper"
Me.txtUpper.Size = New System.Drawing.Size(48, 20)
Me.txtUpper.TabIndex = 31
Me.txtUpper.Text = ""
'
'txtLower
'
Me.txtLower.Location = New System.Drawing.Point(280, 48)
Me.txtLower.MaxLength = 3
Me.txtLower.Name = "txtLower"
Me.txtLower.Size = New System.Drawing.Size(48, 20)
Me.txtLower.TabIndex = 30
Me.txtLower.Text = ""
'
'txtBP
'
Me.txtBP.Location = New System.Drawing.Point(280, 112)
Me.txtBP.Name = "txtBP"
Me.txtBP.Size = New System.Drawing.Size(144, 20)
Me.txtBP.TabIndex = 29
Me.txtBP.Text = ""
'
'Label9
'
Me.Label9.Location = New System.Drawing.Point(88, 24)
Me.Label9.Name = "Label9"
Me.Label9.Size = New System.Drawing.Size(104, 16)
Me.Label9.TabIndex = 28
Me.Label9.Text = "Please Enter BP :"
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
```

```
        Me.ClientSize = New System.Drawing.Size(680, 525)
        Me.Controls.Add(Me.GroupBox2)
        Me.Controls.Add(Me.GroupBox1)
        Me.Name = "Form1"
        Me.Text = "BP Caculation "
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region
    Public Shared x1, x2, x3, x4 As Double
    Public Shared x11, x12, x13, x14 As Double
    Public Shared y1, y2, y3, y4 As Double
    Public Shared y11, y12, y13, y14 As Double

    Public Shared z1, z2, z3, z4 As Double
    Public Shared z11, z12, z13, z14 As Double
    Public Shared s1, s2, s3, s4 As Double
    Public Shared s11, s12, s13, s14 As Double

    Public Shared DiseaseArray(15, 4) As Double 'Disease Array for LBP
    Public Shared SystolicArray(15, 4) As Double 'Systolic Array for LBP
    Public Shared DiastolicArray(15, 4) As Double 'Diastolic Array for LBP
    Public Shared MinimizeArray(15, 4) As Double 'Minimize Array for LBP

    Public Shared HPDiseaseArray(15, 4) As Double 'Disease Array for HBP
    Public Shared HPSystolicArray(15, 4) As Double 'Systolic Array for HBP
    Public Shared HPDiastolicArray(15, 4) As Double 'Diastolic Array for HBP
    Public Shared HPMinimizeArray(15, 4) As Double 'Minimize Array for HBP

    Public Shared MPDiseaseArray(15, 4) As Double 'Disease Array for MBP
    Public Shared MPSystolicArray(15, 4) As Double 'Systolic Array for MBP
    Public Shared MPDiastolicArray(15, 4) As Double 'Diastolic Array for MBP
    Public Shared MPMinimizeArray(15, 4) As Double 'Minimize Array for MBP
    Public Shared theValue As Double

    Public Shared lowDystolicX As Boolean = False
    Public Shared lowSystolicY As Boolean = False

    Public Shared HighDystolicZ As Boolean = False
    Public Shared HighSystolicS As Boolean = False
    Public Shared callAgain As Boolean = False
    Public Shared Generated As Boolean = False


    Public Sub CreateExcelForm()
        Dim MyExcel As New Excel.Application
        Dim oWorkbooks As Excel.Workbooks = MyExcel.Workbooks
        Dim theWorkbook As Excel.Workbook = oWorkbooks.Add
        Dim oSheet As Excel.Worksheet
Dim xlsTemplateFile As String = "C:\Roma_project_graphs.xls"
Dim i, j As Integer

Dim xlsSaveFile As String = "C:\roma.xls"
'If file already exists, remove it
If File.Exists(xlsSaveFile) Then File.Delete(xlsSaveFile)
Try
    theWorkbook = oWorkbooks.Open(xlsTemplateFile)
    oSheet = MyExcel.ActiveSheet()

    If lowDystolicX = True Then
        oSheet.Cells(3, 2) = x1
        oSheet.Cells(4, 2) = x2
        oSheet.Cells(5, 2) = x3
        oSheet.Cells(6, 2) = x4
    Else
        oSheet.Cells(3, 2) = x11
        oSheet.Cells(4, 2) = x12
        oSheet.Cells(5, 2) = x13
        oSheet.Cells(6, 2) = x14
    End If

    If lowSystolicY = True Then
        oSheet.Cells(3, 3) = y1
        oSheet.Cells(4, 3) = y2
        oSheet.Cells(5, 3) = y3
        oSheet.Cells(6, 3) = y4
    Else
        oSheet.Cells(3, 3) = y11
        oSheet.Cells(4, 3) = y12
        oSheet.Cells(5, 3) = y13
        oSheet.Cells(6, 3) = y14
    End If

    If HighDystolicZ = True Then
        oSheet.Cells(8, 2) = z1
        oSheet.Cells(9, 2) = z2
        oSheet.Cells(10, 2) = z3
        oSheet.Cells(11, 2) = z4
    Else
        oSheet.Cells(8, 2) = z11
        oSheet.Cells(9, 2) = z12
        oSheet.Cells(10, 2) = z13
        oSheet.Cells(11, 2) = z14
    End If

    If HighSystolicS = True Then
        oSheet.Cells(8, 3) = s1
        oSheet.Cells(9, 3) = s2
        oSheet.Cells(10, 3) = s3
        oSheet.Cells(11, 3) = s4
    Else
```

```
            oSheet.Cells(8, 3) = s11
            oSheet.Cells(9, 3) = s12
            oSheet.Cells(10, 3) = s13
            oSheet.Cells(11, 3) = s14
        End If
        oSheet.Cells(12, 2) = theValue
        'theWorkbook.Save()
        theWorkbook.SaveAs(xlsSaveFile)
        theWorkbook.Close()
        MyExcel.Quit()

        Marshal.ReleaseComObject(oSheet)
        Marshal.ReleaseComObject(oWorkbooks)
        Marshal.ReleaseComObject(theWorkbook)
        Marshal.ReleaseComObject(MyExcel)

        GC.Collect()
        GC.WaitForPendingFinalizers()
    Catch ex As Exception
        Throw New Exception("Error creating Excel spreadsheet:" & ex.Message.ToString())
    Finally
        oSheet = Nothing
        theWorkbook = Nothing
        oWorkbooks = Nothing
        MyExcel = Nothing
    End Try

    'Delete file from server
    'If File.Exists(xlsSaveFile) Then File.Delete(xlsSaveFile)
  End Sub
  Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    x1 = x2 = x3 = x4 = 0.0
    x11 = x12 = x13 = x14 = 0
    y1 = y2 = y3 = y4 = 0
    y11 = y12 = y1 = y14 = 0
    z1 = z2 = z3 = z4 = 0
    z11 = z12 = z13 = z14 = 0
    s1 = s2 = s3 = s4 = 0
    s11 = s12 = s13 = s14 = 0
    'CreateExcelForm()
    CallAll()
  End Sub

  Private Sub CallAll()
    Dim i As Integer
    For i = 0 To 0
        Randomclick()
        BinaryChange()
        crossover()
        Mutate()
        ApplyFormula()
        If callAgain = True Then
            i = -1
        Else
            i = 1
        End If
    Next
    CallSystolic()
End Sub

Private Sub CallSystolic()
    Dim i As Integer
    For i = 0 To 0
        Randomclick()
        BinaryChange()
        crossover()
        Mutate()
        ApplySystolicFormula()
        If callAgain = True Then
            i = -1
        Else
            i = 1
        End If
    Next
End Sub

Private Sub ApplyFormula()
    Dim ans As Double
    Dim ans1 As Double
    Dim toTry As Double
    Dim first As Boolean
    Dim second As Boolean
    first = False
    second = False
    x1 = Bin2Dec(Mid(TextBox6.Text, 1, 4)) / 10
    'MsgBox(x1)
    x2 = Bin2Dec(Mid(TextBox6.Text, 5, 4)) / 10
    'MsgBox(x2)
    x3 = Bin2Dec(Mid(TextBox6.Text, 9, 4)) / 10
    'MsgBox(x3)
    x4 = Bin2Dec(Mid(TextBox6.Text, 13, 4)) / 10
    'MsgBox(x4)
    ans = ((x1 * 60) + (x2 * 65) + (x3 * 70) + (x4 * 75)) / (x1 + x2 + x3 + x4)

    x11 = Bin2Dec(Mid(TextBox5.Text, 1, 4)) / 10
    'MsgBox(x1)
    x12 = Bin2Dec(Mid(TextBox5.Text, 5, 4)) / 10
    'MsgBox(x2)
    x13 = Bin2Dec(Mid(TextBox5.Text, 9, 4)) / 10
    'MsgBox(x3)
    x14 = Bin2Dec(Mid(TextBox5.Text, 13, 4)) / 10
    'MsgBox(x4)
    ans1 = ((x11 * 60) + (x12 * 65) + (x13 * 70) + (x14 * 75)) / (x11 + x12 + x13 + x14)
    If ans >= 65 And ans < 66 Then
        first = True
    ElseIf ans1 >= 65 And ans1 < 66 Then
        second = True
    End If

    If first = False And second = False Then
        'MsgBox("repeat")
        callAgain = True
        Exit Sub
    Else
        callAgain = False
        'MsgBox("First String of Diastolic :" & ans)
        'MsgBox("Second String of Diastolic :" & ans1)
        'MsgBox("Diastolic done")
        If first = True Then
            'MsgBox("X1 " & x1 & "X2 " & x2 & "X3 " & x3 & "X4 " & x4)
            lowDystolicX = True
            CreateDiastolicArray(x1, x2, x3, x4)
        Else
            'MsgBox("X11 " & x11 & "X12 " & x12 & "X13 " & x13 & "X14 " & x14)
            lowDystolicX = False
            CreateDiastolicArray(x11, x12, x13, x14)
        End If
    End If
End Sub

Private Sub ApplySystolicFormula()
    Dim ans As Double
    Dim ans1 As Double
    Dim toTry As Double
    Dim first As Boolean
    Dim second As Boolean
    first = False
    second = False
    y1 = Bin2Dec(Mid(TextBox6.Text, 1, 4)) / 10
    'MsgBox(Bin2Dec(Mid(TextBox6.Text, 5, 4)))
    y2 = Bin2Dec(Mid(TextBox6.Text, 5, 4)) / 10
    y3 = Bin2Dec(Mid(TextBox6.Text, 9, 4)) / 10
    y4 = Bin2Dec(Mid(TextBox6.Text, 13, 4)) / 10
    ans = ((y1 * 100) + (y2 * 105) + (y3 * 110) + (y4 * 115)) / (y1 + y2 + y3 + y4)

    y11 = Bin2Dec(Mid(TextBox5.Text, 1, 4)) / 10
    'MsgBox(Bin2Dec(Mid(TextBox5.Text, 5, 4)))
    y12 = Bin2Dec(Mid(TextBox5.Text, 5, 4)) / 10
    y13 = Bin2Dec(Mid(TextBox5.Text, 9, 4)) / 10
    y14 = Bin2Dec(Mid(TextBox5.Text, 13, 4)) / 10
    ans1 = ((y11 * 100) + (y12 * 105) + (y13 * 110) + (y14 * 115)) / (y11 + y12 + y13 + y14)

    If ans >= 105 And ans < 106 Then
```

```
        first = True
    ElseIf ans1 >= 105 And ans1 < 106 Then
        second = True
    End If

    If first = False And second = False Then
        'CallSystolic()
        callAgain = True
        Exit Sub
    Else
        callAgain = False
        'MsgBox("First String of Systolic :" & ans)
        'MsgBox("Second String of Systolic :" & ans1)
        'MsgBox("Systolic done")
        If first = True Then
            'MsgBox("Y1 " & y1 & "Y2 " & y2 & "Y3 " & y3 & "Y4 " & y4)
            lowSystolicY = True
            CreateSystolicArray(y1, y2, y3, y4)
        Else
            'MsgBox("Y11 " & y11 & "Y12 " & y12 & "Y13 " & y13 & "Y14 " & y14)
            lowSystolicY = False
            CreateSystolicArray(y11, y12, y13, y14)
        End If
        'CallSystolic()
    End If
End Sub


Private Sub ApplyHPSystolicFormula()
    Dim ans As Double
    Dim ans1 As Double
    Dim toTry As Double
    Dim first As Boolean
    Dim second As Boolean
    first = False
    second = False
    s1 = Bin2Dec(Mid(TextBox6.Text, 1, 4)) / 10
    'MsgBox(Bin2Dec(Mid(TextBox6.Text, 5, 4)))
    s2 = Bin2Dec(Mid(TextBox6.Text, 5, 4)) / 10
    s3 = Bin2Dec(Mid(TextBox6.Text, 9, 4)) / 10
    s4 = Bin2Dec(Mid(TextBox6.Text, 13, 4)) / 10
    ans = ((s1 * 125) + (s2 * 130) + (s3 * 135) + (s4 * 140)) / (s1 + s2 + s3 + s4)

    s11 = Bin2Dec(Mid(TextBox5.Text, 1, 4)) / 10
    'MsgBox(Bin2Dec(Mid(TextBox5.Text, 5, 4)))
    s12 = Bin2Dec(Mid(TextBox5.Text, 5, 4)) / 10
    s13 = Bin2Dec(Mid(TextBox5.Text, 9, 4)) / 10
    s14 = Bin2Dec(Mid(TextBox5.Text, 13, 4)) / 10
    ans1 = ((s11 * 125) + (s12 * 130) + (s13 * 135) + (s14 * 140)) / (s11 + s12 + s13 + s14)

    If ans >= 135 And ans < 136 Then
        first = True
        ElseIf ans1 >= 135 And ans1 < 136 Then
            second = True
        End If

        If first = False And second = False Then
            'CallHPSystolic()
            callAgain = True
            Exit Sub
        Else
            callAgain = False
            'MsgBox("First String of Systolic :" & ans)
            'MsgBox("Second String of Systolic :" & ans1)
            'MsgBox("Systolic done")
            If first = True Then
                'MsgBox("Y1 " & y1 & "Y2 " & y2 & "Y3 " & y3 & "Y4 " & y4)
                HighSystolicS = True
                CreateHPSystolicArray(s1, s2, s3, s4)
                CreateHPDiseaseArray()
                CreateHPMinimizeArray()
            ElseIf second = True Then
                'MsgBox("Y11 " & y11 & "Y12 " & y12 & "Y13 " & y13 & "Y14 " & y14)
                HighSystolicS = False
                CreateHPSystolicArray(s11, s12, s13, s14)
                CreateHPDiseaseArray()
                CreateHPMinimizeArray()
            End If
            'CallSystolic()
        End If
    End Sub

    Private Sub CreateSystolicArray(ByVal a As Double, ByVal b As Double, ByVal c As
Double, ByVal d As Double)
        Dim i, j As Integer
        For i = 0 To 15
            For j = 0 To 4
                If j = 0 Then
                    If i = 0 Then
                        SystolicArray(i, j) = 60 / 100
                    ElseIf i = 1 Then
                        SystolicArray(i, j) = 60 / 105
                    ElseIf i = 2 Then
                        SystolicArray(i, j) = 60 / 110
                    ElseIf i = 3 Then
                        SystolicArray(i, j) = 60 / 115
                    ElseIf i = 4 Then
                        SystolicArray(i, j) = 65 / 100
                    ElseIf i = 5 Then
                        SystolicArray(i, j) = 65 / 105
                    ElseIf i = 6 Then
                        SystolicArray(i, j) = 65 / 110
                    ElseIf i = 7 Then
                        SystolicArray(i, j) = 65 / 115
                    ElseIf i = 8 Then
                        SystolicArray(i, j) = 70 / 100
                    ElseIf i = 9 Then
                        SystolicArray(i, j) = 70 / 105
                    ElseIf i = 10 Then
                        SystolicArray(i, j) = 70 / 110
                    ElseIf i = 11 Then
                        SystolicArray(i, j) = 70 / 115
                    ElseIf i = 12 Then
                        SystolicArray(i, j) = 75 / 100
                    ElseIf i = 13 Then
                        SystolicArray(i, j) = 75 / 105
                    ElseIf i = 14 Then
                        SystolicArray(i, j) = 75 / 110
                    ElseIf i = 15 Then
                        SystolicArray(i, j) = 75 / 115
                    End If
                Else
                    If i = 0 Or i = 4 Or i = 8 Or i = 12 Then
                        SystolicArray(i, j) = a
                    ElseIf i = 1 Or i = 5 Or i = 9 Or i = 13 Then
                        SystolicArray(i, j) = b
                    ElseIf i = 2 Or i = 6 Or i = 10 Or i = 14 Then
                        SystolicArray(i, j) = c
                    ElseIf i = 3 Or i = 7 Or i = 11 Or i = 15 Then
                        SystolicArray(i, j) = d
                    End If
                End If
            Next
        Next
        CreateDiseaseArray()
    End Sub

    Private Sub CreateDiastolicArray(ByVal a As Double, ByVal b As Double, ByVal c As
Double, ByVal d As Double)
        Dim i, j As Integer
        For i = 0 To 15
            For j = 0 To 4
                If j = 0 Then
                    If i = 0 Then
                        DiastolicArray(i, j) = 60 / 100
                    ElseIf i = 1 Then
                        DiastolicArray(i, j) = 60 / 105
                    ElseIf i = 2 Then
                        DiastolicArray(i, j) = 60 / 110
                    ElseIf i = 3 Then
                        DiastolicArray(i, j) = 60 / 115
                    ElseIf i = 4 Then
                        DiastolicArray(i, j) = 65 / 100
                    ElseIf i = 5 Then
                        DiastolicArray(i, j) = 65 / 105
                    ElseIf i = 6 Then
                        DiastolicArray(i, j) = 65 / 110
                    ElseIf i = 7 Then
                        DiastolicArray(i, j) = 65 / 115
                    ElseIf i = 8 Then
                        DiastolicArray(i, j) = 70 / 100
                    ElseIf i = 9 Then
                        DiastolicArray(i, j) = 70 / 105
                    ElseIf i = 10 Then
                        DiastolicArray(i, j) = 70 / 110
                    ElseIf i = 11 Then
                        DiastolicArray(i, j) = 70 / 115
                    ElseIf i = 12 Then
                        DiastolicArray(i, j) = 75 / 100
                    ElseIf i = 13 Then
                        DiastolicArray(i, j) = 75 / 105
                    ElseIf i = 14 Then
                        DiastolicArray(i, j) = 75 / 110
                    ElseIf i = 15 Then
                        DiastolicArray(i, j) = 75 / 115
                    End If
                Else
                    If i <= 3 Then
                        DiastolicArray(i, j) = a
                    ElseIf i >= 4 And i <= 7 Then
                        DiastolicArray(i, j) = b
                    ElseIf i >= 8 And i <= 11 Then
                        DiastolicArray(i, j) = c
                    Else
                        DiastolicArray(i, j) = d
                    End If
                End If
            Next
        Next
    End Sub

    Private Sub CreateMinimizeArray()
        Dim i, j As Integer
        For i = 0 To 15
            For j = 0 To 4
                If j = 0 Then
                    If i = 0 Then
                        MinimizeArray(i, j) = 60 / 100
                    ElseIf i = 1 Then
                        MinimizeArray(i, j) = 60 / 105
                    ElseIf i = 2 Then
                        MinimizeArray(i, j) = 60 / 110
                    ElseIf i = 3 Then
                        MinimizeArray(i, j) = 60 / 115
                    ElseIf i = 4 Then
                        MinimizeArray(i, j) = 65 / 100
                    ElseIf i = 5 Then
                        MinimizeArray(i, j) = 65 / 105
```

```
                    ElseIf i = 6 Then
                        MinimizeArray(i, j) = 65 / 110
                    ElseIf i = 7 Then
                        MinimizeArray(i, j) = 65 / 115
                    ElseIf i = 8 Then
                        MinimizeArray(i, j) = 70 / 100
                    ElseIf i = 9 Then
                        MinimizeArray(i, j) = 70 / 105
                    ElseIf i = 10 Then
                        MinimizeArray(i, j) = 70 / 110
                    ElseIf i = 11 Then
                        MinimizeArray(i, j) = 70 / 115
                    ElseIf i = 12 Then
                        MinimizeArray(i, j) = 75 / 100
                    ElseIf i = 13 Then
                        MinimizeArray(i, j) = 75 / 105
                    ElseIf i = 14 Then
                        MinimizeArray(i, j) = 75 / 110
                    ElseIf i = 15 Then
                        MinimizeArray(i, j) = 75 / 115
                    End If
                Else
                    If (SystolicArray(i, j) <= DiastolicArray(i, j)) And SystolicArray(i, j) <=
DiseaseArray(i, j) Then
                        MinimizeArray(i, j) = SystolicArray(i, j)
                        'MsgBox("systolic")
                    ElseIf (DiastolicArray(i, j) <= SystolicArray(i, j)) And DiastolicArray(i, j) <=
DiseaseArray(i, j) Then
                        MinimizeArray(i, j) = DiastolicArray(i, j)
                        'MsgBox("Dystolic")
                    ElseIf (DiseaseArray(i, j) <= DiastolicArray(i, j)) And DiseaseArray(i, j) <=
SystolicArray(i, j) Then
                        MinimizeArray(i, j) = DiseaseArray(i, j)
                        'MsgBox("Disease")
                    End If
                End If
            Next
        Next
    End Sub


    Private Sub CreateDiseaseArray()
        Dim i, j As Integer
        For i = 0 To 15
            For j = 0 To 4
                If j = 0 Then
                    If i = 0 Then
                        DiseaseArray(i, j) = 60 / 100
                    ElseIf i = 1 Then
                        DiseaseArray(i, j) = 60 / 105
                    ElseIf i = 2 Then
                        DiseaseArray(i, j) = 60 / 110
            ElseIf i = 3 Then
               DiseaseArray(i, j) = 60 / 115
            ElseIf i = 4 Then
               DiseaseArray(i, j) = 65 / 100
            ElseIf i = 5 Then
               DiseaseArray(i, j) = 65 / 105
            ElseIf i = 6 Then
               DiseaseArray(i, j) = 65 / 110
            ElseIf i = 7 Then
               DiseaseArray(i, j) = 65 / 115
            ElseIf i = 8 Then
               DiseaseArray(i, j) = 70 / 100
            ElseIf i = 9 Then
               DiseaseArray(i, j) = 70 / 105
            ElseIf i = 10 Then
               DiseaseArray(i, j) = 70 / 110
            ElseIf i = 11 Then
               DiseaseArray(i, j) = 70 / 115
            ElseIf i = 12 Then
               DiseaseArray(i, j) = 75 / 100
            ElseIf i = 13 Then
               DiseaseArray(i, j) = 75 / 105
            ElseIf i = 14 Then
               DiseaseArray(i, j) = 75 / 110
            ElseIf i = 15 Then
               DiseaseArray(i, j) = 75 / 115
            End If
         ElseIf j = 4 Then
            DiseaseArray(i, j) = 0.5
         Else
            DiseaseArray(i, j) = 1.0
         End If
      Next
   Next
   CreateMinimizeArray()
End Sub

Private Sub StartHPProcess()
   Dim i As Integer
   For i = 0 To 0
      TextBox1.Text = "0101101010101010"
      Randomclick()
      BinaryChange()
      crossover()
      MutateHP()
      ApplyHPFormula()
      If callAgain = True Then
         i = -1
      Else
         i = 1
      End If
   Next
```

```
    CallHPSystolic()
End Sub

Private Sub CallHPSystolic()
    Dim i As Integer
    For i = 0 To 0
        Randomclick()
        BinaryChange()
        crossover()
        MutateHP()
        ApplyHPSystolicFormula()
        If callAgain = True Then
            i = -1
        Else
            i = 1
        End If
    Next
    'CreateHPDiseaseArray()
    'CreateHPMinimizeArray()
End Sub


Private Sub ApplyHPFormula()
    Dim ans As Double
    Dim ans1 As Double
    Dim toTry As Double
    Dim first As Boolean
    Dim second As Boolean
    first = False
    second = False
    z1 = Bin2Dec(Mid(TextBox6.Text, 1, 4)) / 10
    z2 = Bin2Dec(Mid(TextBox6.Text, 5, 4)) / 10
    z3 = Bin2Dec(Mid(TextBox6.Text, 9, 4)) / 10
    z4 = Bin2Dec(Mid(TextBox6.Text, 13, 4)) / 10
    ans = ((z1 * 85) + (z2 * 90) + (z3 * 95) + (z4 * 100)) / (z1 + z2 + z3 + y4)

    z11 = Bin2Dec(Mid(TextBox5.Text, 1, 4)) / 10
    z12 = Bin2Dec(Mid(TextBox5.Text, 5, 4)) / 10
    z13 = Bin2Dec(Mid(TextBox5.Text, 9, 4)) / 10
    z14 = Bin2Dec(Mid(TextBox5.Text, 13, 4)) / 10
    ans1 = ((z11 * 85) + (z12 * 90) + (z13 * 95) + (z14 * 100)) / (z11 + z12 + z13 + z14)
    first = False
    second = False
    If ans >= 95 And ans < 96 Then
        first = True
    ElseIf ans1 >= 95 And ans1 < 96 Then
        second = True
    End If

    If first = False And second = False Then
        callAgain = True
        Exit Sub
    Else
        callAgain = False
        'MsgBox("High Diastolic done")
        If first = True Then
            'MsgBox("Z1 " & z1 & "Z2 " & z2 & "Z3 " & z3 & "Z4 " & z4)
            HighDystolicZ = True
            CreateHPDiastolicArray(z1, z2, z3, z4)
        ElseIf second = True Then
            'MsgBox("Y11 " & y11 & "Y12 " & y12 & "Y13 " & y13 & "Y14 " & y14)
            HighDystolicZ = False
            CreateHPDiastolicArray(z11, z12, z13, z14)
        End If

    End If
End Sub



Private Sub CreateHPDiastolicArray(ByVal a As Double, ByVal b As Double, ByVal c As Double, ByVal d As Double)
    Dim i, j As Integer
    For i = 0 To 15
        For j = 0 To 4
            If j = 0 Then
                If i = 0 Then
                    HPDiastolicArray(i, j) = 85 / 125
                ElseIf i = 1 Then
                    HPDiastolicArray(i, j) = 85 / 130
                ElseIf i = 2 Then
                    HPDiastolicArray(i, j) = 85 / 135
                ElseIf i = 3 Then
                    HPDiastolicArray(i, j) = 85 / 140
                ElseIf i = 4 Then
                    HPDiastolicArray(i, j) = 90 / 125
                ElseIf i = 5 Then
                    HPDiastolicArray(i, j) = 90 / 130
                ElseIf i = 6 Then
                    HPDiastolicArray(i, j) = 90 / 135
                ElseIf i = 7 Then
                    HPDiastolicArray(i, j) = 90 / 140
                ElseIf i = 8 Then
                    HPDiastolicArray(i, j) = 95 / 125
                ElseIf i = 9 Then
                    HPDiastolicArray(i, j) = 95 / 130
                ElseIf i = 10 Then
                    HPDiastolicArray(i, j) = 95 / 135
                ElseIf i = 11 Then
                    HPDiastolicArray(i, j) = 95 / 140
                ElseIf i = 12 Then
                    HPDiastolicArray(i, j) = 100 / 125
                ElseIf i = 13 Then
                    HPDiastolicArray(i, j) = 100 / 130
                ElseIf i = 14 Then
                    HPDiastolicArray(i, j) = 100 / 135
                ElseIf i = 15 Then
                    HPDiastolicArray(i, j) = 100 / 140
                End If
            Else
                If i <= 3 Then
                    HPDiastolicArray(i, j) = a
                ElseIf i >= 4 And i <= 7 Then
                    HPDiastolicArray(i, j) = b
                ElseIf i >= 8 And i <= 11 Then
                    HPDiastolicArray(i, j) = c
                Else
                    HPDiastolicArray(i, j) = d
                End If
            End If
        Next
    Next
End Sub


Private Sub CreateHPSystolicArray(ByVal a As Double, ByVal b As Double, ByVal c As
Double, ByVal d As Double)
    Dim i, j As Integer
    For i = 0 To 15
        For j = 0 To 4
            If j = 0 Then
                If i = 0 Then
                    HPSystolicArray(i, j) = 85 / 125
                ElseIf i = 1 Then
                    HPSystolicArray(i, j) = 85 / 130
                ElseIf i = 2 Then
                    HPSystolicArray(i, j) = 85 / 135
                ElseIf i = 3 Then
                    HPSystolicArray(i, j) = 85 / 140
                ElseIf i = 4 Then
                    HPSystolicArray(i, j) = 90 / 125
                ElseIf i = 5 Then
                    HPSystolicArray(i, j) = 90 / 130
                ElseIf i = 6 Then
                    HPSystolicArray(i, j) = 90 / 135
                ElseIf i = 7 Then
                    HPSystolicArray(i, j) = 90 / 140
                ElseIf i = 8 Then
                    HPSystolicArray(i, j) = 95 / 125
                ElseIf i = 9 Then
                    HPSystolicArray(i, j) = 95 / 130
                ElseIf i = 10 Then
                    HPSystolicArray(i, j) = 95 / 135
                ElseIf i = 11 Then
                    HPSystolicArray(i, j) = 95 / 140
                ElseIf i = 12 Then
```

```
                HPSystolicArray(i, j) = 100 / 125
            ElseIf i = 13 Then
                HPSystolicArray(i, j) = 100 / 130
            ElseIf i = 14 Then
                HPSystolicArray(i, j) = 100 / 135
            ElseIf i = 15 Then
                HPSystolicArray(i, j) = 100 / 140
            End If
        Else
            If i = 0 Or i = 4 Or i = 8 Or i = 12 Then
                HPSystolicArray(i, j) = a
            ElseIf i = 1 Or i = 5 Or i = 9 Or i = 13 Then
                HPSystolicArray(i, j) = b
            ElseIf i = 2 Or i = 6 Or i = 10 Or i = 14 Then
                HPSystolicArray(i, j) = c
            ElseIf i = 3 Or i = 7 Or i = 11 Or i = 15 Then
                HPSystolicArray(i, j) = d
            End If
        End If
    Next
Next

End Sub



Private Sub CreateHPDiseaseArray()
Dim i, j As Integer
For i = 0 To 15
    For j = 0 To 4
        If j = 0 Then
            If i = 0 Then
                HPDiseaseArray(i, j) = 85 / 125
            ElseIf i = 1 Then
                HPDiseaseArray(i, j) = 85 / 130
            ElseIf i = 2 Then
                HPDiseaseArray(i, j) = 85 / 135
            ElseIf i = 3 Then
                HPDiseaseArray(i, j) = 85 / 140
            ElseIf i = 4 Then
                HPDiseaseArray(i, j) = 90 / 125
            ElseIf i = 5 Then
                HPDiseaseArray(i, j) = 90 / 130
            ElseIf i = 6 Then
                HPDiseaseArray(i, j) = 90 / 135
            ElseIf i = 7 Then
                HPDiseaseArray(i, j) = 90 / 140
            ElseIf i = 8 Then
                HPDiseaseArray(i, j) = 95 / 125
            ElseIf i = 9 Then
                HPDiseaseArray(i, j) = 95 / 130
            ElseIf i = 10 Then
```

```
                HPDiseaseArray(i, j) = 95 / 135
            ElseIf i = 11 Then
                HPDiseaseArray(i, j) = 95 / 140
            ElseIf i = 12 Then
                HPDiseaseArray(i, j) = 100 / 125
            ElseIf i = 13 Then
                HPDiseaseArray(i, j) = 100 / 130
            ElseIf i = 14 Then
                HPDiseaseArray(i, j) = 100 / 135
            ElseIf i = 15 Then
                HPDiseaseArray(i, j) = 100 / 140
            End If
        ElseIf j = 1 Then
            HPDiseaseArray(i, j) = 0.5
        Else
            HPDiseaseArray(i, j) = 1.0
        End If
    Next
Next

End Sub

Private Sub CreateHPMinimizeArray()
    Dim i, j As Integer
    For i = 0 To 15
        For j = 0 To 4
            If j = 0 Then
                If i = 0 Then
                    HPMinimizeArray(i, j) = 85 / 125
                ElseIf i = 1 Then
                    HPMinimizeArray(i, j) = 85 / 130
                ElseIf i = 2 Then
                    HPMinimizeArray(i, j) = 85 / 135
                ElseIf i = 3 Then
                    HPMinimizeArray(i, j) = 85 / 140
                ElseIf i = 4 Then
                    HPMinimizeArray(i, j) = 90 / 125
                ElseIf i = 5 Then
                    HPMinimizeArray(i, j) = 90 / 130
                ElseIf i = 6 Then
                    HPMinimizeArray(i, j) = 90 / 135
                ElseIf i = 7 Then
                    HPMinimizeArray(i, j) = 90 / 140
                ElseIf i = 8 Then
                    HPMinimizeArray(i, j) = 95 / 125
                ElseIf i = 9 Then
                    HPMinimizeArray(i, j) = 95 / 130
                ElseIf i = 10 Then
                    HPMinimizeArray(i, j) = 95 / 135
                ElseIf i = 11 Then
                    HPMinimizeArray(i, j) = 95 / 140
                ElseIf i = 12 Then
```

```
                        HPMinimizeArray(i, j) = 100 / 125
                    ElseIf i = 13 Then
                        HPMinimizeArray(i, j) = 100 / 130
                    ElseIf i = 14 Then
                        HPMinimizeArray(i, j) = 100 / 135
                    ElseIf i = 15 Then
                        HPMinimizeArray(i, j) = 100 / 140
                    End If
                Else
                    If (HPSystolicArray(i, j) <= HPDiastolicArray(i, j)) And HPSystolicArray(i, j) <=
HPDiseaseArray(i, j) Then
                        HPMinimizeArray(i, j) = HPSystolicArray(i, j)
                        'MsgBox("systolic")
                    ElseIf (HPDiastolicArray(i, j) <= HPSystolicArray(i, j)) And HPDiastolicArray(i, j)
<= HPDiseaseArray(i, j) Then
                        HPMinimizeArray(i, j) = HPDiastolicArray(i, j)
                        'MsgBox("Dystolic")
                    ElseIf (HPDiseaseArray(i, j) <= HPDiastolicArray(i, j)) And HPDiseaseArray(i, j)
<= HPSystolicArray(i, j) Then
                        HPMinimizeArray(i, j) = HPDiseaseArray(i, j)
                        'MsgBox("Disease")
                    End If
                End If
            Next
        Next
    End Sub

    Private Sub CreateMPDiastolicArray()
        Dim i, j As Integer
        For i = 0 To 15
            For j = 0 To 4
                If j = 0 Then
                    If i = 0 Then
                        MPDiastolicArray(i, j) = 75 / 115
                    ElseIf i = 1 Then
                        MPDiastolicArray(i, j) = 75 / 120
                    ElseIf i = 2 Then
                        MPDiastolicArray(i, j) = 75 / 125
                    ElseIf i = 3 Then
                        MPDiastolicArray(i, j) = 75 / 130
                    ElseIf i = 4 Then
                        MPDiastolicArray(i, j) = 80 / 115
                    ElseIf i = 5 Then
                        MPDiastolicArray(i, j) = 80 / 120
                    ElseIf i = 6 Then
                        MPDiastolicArray(i, j) = 80 / 125
                    ElseIf i = 7 Then
                        MPDiastolicArray(i, j) = 80 / 130
                    ElseIf i = 8 Then
                        MPDiastolicArray(i, j) = 85 / 115
                    ElseIf i = 9 Then
                        MPDiastolicArray(i, j) = 85 / 120
            ElseIf i = 10 Then
                MPDiastolicArray(i, j) = 85 / 125
            ElseIf i = 11 Then
                MPDiastolicArray(i, j) = 85 / 130
            ElseIf i = 12 Then
                MPDiastolicArray(i, j) = 90 / 115
            ElseIf i = 13 Then
                MPDiastolicArray(i, j) = 90 / 120
            ElseIf i = 14 Then
                MPDiastolicArray(i, j) = 90 / 125
            ElseIf i = 15 Then
                MPDiastolicArray(i, j) = 90 / 130
            End If
        Else
            If i <= 3 Then
                MPDiastolicArray(i, j) = 0.5
            ElseIf i >= 4 And i <= 7 Then
                MPDiastolicArray(i, j) = 1.0
            ElseIf i >= 8 And i <= 11 Then
                MPDiastolicArray(i, j) = 0.5
                'Else
                '    MPDiastolicArray(i, j) = 0.0
            ElseIf i = 12 Then
                MPDiastolicArray(i, j) = 0.1
            ElseIf i = 13 Then
                MPDiastolicArray(i, j) = 0.2
            ElseIf i = 14 Then
                MPDiastolicArray(i, j) = 0.3
            ElseIf i = 15 Then
                MPDiastolicArray(i, j) = 0.4
            End If
        End If
    Next
  Next
End Sub


Private Sub CreateMPSystolicArray()
  Dim i, j As Integer
  For i = 0 To 15
    For j = 0 To 4
        If j = 0 Then
            If i = 0 Then
                MPSystolicArray(i, j) = 75 / 115
            ElseIf i = 1 Then
                MPSystolicArray(i, j) = 75 / 120
            ElseIf i = 2 Then
                MPSystolicArray(i, j) = 75 / 125
            ElseIf i = 3 Then
                MPSystolicArray(i, j) = 75 / 130
            ElseIf i = 4 Then
                MPSystolicArray(i, j) = 80 / 115
            ElseIf i = 5 Then
                MPSystolicArray(i, j) = 80 / 120
            ElseIf i = 6 Then
                MPSystolicArray(i, j) = 80 / 125
            ElseIf i = 7 Then
                MPSystolicArray(i, j) = 80 / 130
            ElseIf i = 8 Then
                MPSystolicArray(i, j) = 85 / 115
            ElseIf i = 9 Then
                MPSystolicArray(i, j) = 85 / 120
            ElseIf i = 10 Then
                MPSystolicArray(i, j) = 85 / 125
            ElseIf i = 11 Then
                MPSystolicArray(i, j) = 85 / 130
            ElseIf i = 12 Then
                MPSystolicArray(i, j) = 90 / 115
            ElseIf i = 13 Then
                MPSystolicArray(i, j) = 90 / 120
            ElseIf i = 14 Then
                MPSystolicArray(i, j) = 90 / 125
            ElseIf i = 15 Then
                MPSystolicArray(i, j) = 90 / 130
            End If
        Else
            If i = 0 Or i = 4 Or i = 8 Or i = 12 Then
                MPSystolicArray(i, j) = 0.5
            ElseIf i = 1 Or i = 5 Or i = 9 Or i = 13 Then
                MPSystolicArray(i, j) = 1.0
            ElseIf i = 2 Or i = 6 Or i = 10 Or i = 14 Then
                MPSystolicArray(i, j) = 0.5
            ElseIf i = 3 Or i = 7 Or i = 11 Or i = 15 Then
                MPSystolicArray(i, j) = 0.1
            End If
        End If
    Next
Next
End Sub

Private Sub CreateMPDiseaseArray()
    Dim i, j As Integer
    For i = 0 To 15
        For j = 0 To 4
            If j = 0 Then
                If i = 0 Then
                    MPDiseaseArray(i, j) = 75 / 115
                ElseIf i = 1 Then
                    MPDiseaseArray(i, j) = 75 / 120
                ElseIf i = 2 Then
                    MPDiseaseArray(i, j) = 75 / 125
                ElseIf i = 3 Then
                    MPDiseaseArray(i, j) = 75 / 130
                ElseIf i = 4 Then
                MPDiseaseArray(i, j) = 80 / 115
              ElseIf i = 5 Then
                MPDiseaseArray(i, j) = 80 / 120
              ElseIf i = 6 Then
                MPDiseaseArray(i, j) = 80 / 125
              ElseIf i = 7 Then
                MPDiseaseArray(i, j) = 80 / 130
              ElseIf i = 8 Then
                MPDiseaseArray(i, j) = 85 / 115
              ElseIf i = 9 Then
                MPDiseaseArray(i, j) = 85 / 120
              ElseIf i = 10 Then
                MPDiseaseArray(i, j) = 85 / 125
              ElseIf i = 11 Then
                MPDiseaseArray(i, j) = 85 / 130
              ElseIf i = 12 Then
                MPDiseaseArray(i, j) = 90 / 115
              ElseIf i = 13 Then
                MPDiseaseArray(i, j) = 90 / 120
              ElseIf i = 14 Then
                MPDiseaseArray(i, j) = 90 / 125
              ElseIf i = 15 Then
                MPDiseaseArray(i, j) = 90 / 130
              End If
            Else
              If j = 1 Then
                MPDiseaseArray(i, j) = 0.5
              ElseIf j = 2 Then
                MPDiseaseArray(i, j) = 1.0
              ElseIf j = 3 Then
                MPDiseaseArray(i, j) = 0.5
              Else
                MPDiseaseArray(i, j) = 0.0
              End If
            End If
          Next
        Next
End Sub

Private Sub CreateMPMinimizeArray()
  Dim i, j As Integer
  For i = 0 To 15
    For j = 0 To 4
      If j = 0 Then
        If i = 0 Then
          MPMinimizeArray(i, j) = 75 / 115
        ElseIf i = 1 Then
          MPMinimizeArray(i, j) = 75 / 120
        ElseIf i = 2 Then
          MPMinimizeArray(i, j) = 75 / 125
        ElseIf i = 3 Then
          MPMinimizeArray(i, j) = 75 / 130
```

```
                ElseIf i = 4 Then
                    MPMinimizeArray(i, j) = 80 / 115
                ElseIf i = 5 Then
                    MPMinimizeArray(i, j) = 80 / 120
                ElseIf i = 6 Then
                    MPMinimizeArray(i, j) = 80 / 125
                ElseIf i = 7 Then
                    MPMinimizeArray(i, j) = 80 / 130
                ElseIf i = 8 Then
                    MPMinimizeArray(i, j) = 85 / 115
                ElseIf i = 9 Then
                    MPMinimizeArray(i, j) = 85 / 120
                ElseIf i = 10 Then
                    MPMinimizeArray(i, j) = 85 / 125
                ElseIf i = 11 Then
                    MPMinimizeArray(i, j) = 85 / 130
                ElseIf i = 12 Then
                    MPMinimizeArray(i, j) = 90 / 115
                ElseIf i = 13 Then
                    MPMinimizeArray(i, j) = 90 / 120
                ElseIf i = 14 Then
                    MPMinimizeArray(i, j) = 90 / 125
                ElseIf i = 15 Then
                    MPMinimizeArray(i, j) = 90 / 130
                End If
            Else
                If (MPSystolicArray(i, j) <= MPDiastolicArray(i, j)) And MPSystolicArray(i, j) <=
MPDiseaseArray(i, j) Then
                    MPMinimizeArray(i, j) = MPSystolicArray(i, j)
                ElseIf (MPDiastolicArray(i, j) <= MPSystolicArray(i, j)) And MPDiastolicArray(i,
j) <= MPDiseaseArray(i, j) Then
                    MPMinimizeArray(i, j) = MPDiastolicArray(i, j)
                ElseIf (MPDiseaseArray(i, j) <= MPDiastolicArray(i, j)) And MPDiseaseArray(i, j)
<= MPSystolicArray(i, j) Then
                    MPMinimizeArray(i, j) = MPDiseaseArray(i, j)
                End If
            End If
        Next
    Next
End Sub

Private Sub btnBP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnBP.Click
    Dim i, j As Integer
    Dim newx1, newx2, newx3, newx4 As Double
    Dim pState As String
    Dim mini As Double
    Dim maxi As Double
    Dim medi As Double
    Dim wrongRange As Boolean
    Dim bpvalue As String
    If Generated = False Then
    MsgBox("Please Generate Pressure Tables")
    Button1.Focus()
    Exit Sub
ElseIf Trim(txtLower.Text) = "" Then
    MsgBox("Please enter Diastolic Pressure")
    txtLower.Focus()
    Exit Sub
ElseIf Trim(txtUpper.Text) = "" Then
    MsgBox("Please enter Systolic Pressure")
    txtUpper.Focus()
    Exit Sub
ElseIf Val(txtLower.Text) < 40 Or Val(txtLower.Text) > 120 Then
    MsgBox("Diastolic Pressure value below 40 AND above 120 not acceptable")
    txtLower.Focus()
    Exit Sub
ElseIf Val(txtUpper.Text) < 90 Or Val(txtUpper.Text) > 160 Then
    MsgBox("Systolic Pressure value below 90 AND Above 160 not acceptable")
    txtUpper.Focus()
    Exit Sub
End If

wrongRange = False
newx1 = newx2 = newx3 = newx4 = 0.0
If (Val(txtLower.Text) Mod 10) >= 3 Then
    txtLower.Text = ((Val(txtLower.Text) \ 10) * 10) + 5
Else
    txtLower.Text = ((Val(txtLower.Text) \ 10) * 10)
End If

If (Val(txtUpper.Text) Mod 10) >= 3 Then
    txtUpper.Text = ((Val(txtUpper.Text) \ 10) * 10) + 5
Else
    txtUpper.Text = ((Val(txtUpper.Text) \ 10) * 10)
End If
'MsgBox(txtLower.Text)
'MsgBox(txtUpper.Text)
'Exit Sub

txtBP.Text = Val(txtLower.Text) / Val(txtUpper.Text)
For i = 0 To 15
    mini = MinimizeArray(i, 0)
    maxi = HPMinimizeArray(i, 0)
    medi = MPMinimizeArray(i, 0)

    'If i = 8 Then
    'MsgBox("Low" & MinimizeArray(i, 0))
    'MsgBox("Medium" & MPMinimizeArray(i, 0))
    'MsgBox("High" & HPMinimizeArray(i, 0))
    'MsgBox(Val(txtBP.Text))
    'End If
```

```
      If (Math.Round(mini, 15) = Math.Round(Val(txtBP.Text), 15) Or (txtLower.Text = 75
And txtUpper.Text = 115)) Then
        If (txtLower.Text <= 75 And txtUpper.Text <= 115) Or txtLower.Text <= 70 Then
          If txtLower.Text >= 40 And txtUpper.Text <= 115 Then
            newx1 = MinimizeArray(i, 1)
            newx2 = MinimizeArray(i, 2)
            newx3 = MinimizeArray(i, 3)
            newx4 = MinimizeArray(i, 4)
            pState = "Low"
            Exit For
          End If
        End If
      ElseIf (Math.Round(medi, 15) = Math.Round(Val(txtBP.Text), 15)) Or (txtLower.Text =
80 And txtUpper.Text = 115) Or (txtLower.Text = 85 And txtUpper.Text = 130) Or
(txtLower.Text = 85 And txtUpper.Text = 125) Then
        If (txtLower.Text >= 75 And (txtLower.Text <= 90 And txtUpper.Text <= 120)) Or
(txtLower.Text <= 85 And txtLower.Text >= 75) Then
          newx1 = MPMinimizeArray(i, 1)
          newx2 = MPMinimizeArray(i, 2)
          newx3 = MPMinimizeArray(i, 3)
          newx4 = MPMinimizeArray(i, 4)
          pState = "Medium"
          Exit For
        End If
      ElseIf Math.Round(maxi, 15) = Math.Round(Val(txtBP.Text), 15) Then
        If (txtLower.Text >= 85 And txtLower.Text <= 120 And txtUpper.Text <= 160) Then
          newx1 = HPMinimizeArray(i, 1)
          newx2 = HPMinimizeArray(i, 2)
          newx3 = HPMinimizeArray(i, 3)
          newx4 = HPMinimizeArray(i, 4)
          pState = "High"
          Exit For
        End If
      End If
    Next
    If txtLower.Text >= 40 And txtLower.Text <= 55 And txtUpper.Text >= 90 And
txtUpper.Text <= 120 Then
      newx1 = MinimizeArray(0, 1)
      newx2 = MinimizeArray(0, 2)
      newx3 = MinimizeArray(0, 3)
      newx4 = MinimizeArray(0, 4)
      pState = "Low"
    End If

    If txtLower.Text >= 60 And txtLower.Text <= 70 And txtUpper.Text >= 120 And
txtUpper.Text <= 160 Then
      newx1 = MinimizeArray(0, 1)
      newx2 = MinimizeArray(0, 2)
      newx3 = MinimizeArray(0, 3)
      newx4 = MinimizeArray(0, 4)
      pState = "Low"
    End If
    If txtLower.Text >= 90 And txtLower.Text <= 120 And txtUpper.Text > 140 And txtUpper.Text <= 160 Then
      newx1 = HPMinimizeArray(15, 1)
      newx2 = HPMinimizeArray(15, 2)
      newx3 = HPMinimizeArray(15, 3)
      newx4 = HPMinimizeArray(15, 4)
      pState = "High"
    End If
    MsgBox(newx1 & "-" & newx2 & "-" & newx3 & "-" & newx4)
    If pState = "Low" Then
      theValue = ((newx1 * 1) + (newx2 * 1.5) + (newx3 * 2) + (newx4 * 2.5)) / (newx1 + newx2 + newx3 + newx4)
    ElseIf pState = "Medium" Then
      theValue = ((newx1 * 2.5) + (newx2 * 3.0) + (newx3 * 3.5) + (newx4 * 4.0)) / (newx1 + newx2 + newx3 + newx4)
      'MsgBox(((2.5 * 0.1) + (0.1 * 3) + (0.1 * 3.5)) / (0.1 + 0.1 + 0.1))
    ElseIf pState = "High" Then
      theValue = ((newx1 * 3.5) + (newx2 * 4) + (newx3 * 4.5) + (newx4 * 5.0)) / (newx1 + newx2 + newx3 + newx4)
    End If
    txtBP.Text = theValue
    MsgBox("Pressure is" & theValue & "and that is " & pState)
  End Sub

  'For Low Blood Pressure
  Private Sub Mutate()
    Dim str1 As String
    Dim str2 As String
    Dim newstr1 As String
    Dim newstr2 As String
    str1 = TextBox3.Text
    str2 = TextBox4.Text
    newstr1 = str1.Remove(5, 4)
    newstr2 = str2.Remove(5, 4)
    TextBox6.Text = newstr1.Insert(4, "1010")
    TextBox5.Text = newstr2.Insert(4, "1010")
  End Sub

  'For High Blood Pressure
  Private Sub MutateHP()
    Dim str1 As String
    Dim str2 As String
    Dim newstr1 As String
    Dim newstr2 As String
    str1 = TextBox3.Text
    str2 = TextBox4.Text
    newstr1 = str1.Remove(9, 4)
    newstr2 = str2.Remove(9, 4)
    TextBox6.Text = newstr1.Insert(8, "1010")
    TextBox5.Text = newstr2.Insert(8, "1010")
  End Sub
```

```
'Common sub for all
Private Sub Randomclick()
   Dim intResult As Long
   Randomize()
   intResult = Int((65535 * Rnd()) + 0)
   TextBox2.Text = intResult
End Sub

'Common sub for all
Private Sub BinaryChange()
   TextBox2.Text = LongToBinary(TextBox2.Text)
   TextBox2.Text = Mid(TextBox2.Text, 1, 16)
End Sub

'Common sub for all
Function Bin2Dec(ByVal Num As String) As Long
   Dim n, a As Integer
   Dim x As String
   n = Len(Num) - 1
   a = n
   Do While n > -1
      x = Mid(Num, ((a + 1) - n), 1)
      Bin2Dec = IIf((x = "1"), Bin2Dec + (2 ^ (n)), Bin2Dec)
      n = n - 1
   Loop
End Function

'Common sub for all
Private Function LongToBinary(ByVal long_value As Long, Optional ByVal separate_bytes
As Boolean = True) As String
   Dim hex_string As String
   Dim digit_num As Integer
   Dim digit_value As Integer
   Dim nibble_string As String
   Dim result_string As String
   Dim factor As Integer
   Dim bit As Integer
   ' Convert into hex.
   hex_string = Hex$(long_value)
   ' Zero-pad to a full 8 characters.
   'hex_string = Right$(String$(8, "0") & hex_string, 8)
   hex_string = hex_string.PadRight(8, "0")
   ' Read the hexadecimal digits
   ' one at a time from right to left.
   For digit_num = 8 To 1 Step -1
      ' Convert this hexadecimal digit into a
      ' binary nibble.
      digit_value = CLng("&H" & Mid$(hex_string, digit_num, 1))
      ' Convert the value into bits.
      factor = 1
      nibble_string = ""
      For bit = 3 To 0 Step -1
```

```
            If digit_value And factor Then
               nibble_string = "1" & nibble_string
            Else
               nibble_string = "0" & nibble_string
            End If
            factor = factor * 2
         Next bit
         ' Add the nibble's string to the left of the
         ' result string.
         result_string = nibble_string & result_string
      Next digit_num

      ' Add spaces between bytes if desired.
      'If separate_bytes Then
      '    result_string = _
      '        Mid$(result_string, 1, 8) & " " & _
      '        Mid$(result_string, 9, 8) & " " & _
      '        Mid$(result_string, 17, 8) & " " & _
      '        Mid$(result_string, 25, 8)
      'End If
      ' Return the result.
      LongToBinary = result_string
   End Function

   'Common sub for all
   Private Sub crossover()
      Dim str1 As String
      Dim str2 As String
      Dim crsValuStr1 As String
      Dim crsValuStr2 As String
      Dim newstr1 As String
      Dim newstr2 As String
      str1 = TextBox1.Text
      str2 = TextBox2.Text
      crsValuStr1 = Mid(str1, 5, 8)
      crsValuStr2 = Mid(str2, 5, 8)

      newstr1 = str1.Remove(5, 8)
      newstr2 = str2.Remove(5, 8)

      TextBox3.Text = newstr1.Insert(4, crsValuStr2)
      TextBox4.Text = newstr2.Insert(4, crsValuStr1)
   End Sub

   Private Sub btnOutput_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnOutput.Click
      CreateExcelForm()
   End Sub

   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
      StartHPProcess()
```

```
    CreateMPDiastolicArray()
    CreateMPSystolicArray()
    CreateMPDiseaseArray()
    CreateMPMinimizeArray()
    Generated = True
  End Sub

  'Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
  '  Dim i, j As Integer
  '    For i = 0 To 15
  '        MsgBox("1- " & HPMinimizeArray(i, 1) & "2- " & HPMinimizeArray(i, 2) & "3- " &
HPMinimizeArray(i, 3) & "4- " & HPMinimizeArray(i, 4))
  '    Next
  'End Sub
End Class
```

REFERENCES

1. S.Rajasekaran, G.A. Vijayalakshmi Pai: Neural Networks, Fuzzy Logic,and Genetic Algorithms.
2. Soh, c.k and J.yang: Fuzzy controlled Genetic Algorithm Search for shape optimization.
3. Witold Pedrycz and Fernando Gomide: An Introduction to Fuzzy Sets, Analysis and Design.
4. Internet search www/Google,com website
5. IEEE journals.