# Encryption algorithm

Genetic algorithm is an adaptive heuristic searching algorithm that based its ideas from the evolutionary ideas of natural selection of Darwinism. (Doc.ic.ac.uk, n.d.) It comprises of 3 main functionality, namely, reproduction, crossover, and mutation. In my implementation of the encryption with genetic algorithm, I took partial inspiration from the pseudocode provided by the article “Secret Key Encryption Algorithm Using Genetic Algorithm” by Ankita Agarwal. I also included my own modifications to make this a better encryption rather than just to encrypt marks, it is now able to encrypt any texts to multi-paragraph levels. In nature, our modifications allow each same sequence of input text to be converted to different ciphertext and keys for each encryption process.

The input is first specified by the user, and then it is converted into ASCII decimal representation, then it is converted to a 8-bit binary representation for each and every single character. For example, the user entered “A1”, the ASCII code decimal representation of the character “A” will be 65, while the representation for integer “1” would be 49. Then the decimal is converted into a 8-bit representation, the 8-bit of 65 will be [0,1,0,0,0,0,0,1], and [0,0,1,1,0,0,0,1] for 49. Each character is represented separately and encrypted separately.

The pseudocode in reference did not state about the selection process for the crossover, therefore in this section I’ve choose to randomize the index of bit selection for every single encryption process in crossover and mutation. This ensures maximum anonymity as nothing is certain, the secret keys generated will be the selected bits to be crossed and mutated. Following the pseudocode, the crossover function is done swapping the 2 selected key’s position. Although it is a simple process, but swapping bits around makes a big differences in the representation of the binary. For example, the keys for the first encryption for character “A” is 4 and 8, then the result 8-bit will be [1,1,0,0,0,0,0,0], if we convert back to decimal representation this result would be 192, a big differences from 65.

After crossing, the mutation of the value will take place, similarly, the author did not mention about the possibility of mutation to occur. I choose to allow a 100% occurrence of the mutation because the generation of encryption is very small, therefore allowing 100% mutation rate allows us to achieve the optimum resolution faster. Following the pseudocode, the bit inversion operation is implemented, which changes the state of the bit from 0 to 1 and vice versa. The secret key which is used to represent the chosen element to be inversed is also generated randomly to maximize anonymity. Continuing the previous example, if the secret key generated is 1, the first element in the value will be inverted, the result would be [0,1,0,0,0,0,0,0], which is 64 in decimal.

Generation of encryption is random to maximize anonymity and to provide better security. Once there is at least a 4 bit of difference between the masked value and the original value, it will exit the loop of the encryption and it is considered masked. For example: “A” original ASCII bit representation [0,0,1,1,0,0,0,1], the value [1,0,0,1,1,0,0,0] is achieved in the 10th generation of encryption, the encryption process is considered done. The differences here is the first, third, fifth, and the eighth bit, which sums up to total of 4 differences in bit. Once the last character has been fully encrypted, the 8-bit representations is converted back to ASCII decimal, and then back to the actual integer and characters.

The secret keys used during the crossover and mutation are all recorded under a variable using the stringBuilder class provided by java API, the keys are then pass to the decrypter to be used to do reverse engineering on the encryption.

During the development of the genetic algorithm, we faced the problem to choose the most suitable generation for the encryption process. Due to the nature of having a constant value, the hacker may be able to see through this weak point and obtain the exact value of the generations we are using the encrypt the information. One alternatives we thought up was to set the generation up until the bits are the direct opposite of the original value. However, this exposes another weak point where every word’s ASCII decimal representation is the exact opposite of itself. Thus, finally we’ve set up to 4 differences, when the 8-bit representation has 4 differences, we will end the encryption. This will ensure anonymity of the original value because the attacker will have a hard time to determine whether which 4 is the different even if they found out how the algorithm works.

# Decryption algorithm

In the context of devising an algorithm to decrypt encryption as proposed on the Question1B, we’re using a supervised machine learning algorithm for decryption, in particular with the use of multi-layer perceptron neural network as the concept. The reasoning of the choice of using supervised learning algorithm is due to the fact that the steps of encryption has been properly understood by decrypter, therefore it is possible to come up with a training set, which will yield a better accuracy than the use of unsupervised learning algorithm.   
Along with it is also implemented the backpropagation algorithm with sigmoid activation function.  
The general decryption steps of the implementation of our algorithm is that it decrypts any user-inputted sentences by breaking it down to a “letter” level and since the previous crossover points and mutation points are recorded as a secret key which is transferred along with the message, it is therefore possible to perform reverse crossover and reverse mutation process by using ANN.  
In addition, as the encryption consists of a ciphertext and keys that are highly-reactive and variable, it is therefore extremely important to do a certain amount of preprocessing before the data can be passed into a neural network in order to minimize the scope of considerations by the neural network. This approach ultimately will shorten the training time significantly for the neural network.   
To accomplish this task with efficiently, the network system of the ANN is composed of 2 sub-neural networks, each is responsible for their respective task.

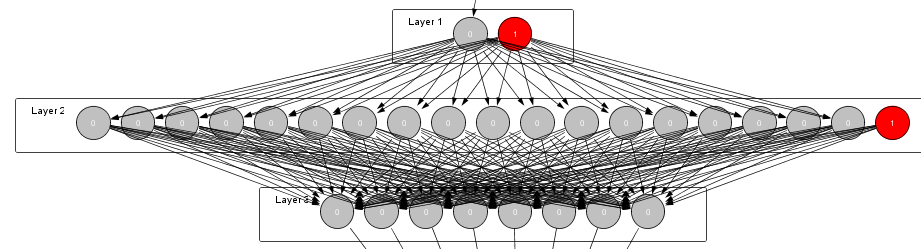
## 1st Neural Network(“Categorization” Neural Network)

The first neural network system serves the function to categorize the ciphertext and the secret keys generated from the encryption into a binary format consisting maximum amount of bits (8 bits) required to represent every single computer-supported character in ASCII format.

**Training set for the First Neural Network:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input | Output1 | Output2 | Output3 | Output4 | Output5 | Output6 | Output7 | Output8 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

To train a neural network with the training set above effectively, we have tested that by using a multi-layer perceptron consisting of 1 hidden layer is sufficient, with the amount of hidden layer containing 18 units, with bias each at every hidden units and output units.  
The objective of feeding the training set above to our system is to make the neural network understand the concept of positional index in order to represent a mutation point and crossover points in a binary format, and to pass all of the value to the next neural network. Essentially, this means the input neuron shall translate the current value it holds, for example “i”, to and activate a signal at output neuron “i+1”. If an output neuron (i+1) is activated, it is inferred that it is must either be a crossover point or a mutation point.  
In order for the system to recognize which one is a crossover point and a mutation point, we implemented a custom algorithm for the recognition.   
The connections of the network are such that each input unit is connected to each hidden unit, and each hidden unit is connected to each output unit. The reason of using significant number of weights are due to the fact that overfitting is not a possibility in this network as the training input has already covered every single possible value that is passable to the neural network.   
The following figure represents the visual topology of the first neural network :



## 2nd Neural Network / “SwapElement” Neural Network

In the 2nd neural network, one of the input received is from the signal received from previous neural network. Since in previous neural network, we implemented a custom algorithm to separate the mutation point, all of the layers in this particular neural network only aims to perform the reverse operations from mutation and crossover. However since the amount of generations are randomized for previous text encryption are randomized, we provide to use a iterative method to loop through every index of the ciphertext to fetch the value of bit at every index.  
Using only 2 of these training set for each reverse operations is enough to train all possible iterations for use of the network.

## Crossover Decryption Training Set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input1 | Input2 | Input3 | Output1 | Output2 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

The training set above is used to enable the system to simulate the process of decrypting a crossover.   
In a test/running scenario,both input1 and input3 denotes the value of the bit at both different locations and input2 denotes whether there is an activation value for crossover at the index. In this case, if the activation value is 1, we provide the value of input1 and input3 is switched, and the value from the switch will be represented as Output1 and Output3 respectively.  
The amount of hidden units in hidden layer implemented are also 18 units, with each input neuron connected to each hidden neuron, and each hidden neuron connected to each output neuron.

## Mutation Decryption Training Set

|  |  |  |
| --- | --- | --- |
| Input1 | Input2 | Output1 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

n order to simulate the reverse mutation operation, we will be using a training set similar to that of a XOR training set. Input1 functions as the iterated/looped output unit while Input2 serves as the activation value from the 1st neural network. Value at Output1 represents the switch.  
The steps for mutation decryption and crossover decryption is carried out repeatedly until every single generation has been traversed and every single character has been decrypted.

# Air-conditioned Fuzzy System

Air-conditioners can be set to have an absolute temperature, however this does not reflect the same level of coldness every day to every person despite maintain the exact same air-conditioner temperature. A survey was conducted to research whether if there are any other factors which caused this beside the person’s temperature preferences.

One of the key factors that we are able to obtain from the survey is the outside temperature. On rainy days, the temperature would be in the low range, while on a hot day, the outside temperature would be in the high range and anything in between would be in the cloudy range. The reason that we have chosen outside temperature as another factor that affects the coldness level is because the outside temperature can be measured, which allows us to design our fuzzy system and be able to output a crisp value being the coldness level.

The method that was used in our fuzzy system is Mamdani instead of Sugeno. This is because Mamdani allows the user to select an imprecise input instead of an absolute value (Alonso, 2016). In our program, a person may think that the air-conditioner temperature is quite cold, which may indicate that it is not absolutely cold but somewhere in between cold and normal, thus the person allows the user to select the option of temperature being cold and hot. Once selected, the program will generates a range within the linguistic variable provided and output a crisp value. Not only that, Mamdani method also allows us to determine the membership of our output, this can be done because the inference result before defuzzification is a fuzzy set (Wang and Chen, 2014).

In our program, there are 2 factors which produces 1 output, all of them has its own fuzzy set containing its own range and linguistic variables. The membership function types that we decided to use out of all the existing possible ones are trapeziums and trapezoids. The first factor is the air-conditioner temperature, the range of the air-conditioner temperature is 16 to 25 degrees and this is determined by the range of predictions that were given to us by our surveyors and the basic temperature range of air-conditioners. The range is then categorized using linguistics variables being cold, medium and hot. The fit vectors of the linguistic variable cold are {1/16, 1/17, 0/20}, for medium, the fit vectors are {0/17, 1/20, 1/21, 0/24} and lastly for hot, {0/21, 1/24, 1/25}. The visual representation of the fuzzy set of the air-conditioner temperature is as below:

The second factor is the outside weather temperature with a range from 24 to 33 degrees. The range is deduced from MOSTI, where they recorded the average minimum and maximum temperatures of multiple stations all over Malaysia (MOSTI, 2016). The linguistic variable that are used to describe the fuzzy set are raining, cloudy and sunny. Weather conditions are used as linguistic variables as according to the survey, it can be concluded that weather condition strongly reflects the outside temperature. For raining, the fit vectors are {1/24, 1/25, 0/28}, whereas for cloudy, the fit vectors are {0/25, 1/28, 1/29, 0/32} and for sunny, the fit vectors are {0/29, 1/30, 1/33}. The visual representation of the fuzzy set of the weather temperature is as below:

The last set is the output set which is the coldness level, it has a scale from 0 to 10. The linguistic variables that are used are low, medium, high. The fit vectors of cold are {1/0, 1/2, 0/4}, medium is {0/2, 1/4, 1/6, 0/8}, high is {0/6, 1/8, 1/10}.

The rules are deduced through our analysis on the survey that was conducted, it is found that the air-conditioner’s temperature plays a more effective role in determining the coldness of the library compared to the weather. We have deduced 9 rules which would enable us to provide an output for every possible case scenario. The rules are as below

Rule 1: If air-conditioner temperature is **cold** and the weather is **raining**, the coldness level is **high**

Rule 2: If air-conditioner temperature is **cold** and the weather is **cloudy**, the coldness level is **high**

Rule 3: If air-conditioner temperature is **cold** and the weather is **sunny**, the coldness level is **medium**

Rule 4: If air-conditioner temperature is **normal** and the weather is **raining**, the coldness level is **high**

Rule 5: If air-conditioner temperature is **normal** and the weather is **cloudy**, the coldness level is **medium**

Rule 6: If air-conditioner temperature is **normal** and the weather is **sunny**, the coldness level is **low**

Rule 7: If air-conditioner temperature is **hot** and the weather is **raining**, the coldness level is **medium**

Rule 8: If air-conditioner temperature is **hot** and the weather is **cloudy**, the coldness level is **low**

Rule 9: If air-conditioner temperature is **hot** and the weather is **sunny**, the coldness level is **low**

As seen from the above rules, weather temperatures do have an effect towards the coldness level but it still leans more towards the air-conditioner’s temperature. Thus as we can see in rule 1 and 2, although the weather temperature is changed, the coldness level still belongs to the high range. For rule 3 however, there has been some data which states that on a sunny weather where the temperature is near or at its peak, some would consider the coldness level to be normal and thus the rule is so. For rule 4, 5 and 6, the outside weather proves more significance when the air-conditioner temperature is in the normal range, thus making the coldness level to lean more towards the outside temperature. For rule 7, 8 and 9, the rules are similar to rule 1, 2 and 3.

A rule matrix is represented below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Weather | | | |
| Air-conditioner  Temperature |  | **Raining** | **Cloudy** | **Sunny** |
| **Cold** | High | High | Medium |
| **Normal** | High | Medium | Low |
| **Hot** | Medium | Low | Low |

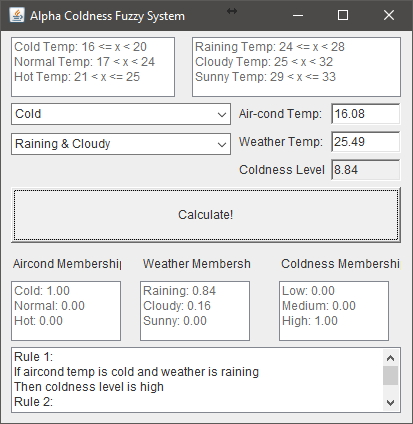
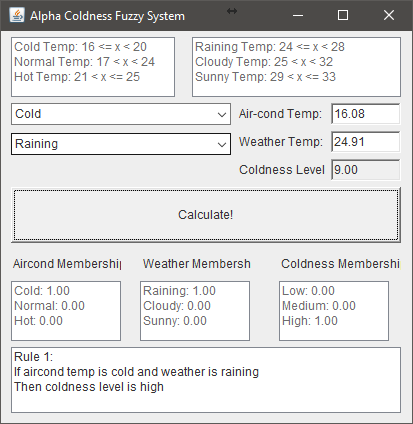
Our program uses the Mamdani method, which has four major steps, being the fuzzification, rules evaluation, aggregation and lastly defuzzification. In the beginning, the system goes through the fuzzification step where it computes the membership value of the given data, which in this case would be the air-conditioner temperature and the weather temperature. Once the membership values are obtained, these values are used in rule evaluation whereby each rule takes in the corresponding membership value and decide which rule is suitable to be fired. After that, the next step is aggregation where we obtain the three highest coldness membership value that was fired by the rules and use this to create a single polygon fuzzy set. Once the polygon is formed, the final step is defuzzification where we obtain a crisp value from the polygon.

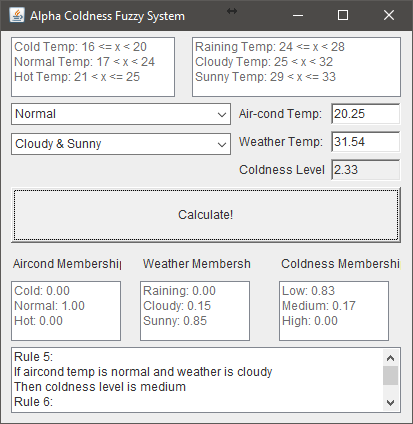
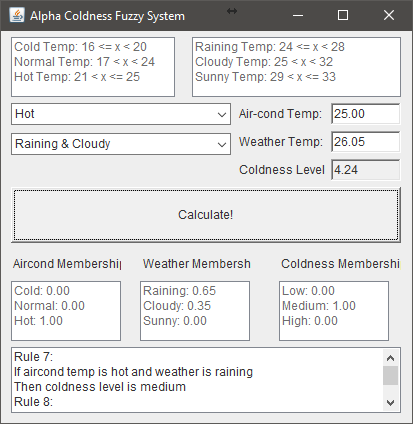
Defuzzification can be done in many ways with the most famous centroid technique which is known for its accuracy to find the crisp value. However, the problem with the technique is that, it is very difficult to implement as finding the intersection between two fuzzy sets is an extremely difficult task. Thus we moved on to an alternative technique being the center of sum technique, which goes by the formula:

https://zone.ni.com/images/reference/en-XX/help/370401G-01/loc_eq_centerofsums.gif

This formula work by firstly calculating the centroid of a single fuzzy set, and then multiplying this with the area of that same fuzzy set. This is done for every set that is available and the values obtained from every set is summed up together and the total is divided with the sum of all the areas of the sets available. Although the center of sum technique may not be as accurate as the centroid technique however it is feasible to be used to find the coldness of the library as there are many more factors which have yet to be considered, thus the crisp value obtained for the program is not absolutely accurate regardless of what technique is used.

Below are some sample screenshots of the implementation of our fuzzy system:





References

Agarwal, A. (2012). Secret Key Encryption Algorithm Using Genetic Algorithm. International Journal of Advanced Research in Computer Science and Software Engineering, 2(4), pp.216 - 218.

Alonso, S. (2016). *eMathTeacher: Mamdani's fuzzy inference method - FIS Introduction*. [online] Dma.fi.upm.es. Available at: http://www.dma.fi.upm.es/recursos/aplicaciones/logica\_borrosa/web/fuzzy\_inferencia/introfis\_en.htm [Accessed 10 Oct. 2016].

Doc.ic.ac.uk. (n.d.). Introduction to Genetic Algorithms. [online] Available at: https://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol1/hmw/article1.html [Accessed 13 Oct. 2016].

MOSTI, (2016). *Min / Max Temperature - Malaysian Meteorological Department*. [online] Met.gov.my. Available at: http://www.met.gov.my/en/web/metmalaysia/observations/surface/minmaxtemperature [Accessed 12 Oct. 2016].

Wang, Y. and Chen, Y. (2014). A Comparison of Mamdani and Sugeno Fuzzy Inference Systems for Traffic Flow Prediction. *JCP*, 9(1).