CHAPTER 1
INTRODUCTION

1.1. Background

Heat convection involves energy transfer between a surface and a flowing fluid due to the temperature difference between the surface and fluid. To solve analytically, one would require the knowledge of flow field and temperature distribution. This can be obtained by solving the mass, momentum and energy equations (namely the Navier-Stokes equation) considering the geometry and fluid properties. However the solution is only available for simple geometries whereas practical heat transfer problems involve complex geometries. The approach via Newton's law of cooling is more practical as it reduces convection problem to obtaining convective heat transfer coefficients.

Therefore, this approach can be used to solve heat exchanger problems, especially if the aim is to determine the Nusselt number correlation. Then the related heat transfer coefficient values can be found. Ultimately, the heat transfer coefficients are used for the purpose of sizing the heat exchanger, for instance the diameter and length of tube. The determination of the heat transfer coefficients is best described using a tube in tube heat exchanger, which is the focus of this project.

Take the example of a fluid (fluid A) flowing in a tube which is contained in another tube (both forms the tube in tube heat exchanger).

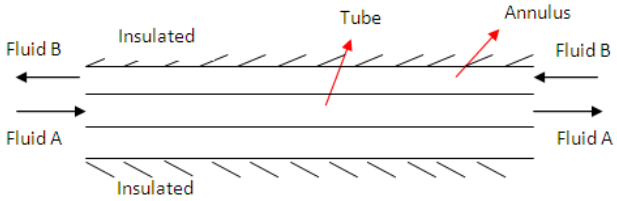


Figure 1.1 Typical Configuration of Concentric Heat Exchanger

The fluid flowing in the annulus (fluid B) is in the opposite the fluid direction of fluid A, thus forming a counter-flow arrangement. Say that fluid A is of a higher temperature; therefore it loses heat to fluid B. Another assumption is that the surface outside the annulus is insulated, thus no heat transfer occurs from annulus to the surroundings.

The heat transfer in the tube can be calculated as follows,

$$Q = \dot{m} * C_p * \Delta T \quad (1)$$

The heat transfer coefficients of the system are of interests, therefore

$$Q = U * A * \Delta T_{LMTD} \quad (2)$$

With the background established, it is easier for the reader to grasp all the information presented in the proceeding sections.

1.2. Problem Statement

From equation 1, different values of Q can be obtained by varying the mass flow rates of Fluid A. Overall, there are three main thermal resistances participating in the heat transfer process, and they can be represented by an overall total thermal resistance, R_{ov}.

In the problem, R_{ov} consists of three terms, one being the thermal resistance caused by convection inside the tube, second being the thermal resistance caused by the tube wall and the third being the thermal resistance caused by convection outside the tube.

$$R_{ov} = R_i + R_w + R_o$$

Expanding the individual terms,

$$R_{ov} = \frac{1}{h_i A_i} + \frac{\ln(d_o/d_i)}{2*\pi*k_w*L_w} + \frac{1}{h_o A_o} \quad (3)$$

Where A_i and A_o are the surface areas of the inner and outer tube diameters, respectively, k_w is the thermal conductivity of the tube wall material.

From equations 1 and 2,

$$\dot{m}_{fluid\ A} * C_p * \Delta T = U * A_i * \Delta T_{LMTD}$$

With,

$$\frac{1}{U * A} = R_{ov}$$

Finally, a term for R_{ov} can be obtained:

$$R_{ov} = \frac{\Delta T_{LMTD}}{\dot{m}_{fluid\ A} * C_p * \Delta T}$$

Now that the value of R_{ov} has been obtained and R_w, being a constant, two unknowns h_i and h_o are yet to be determined and this leads to the mathematical conundrum of indeterminacy in the form of two unknowns and one equation.

One approach is to utilize the Wilson Plot method. In this method, R_{ov} is varied by having $\dot{m}_{fluid\ A}$ changed, since fluid velocity would influence the value of h_i. On the other hand, h_o is kept constant by holding the temperature difference constant, as well as fluid B's mass flow rate being constant.

Wilson deduces that, for fully developed liquid flow inside a tube,

$$h_i = C_2 * V_r^{\ n} \tag{4}$$

Since ho and the tube wall thermal resistance are constant, we group them as

$$C_1 = R_w + R_o$$

Rearranging equation 3,

$$R_{ov} = \frac{1}{C_2 * A_i} * \frac{1}{V_r^{\ n}} + C_1 \tag{5}$$

In which a straight graph (y= mx+c) can be observed from this equation. A straight graph then can be plot for different values of R_{ov} and its respective value of $\frac{1}{V_r^n}$.

The slope can be calculated as $\frac{1}{C_2 * A_i}$ to obtain C_2, whereas to C_1 is obtained via the intersection of the graph. Different values of h_i can be obtained with different flow rates. From the value of C_1, the value of h_o can be obtained too.

The form of the Nusselt Correlation as shown in equation 4 is of a simple one. In many applications, the form is much more complex, with the inclusion of Reynolds number, Prandtl number and other factors. With such inclusion, there are more exponents (in which there is only one in equation 4, and that is n) to be deduced and industrial practitioners such as heat exchanger manufacturers or heat exchanger researches rely upon the values of the exponents obtained via the literature. This presents a form of uncertainty, and the correlations are used to determine the dimensions of the heat exchanger, using equation 4, the size is not of an optimal one, resulting in the usage of more materials to construct the heat exchanger and the cost is increased as well.

Therefore, in Wilson plot method a suitable Nusselt Number correlation form and its related exponents are assumed, as evident in the above discussion. However, there should be a method in which both inside and outside heat transfer coefficients can be directly obtained without using any correlations a priori. That is why it is hoped that through this project, GA can be used, coupled with the processing power of modern computers, to avoid the cumbersomeness of the Wilson plot method, as well as to take the advantage of GA's global search properties.

1.3. Objectives

The main objective of this project is to determine the various exponents and coefficients of the Nusselt Number correlations in the plain tube and brazed plate heat exchangers using Genetic Algorithm. Having determined these exponents and coefficients, the heat transfer rate can be predicted and this will be known as the Genetic Algorithm simulated heat transfer rate, Q_{GA}.

Q_{GA} will be compared with the heat transfer rate predicted as a result of using the industrial given Nusselt correlations, $Q_{industry}$. Based on the concept that GA are efficient search methods that can usually result in optimum solution and that they are also robust searching tools, the objective is to determine the hi and ho pair that can minimize equation 6 as follows:

$$S = \frac{1}{N}\sum_{i=1}^{N}(Q_i{}^{E} - Q_i{}^{S}) \tag{6}$$

where the superscripts E and S represent experimental (actual) and simulated respectively, and N is the number of experimental data. Therefore, the actual heat transfer rate obtained from the industry will serve as a basis in this project, as it becomes the measure of the degree of fit between the actual heat transfer rate and the Genetic Algorithm simulated heat transfer rate. As mentioned earlier, greater accuracy in correlation would ensure proper sizing of heat exchangers, reducing cost ultimately. With proper modeling and coding, this project is definitely feasible and is expected to be completed within the stipulated time frame.

CHAPTER 2
LITERATURE REVIEW

2.1. What is Genetic Algorithm?

Genetic algorithm is a searching method inspired by natural evolution. Darwinian's theory of evolution states that only the best will survive. Therefore, GA incorporates the principles of evolution during the searching process.

Ultimately, the searching process in GA is used to determine a single or a set of solutions that are optimal. Searching can be either exhaustive or intelligent. In the former, there are no guidelines in searching – searching is performed by merely experimenting all the possible solutions, hence the idea of brute force is noted here. When the number of solutions is confined to hundreds, perhaps it is possible. However, what if millions of solutions are available? Hence, certain search mechanisms that quickly ignore inferior solutions and converge to the optimal ones are required. GA is fit to perform such tasks.

In essence, GA is utilized in the process of searching for a set of parameters that can optimize a problem or function. A good example would the design of a heat exchanger. Say that the focus of design is in the diameter, and it can be correlated with temperature, pressure drop, and fluid velocity. Under certain restrictive criteria for temperature, pressure drop, and fluid velocity that can provide the optimal diameter, one can employ GA to search for these three parameters that offer the best diameter for the heat exchanger. In all optimization problems, constraints form the backbone in the searching process. It could be said that constraints guide the search

towards more feasible regions, otherwise a blind search would not produce any good results. Just as in real life, one has limited operating resources, optimization are to be limited by constraints.

2.2. Uses of GA

2.2.1. Optimization design of shell-and-tube heat exchanger by entropy generation minimization and genetic algorithm [8]

In this paper, a new shell and tube heat exchanger optimization method is developed in which the dimensionless entropy generation rate obtained by scaling the entropy generation on the ratio of the heat transfer rate to the inlet temperature of cold fluid is employed as the objective function, some geometrical parameters of the shell-and-tube heat exchanger are taken as the design variables and the genetic algorithm is applied to solve the optimization problem. The fitness function is maximizing the negative of the entropy generation function suggested by Hesselgreaves. GA is applied to solve the multi-variable problems which not only yield global optimum, but also demonstrates the flexibility to select the design variables and constraint conditions.

2.2.2. Robot Trajectory Planning [2]

A robot trajectory describes the position, orientation, velocity and acceleration of each robot component as a function of tie. In this example, the robot is a two-link arm having two degrees of freedom in a plane called the robot workspace. An obstacle has to be taken into account when designing the cost function, since the robot is required to move without colliding with the obstacle. Thus, the cost or fitness function represents the length of the line required to get from the starting point to the ending point as prescribed by the user, with the inclusion of the obstacle as mentioned. The genetic algorithm is then utilized to determine the shortest path from start to finish. The results show that the path becomes shorter as the GA progresses.

2.2.3. Solving Job-Shops Scheduling Problems [2]

In this paper, GA is applied to determine the sequence of task that can satisfy the constraint of the job-shop problem, namely the precedence constraint, which is defined by the sequential routing of tasks within a job since the inter-tasks sequence are predetermined and capacity constraints which restrict the use of each resource to only one task at a time, in a sense that two tasks that use the same resource cannot overlap or precede over one another. To come up with the fitness function, a strategy is defined to build a scheduling from the individual representation of the sequence, then the completion time of every job is calculated, and the maximum value of these times, that is the makespan, is the fitness value of the individual.

2.2.4. Optimal Design of Heat Exchangers: A Genetic Algorithm Framework [1]

In this paper, the authors apply GA to come up with the optimal design of heat exchanger. Particularly, they were attempting to reduce the heat transfer area for a given amount of heat duty. They encoded the solution to the optimization problem in the form of a particular heat exchanger configuration. This includes baffle choices, number of shells in series, number of tube passes, tube length and layout, etc, subjected to feasibility and vibration constraints. Feasibility constraint is to determine if the particular heat exchanger configuration generated is feasible or not, while vibration constraints takes into account that less baffles are used to reduce pressure drop due to large flow velocities and this constraint helps the GA to come up with practical HX configurations solutions. The authors compared the results supplied by a company that use base-case design to come up with the optimal heat exchanger. Results show that GA could improve the results by further reducing the area and cost for the given heat load, and present savings in computational time as well.

2.2.5. Determination of Thermal Compact Model via Evolutionary Genetic Optimization Method [5]

In this paper, GA is applied to determine a thermal compact model of a micro lead frame package, which is then used to compute the junction temperature for various boundary conditions. First a star shaped network is used as a basis network since it is basic and the simplest network.

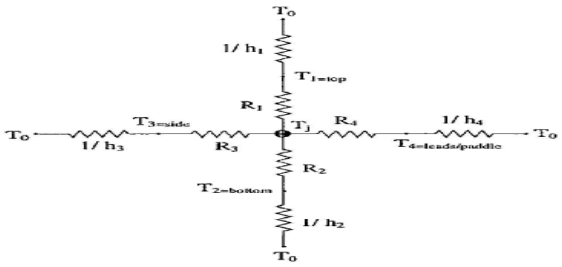


Figure 2.1 Star shaped resistance network

The R1, R2, R3 and R4 values are considered the solutions in GA. Together they are tested for 38 different boundary conditions that will give rise to the different values of h1, h2, h3 and h4. Then they are solved via the Gaussian elimination method to obtain the junction temperatures since the following matrix is needed:

$$\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4} & -\frac{1}{R_1} & -\frac{1}{R_2} & -\frac{1}{R_3} & -\frac{1}{R_4} \\ -\frac{1}{R_1} & \frac{1}{R_1}+h_1A_1 & 0 & 0 & 0 \\ -\frac{1}{R_2} & 0 & \frac{1}{R_2}+h_2A_2 & 0 & 0 \\ -\frac{1}{R_3} & 0 & 0 & \frac{1}{R_3}+h_3A_3 & 0 \\ -\frac{1}{R_4} & 0 & 0 & 0 & \frac{1}{R_4}+h_4A_4 \end{bmatrix} \begin{bmatrix} Tj^{CM} \\ T_1^{CM} \\ T_2^{CM} \\ T_3^{CM} \\ T_4^{CM} \end{bmatrix} = \begin{bmatrix} P \\ T_oh_1A_1 \\ T_oh_2A_2 \\ T_oh_3A_3 \\ T_oh_4A_4 \end{bmatrix}$$

Figure 2.2 Simultaneous Equations to Solve for the Desired Temperatures

The obtained temperatures are compared with the Finite Element method generated junction temperature and the difference becomes the objective or cost function. The first cost function value obtained is used as a basis of comparison for other networks.

2.2.6 Hybrid Genetic Algorithms for Structural Reliability Analysis [9]

In this paper, GA is used to search for the minimum reliability index of a structural system with ANN (Artificial Neural Network) is utilized as a metamodel. Metamodel resembles that of a model without any mathematical derivations and equations. The reliability index is subjected to the minimization of the limit state

function. In real life, the design problem can be large, and the limit state functions which are generated from finite element codes, in which they are implicit in terms of random variables. In other words, if GA alone is used, then each and every time when the limit state function is required, the finite element codes or program have to run and this will result in severe down-time, as finite element program usually takes a long time to reach a solution. However, this problem can be avoided by having an approximate limit state function by first employing the inputs and outputs of the finite element software into ANN. Once ANN recognizes the behavior that determines the limit state function, future predictions of the limit state function can be made, though they are more of an approximation. In spite of this, a lot of time can be saved. Again, GA relies on the parameters generated by the ANN metamodel.

2.3 Summary

In essence, the papers above discuss the capabilities of GA. The paper that bears greatest resemblance and importance to this project is the Determination of Thermal Compact Model via Evolutionary Genetic Optimization Method paper. In this paper, the temperatures at each junction are to be optimized by considering the boundary conditions and resistances of each junction. The fitness function is the percentage difference incurred between the GA derived temperatures and the Finite Element model generated temperatures. Therefore, the authors were interested in obtaining the values of the resistances, which are later utilized to introduce a better thermal compact model of the circuit that can be used to predict a more accurate junction temperature of complicated components. For this project, since the experimental data of the heat transfer is available, the inside and outside heat transfer coefficients can be determined by lowering the difference between the experimental and GA simulated heat transfer, the method being similar to the aforementioned paper. Thus, through the literature review, a comprehension of the applicability of GA can be made, and the way to model the problem in such a way that it can be implemented in GA can also be found and studied from the papers.

CHAPTER 3
METHODOLOGY

3.1. Implementation of Genetic Algorithm

Observing equation 3 again, it is a problem with one equation and two unknowns. In conventional method, this can be reduced to one equation and one unknown by using a Nusselt Number correlation to first either the inside or outside heat transfer coefficient. However, errors incurred in the correlation for one side of the heat transfer coefficient will have a pronounce effect in determining the other side of heat transfer coefficient.

On the other hand, performing exhaustive search to determine the heat transfer coefficients directly is not an efficient method. This is due to the multidimensional nature of the problem. If it were only one dimensional then the exhaustive search appears to be feasible. Attempting different combinations for the values of ho and hi seem to be a close to impossible task, even with today's modern parallel computers.

GA is an intelligent search method. Used in many heat transfer problems, particularly related to the heat exchanger design problems, GA shows good potential at arriving in optimum solutions. Thus, GA searches for the parameters or solutions that will maximize or optimize a user-defined function. An added advantage is that GA does not require the derivative information in guiding its search. Full-blown or deliberate mathematical modeling is not needed. The basics of GA will be briefly described in the following.

3.2. Encoding

GA works on the encoding of the solution of the problem, meaning that the parameters of a function are encoded in a way that can be manipulated by GA. For instance, in the function of $y = x^2 + 5$, where one would wish to find the value of x that optimize y, x can be encoded as binary numbers, 1 or 0, such as

1	0	0	1	1

This binary representation can be converted to decimal and vice versa. In GA, one binary representation of x is known as individual. A group of individuals is known as population. However, it is important that one establishes a constraint to the values of x, so that GA would 'know' which boundary it should operate at. This is particularly important in real-life problems, for instance in heat exchanger optimization, where pressure drop is one of the parameters to be optimized, a certain amount of pressure drop has to be defined. Otherwise the solution would be impractical and infinite. In the GA that the author has implemented, continuous encoding was utilized, in which the two parameters are encoded in this manner:

200	300

The first cell represents the first parameter in the objective function, and the same goes for the second cell. Such encoding paves the way to solve future problems, where the heat transfer coefficient could reach the range of 1000 W/m.K, and such value is impossible to be represented by binary encoding.

3.3. Fitness Measure

Each individual in GA will be assigned a fitness measure, which measures how good the solution is. In the previous example, the fitness value is the value of y corresponding to the particular value of x. Be noted that the binary values of x has to

be converted to integer values before y can be evaluated. Sometimes the fitness function is unknown and it could be generated by external software, without the user having to derive any mathematical models.

Fitness therefore serves as an indicator of the individuals' performance in GA. The relative fitness of each individual is the individual's fitness divided by the population's total fitness. A larger relative fitness of an individual means that the individual has a better performance or quality, relatively to other individuals. Thus, in Roulette Wheel selection, this individual has a larger chance of being selected.

3.4 Selection

Roulette Wheel selection represents another phase of GA which is the selection process. The selection process is to select individuals that are deemed 'good' into the mating pool. The mating pool will be discussed later. A fitter individual will have a larger slice of pie in the wheel, and as it is spun, it has a greater probability of being selected. Therefore, through the selection process, inferior individuals can be eliminated to give way to better individuals.

3.5 Crossover

In the mating pool, individuals selected will be known as parents. Randomly, two dissimilar parents will be selected to create new offsprings via crossover.

Parent 1: 11001|010
Parent 2: 00100|111

The | line indicates the crossover point and it will also be randomly selected. In this example, the bits that appear after | will be swapped, producing two offsprings:

Offspring1: 11001|111
Offspring2: 00100|010

Crossover exchanges information between the two parents and produce different offsprings as a result. The offsprings retain the good features of the parents and since the selection ensures that 'good' parents are selected, one can expect the average fitness to increase. However, since continuous parameter encoding is utilized, consider:

Parent 1: 200,100

Parent 2: 500,350

The comma is a representative of the cell, as mentioned earlier. To perform crossover, select a crossover site, which is the second cell as depicted. Then,

$$\text{Child 1}_{(2):} = \text{Parent 1}_{(2)} - \text{beta}*(\text{Parent 1}_{(2)} - \text{Parent 2}_{(2)})$$

$$\text{Child 2}_{(2):} = \text{Parent 2}_{(2)} + \text{beta}*(\text{Parent 1}_{(2)} - \text{Parent 2}_{(2)})$$

The number 2 in parentheses represents the value of parent 1 at cell 2, and so forth. Then,

$$\text{Child 1: Parent 1}_{(1)}, \text{Child 1}_{(2)}$$

$$\text{Child 2: Parent 2}_{(1)}, \text{Child 2}_{(2)}$$

after the crossover process. Beta is a random value between 0 to 1. According to Haupt in [6], such crossover method in continuous parameter GA can avoid the child values from being out of the desired range.

3.6 Mutation

Mutation, on the other hand, introduces diversity into the population. In search space whereby many local optima exist, mutation can help the searching process in escaping from the local optima. Do perform mutation, a bit is randomly selected and it is indicated by the bit being underlined:

Individual: 1100111<u>1</u>

In this example, bit 1 at the sixth position from the left is selected. Mutation merely flips that bit into bit 0:

Individual: 11001011

Mutation clearly alters the structure of the individual that has undergone crossover. This can sometimes induce instability in the search process and therefore it should be used sparingly. For mutation in continuous parameter GA, the mutation site is selected, which is underlined as shown:

Individual: 200,100

Then, what needs to be done is the replacement of the value at the mutation site with a random value that is within the desired range.

3.7. Framework/Flow chart of GA

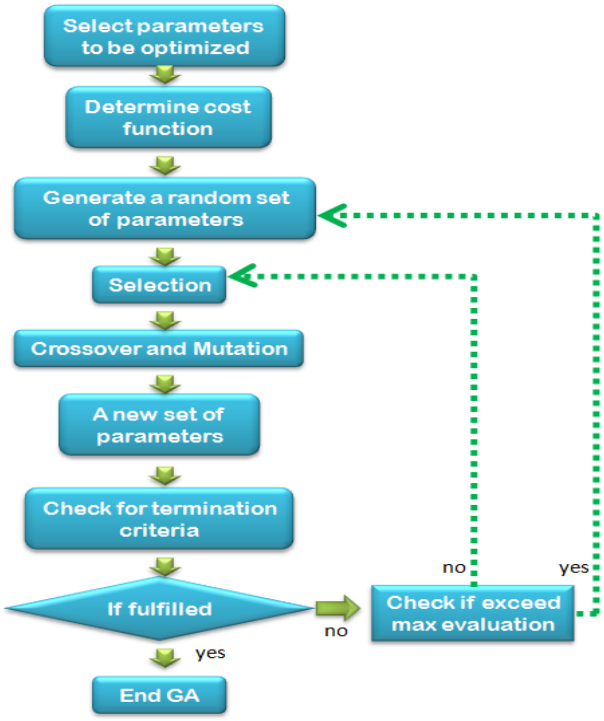


Figure 3.1 Flow Chart of GA

GA is in truth an iterative method and the way it runs can be described by the figure above. All GA processes are covered in the preceding sections. The termination criterion, a search limit imposed by the user, is checked. If that criterion is fulfilled, then the GA iteration is completed. If not, the number of evaluations will be checked.

If it exceeds the maximum number of evaluations, the whole searching process has to be restarted, this time around with another group of random parameters to begin with. If it does not exceed the maximum number of evaluations, the new parameters will be brought to the selection process again as shown in the framework.

3.7.1. Encoding

Encoding refers to the representation of parameters to be optimized. There are several encoding methods such as binary and real number encoding. In this problem, the real number encoding method is employed as it provides an easier form of implementation. The encoding takes the form of an array of numbers, whose size is determined by the numbers of parameters. Using this array, various manipulations in GA such as selection, crossover and mutation can be implemented.

Ultimately, it is of great interest to know the form of the Nusselt Number correlation as this provides an opportunity for greater prediction of heat transfer rate, thereby optimizing the area of heat exchanger with respect to the heat transfer rate.
However, it should be noted that GA is not an elixir to any optimization problems. Much would depend on how one could provide essential information for the GA to search in a manner that one desires. That is, as evident from the previous results of MNCGA, one needs to modify the cost function and fitness evaluation so as to enable GA to land at feasible answers.

Even better for GA, if it could ‘overlook’ the overall trend surrounding the search, especially in curve-fitting type of functions. For example, the correlation involved would depend on several non-dimensional numbers, namely Nusselt Number and Prandtl Number. It would be more efficient for GA to perform the search if these terms are included as part of the searching mechanism.
Instead of having to find individual pairs of inside-outside heat transfer coefficient values, and form Nusselt number correlations using curve-fitting software, these cumbersome steps could be alleviated via searching for the coefficient values related to each non-dimensional terms.

For instance, a typical form of the Nusselt number correlation is considered:

$$Nu = C \times Re^m \times Pr^n \quad (7)$$

The C, m and n values can be found via GA. Subsequently, the U value can be obtained and the heat transfer rate value, denoted here as Q_{GA}, can be calculated. Q_{GA} will be used in the cost function.

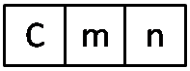


Figure 3.2 The encoding of the parameters, in an array

The array of parameters are also known as chromosome, and through the processing in GA, the optimal C, m and n values can be estimated. As mentioned in the methodology section, at the beginning a group of these arrays or chromosomes will be generated, with their values randomly dispersed, in the range specified by the user.

3.7.2. Fitness Measure/Function

As mentioned previously, the cost function is required to measure the quality of the parameters. It is also used in the selection process to eliminate inferior parameters. In this problem, the cost function can be employed as shown in equation (4). Literally, it would imply that parameters that produce the heat transfer rate which is close to that given as experimental heat transfer rate values would be considered to be of higher quality and feasibility of selection of these parameters are much higher, and vice versa.

$$Fitness\ Measure = \frac{1}{abs(Q_{experiment} - Q_{GA})} \quad (8)$$

3.8. Niching

Imagine an environment inhibited by various species of nature. In accordance to the herd rule, animals of the same type will compete for food, and eventually the strongest animal of all in that category/species will dominate and get the biggest

portion of the food. However, there will be stronger animals of other types and eventually the weaker ones will be dominated by the stronger types.

Humans are a good depiction of the above scenario. Being the most intelligent species, humans have control on the environment and its other inhabitants, not the other way around. Hence, niching in GA aims to promote inter-niche (in biological terms meaning the competition between the same species of animal) competition. Of course, there will be situations whereby the competition will also be held between animals of different types but this is inevitable in nature. By promoting inter-niche competition, it is hoped that the GA can have individuals that converge to various local optima (winners in the respective niche) and with this stepping stone, the global optima could be achieved.

3.8.1. Multi Niching GA (MNCGA) [7]

MNCGA was developed by Cedeno and Rao from the Hewlett Packard Laboratories. The aim of MNCGA is to promote competition among similar individuals. The main purpose of MNCGA is to ameliorate the selection pressure caused by the fitness proportion selection/roulette wheel selection in the normal GA. This objective is achieved by encouraging mating and replacement within members of the same niche while allowing some competition for the population slots among the niches. By doing so, this offers a balance between exploration where no restrictions are imposed (this is evident in the fitness proportionate selection process), allowing freedom in exploring more individuals of different type in the search space, while converging to the best individuals in different niches. MNCGA is also utilized, in addition to the normal GA, in this report.

It is best to describe the MNCGA using its algorithm.

1. Crowding Selection

This step replaces the fitness proportionate selection. Instead of choosing the fitter individuals to survive into the next generation, crowding selection first selects a parent, denoted by Ix. Then its mate, Iy is selected from a small group of individuals

with the size s (not from the entire population) by virtue of their phenotypic distance. This distance can be defined as the difference between the decoded values of the individuals. The smaller the difference, the more similar the individuals are.

Individuals in the group of size s are chosen randomly without any restrictions from the population, and they can be picked with replacement. Thus, with crowding selection, every individual in the population has a chance to mate, unlike the fitness proportionate selection method whereby individuals of relatively higher fitness get chosen for most of the time. This process also allows a great deal of exploration while exploiting the similarity between individuals from the same niches. By having individuals of the same niche to mate, one is ensured convergence to local optimum. On the other hand, there is also the probability that individuals of slightly different niches will mate, but there is no reason to fret on this because this is where the exploration comes in. Having selected the mate, one just has to perform the crossover operator on the parents. If there are two offsprings, one can either opt to insert the two offsprings back into the population based on the WAMS operator to be described later, or select the best offspring of the two to be inserted back into the population.

2. Worst Among Most Similar (WAMS) Operator

This operator is performed after the crowding selection step. This step aims to replace the worst individual based on the WAMS operator with the offspring that is produced from the crowding selection operator. First, we need to select f number of crowding groups with g number of individuals from the population. Then one individual from each f number of crowding factor groups is selected to form f number of individuals as candidates for replacement by the offspring. The selection of f number of candidates for replacement is based on the similarity with the offspring. One member which is of the lowest fitness in the crowding factor group will be eliminated and replaced by the offspring.

For a better depiction of the WAMS operator,

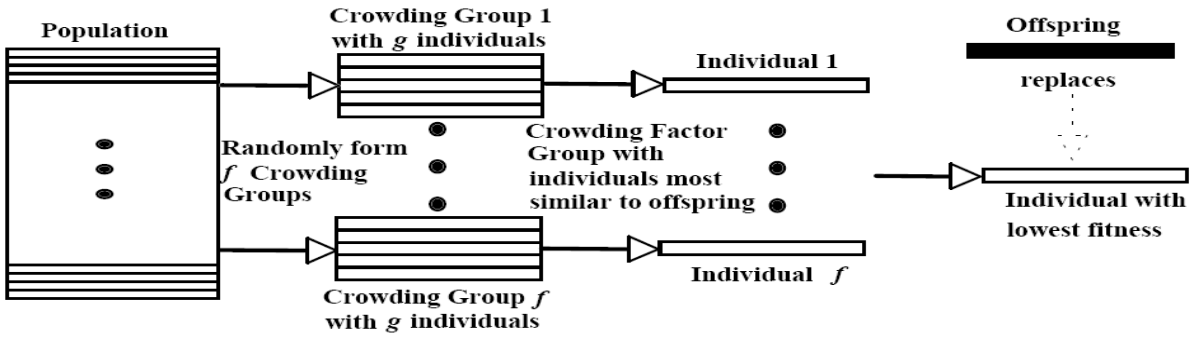


Figure 3.3 **WAMS operator**

After the offspring is inserted into the population, it will also compete for survival with other individuals. In WAMS operator, the offspring will most likely replace a lower fitness individual from the same niche. On the other hand, there exists the possibility of the offspring replacing the individual of higher fitness (with respect to the offspring) from the same niche or other niche. This provides the balance to the MNCGA by allowing diversity to exist within the population.

The MNCGA framework is as follows:

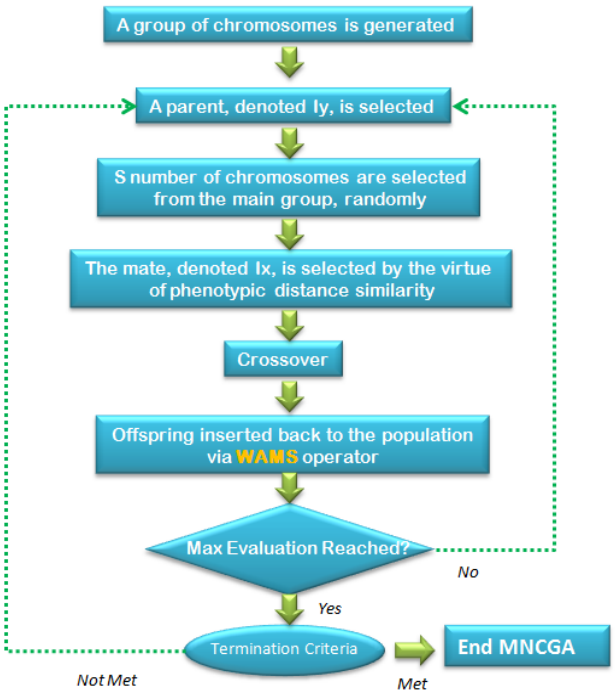


Figure 3.4 Flow Chart of MNCGA

3.9. Project Execution

3.9.1. Project Activities

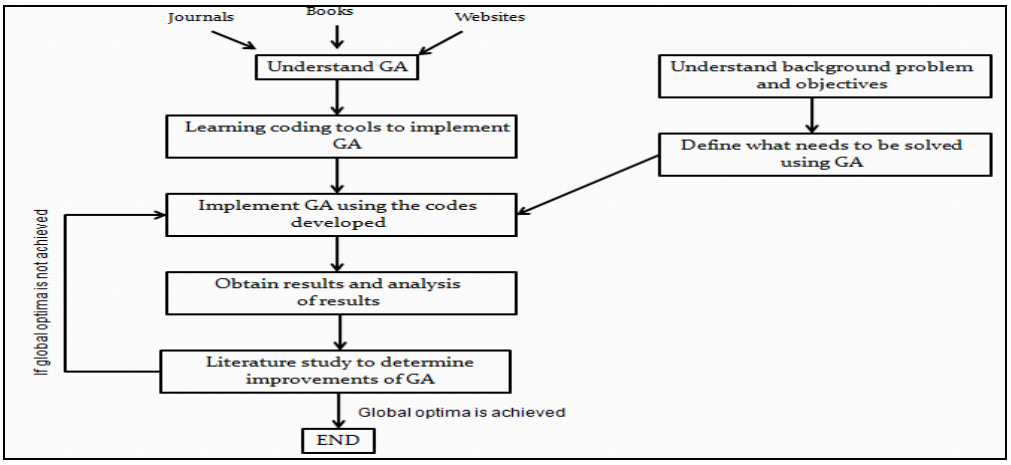


Figure 3.5 The Framework of the Project

The figure above depicts the main activities or methodology to achieve the objectives of this project. After establishing the problem statement and objectives, the theories and information related GA are obtained via journals, text books or websites, and these information would provide the author an idea on how to execute the project, in relation to the objectives. Undoubtedly, the tools in which the GA can be coded upon are of essence. Microsoft Visual Basic provides the means of which

the GA can be coded to the user. Utilizing the coding tools and theories of GA, the codes needed to achieve the objectives are spelt out from scratch. No built-in GA function is available in VB as VB is a coding tool that is similar to C++ (object orientated programming tool).

Using the theory of GA, the model can be coded in a procedural manner. The main loop consists of application of GA until it reaches the number of generations specified by the user. When the termination criterion is not reached, GA will restart from the beginning.

Having done with the GA model, the industrial data is required. After the Nusselt number correlations' exponents and coefficients are obtained from GA, the heat transfer values can be obtained and compared with the experimental heat transfer values to observe error incurred between the GA simulated heat transfer rate and the experimental (actual) heat transfer rate. The aim is then to ensure the degree of error is small as compared to the industrial application's form of Nu correlation.

3.9.2. Gantt Chart

FYP 2 Gantt Chart

Final Year Project - Gantt Chart
Using Genetic Algorithm to Obtain Heat Transfer Coefficient

No.	Details	Status	Week												
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Programming work	done														
2	Testing, verification and analysis of results	done														
3	Progress report 1 submission	done														
4	Progress report 2 submission	done														
5	Seminar	done														
6	Poster exhibition	done														
7	Submission of dissertation final draft	in progress														

Figure 3.6 FYP 2 Gantt Chart

The figure above shows the details that make up the entire activities for the FYP 2, current status and duration taken (indicated by yellow boxes). Since the current stage represents the last phases of FYP 2, most tasks have already been done or performed.

The use of Gantt chart enables the author to track and monitor the project progress, thus ensuring all deliverables are handed in on time.

3.10. Summary of the Code

Basically the code structure is procedural, meaning that all functions are executed from beginning to the end. The class object in VB is utilized thoroughly to aid in the coding of the GA. Such method would allow the chromosome structure to be codified in a simpler manner, and could be easily manipulated by the various GA processes in the later stage. Furthermore, the graphical user interface options could allow for easier view of the results, as well as the input of the data, in which the data just needs to be inserted into a built-in spreadsheet in VB, and a few lines of codes would allow these data to be included into the GA code for process.

CHAPTER 4
RESULTS

4.1. Overview of Results

In this section, several results will be discussed. Firstly, the results of two simple function minimization problems will be presented, with the aim to make initial comprehension easier to the reader. A sample problem in the Incropera's heat transfer text book will be attempted as well. Then, the results of the exponents and coefficients that are related to the Nu correlation will also be presented, together with the RMS error versus the data point curve, to establish the difference between the GA-formed Nu correlation and Industrial-given Nu correlation.

4.2. Function Minimization Problem

Two function minimization problems were attempted and they were:

1. $F_1(X,Y) = X \sin 4X + 1.1Y \sin 2Y$
2. $F_2(X,Y) = (X^2 + Y - 11)^2 + (X + Y^2 - 7)^2$

In which X and Y were the parameters to be operated by GA in order to arrive at the global minimum of the two functions. To make observations easier, the results will be elaborated based on the two functions. The fitness landscape which represents the level of fitness, based on the function itself, can be shown as in the next two figures:

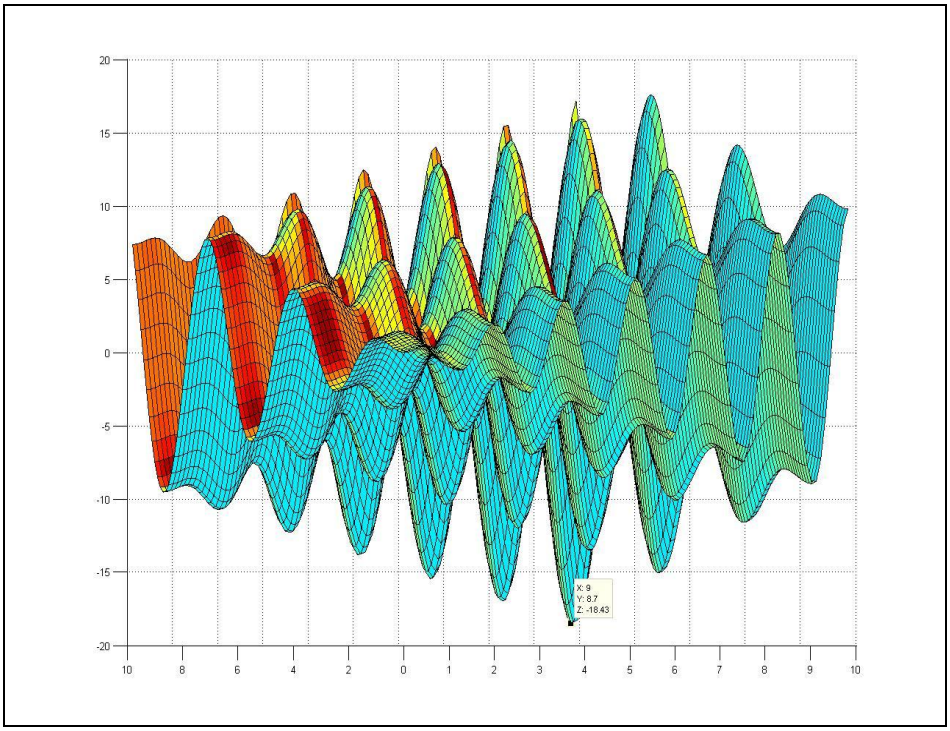


Figure 4.1 Fitness Landscape of Function 1

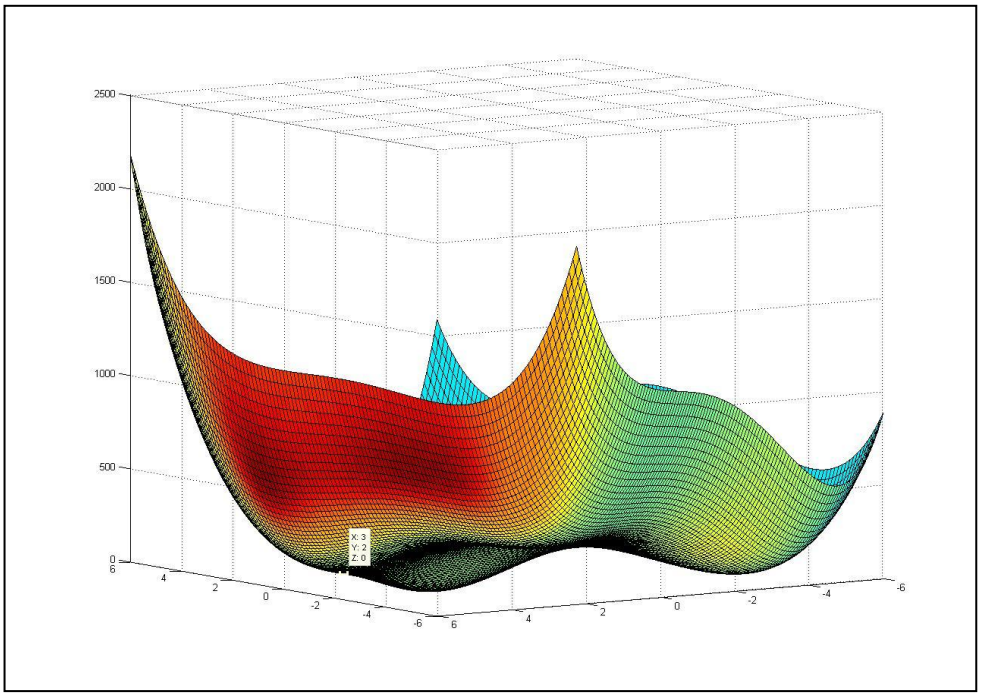


Figure 4.2 Fitness Landscape of Function 2

As

evident from the fitness landscape of these two functions, there exist multiple local optima (in the form of valleys as can be observed from figure 4.1 and 4.2), in which normal optimization methods like hill climbing often find the searching process undone at this local optima points. GA, being a global search method, is capable to search for the solutions are very close to the global optima.

4.2.1. Function F_1

For simplicity's sake, only the first 20 results are shown, where they have been arranged

in descending order, from high to low final fitness values:

Table 4.1 Results of Function 1

X random	Y random	fitness start	X final	Y final	fitness final
9.572588936	6.433529	-0.001	9.051674	8.664702	-18.54283533
1.261968385	9.812288	-0.001	9.051674	8.664048	-18.54273976
9.572588936	6.433529	-0.001	9.051674	8.663688	-18.54268026
6.912926113	1.114203	-0.001	9.051674	8.663415	-18.54263168
4.819266617	6.926884	-0.001	9.051674	8.663024	-18.54255718
0.475101327	8.238554	-5.853562972	9.051674	8.662531	-18.54245506
7.530439048	2.349229	-9.828050884	9.051674	8.661787	-18.54228333
1.986808764	1.261968	-0.001	9.051674	8.659173	-18.54151165
9.516757023	6.321865	-0.001	9.051674	8.658768	-18.54136848

The first function F_1 is to be minimized with the conditions or constraints of {0 : X : 10} and {0 : Y : 10}. Due to the fact that this is a minimization problem, all negative fitness values are multiplied with negative one to make computations and coding easier. However, if fitness values computed are positive, they are imposed with penalty and end up as:

$$F_1 = 0.001$$

Therefore, unfeasible solutions of X_1 and X_2 are slowly eliminated. Another observation that can be made is based the graph of average population fitness vs generation:

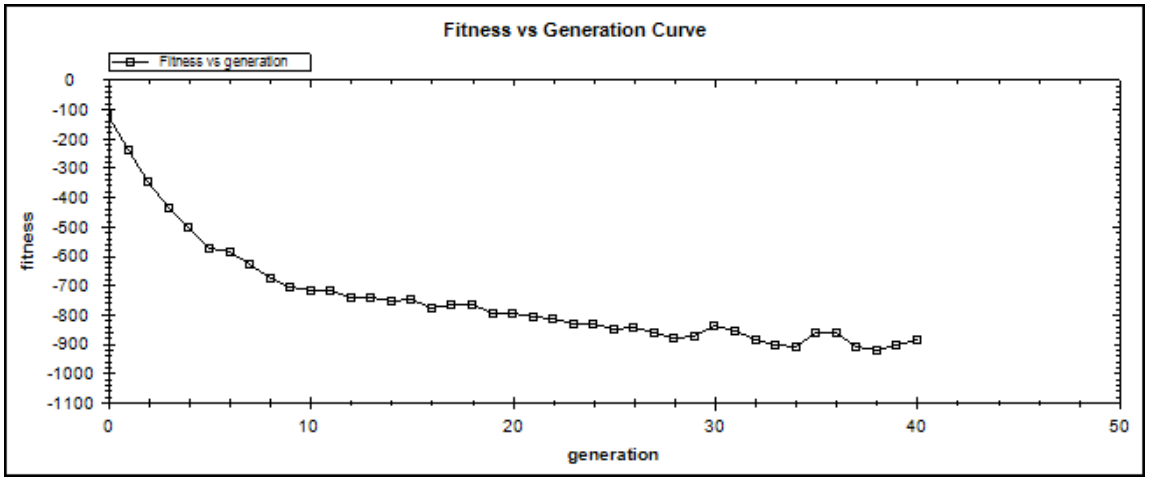


Figure 4.3 Fitness vs Generation Curve for Function 1

The cost function decreases with the increase of generation, corresponding to the objection of minimization. This is due to the elimination of unfeasible solutions and the recombination via crossover operations to produce fitter individuals or parameters.

4.2.2 Function F_2

For simplicity's sake, only the first 20 results are shown, where they have been arranged in descending order, from high to low final fitness values:

Table 4.2 Results of Function 2

X random	Y random	fitness start	X final	Y final	fitness final
1.492012	1.245175	0.01382289	2.999944	2.000072	8204025.02
5.764157	3.789466	0.00117782	3.000188	2.000017	727727.441
3.941319	0.143791	0.03214277	3.000211	1.999829	701857.273
1.492012	1.245175	0.01382289	3.000209	2.000017	591854.683
0.143791	4.548734	0.00429639	3.00017	1.999675	568861.035
3.033649	4.328451	0.00445363	3.000211	1.999711	539998.317
4.697031	1.655214	0.00617586	3.000211	1.999681	491320.813
0.918053	0.097259	0.00724233	3.000188	1.999611	414334.707
1.655214	1.571579	0.01886659	3.000253	2.000017	406862.237

The second function F_2 is to be minimized with the conditions or constraints of {-6 : X : 6} and {-6 : Y : 6}. Due to the fact that this is a minimization problem, all fitness values are to be in such a form, to make

$$\text{Fitness} = 1 / F_2(X , Y)$$

computations and coding easier. Another observation can be made with respect to the graph of average population fitness vs generation:

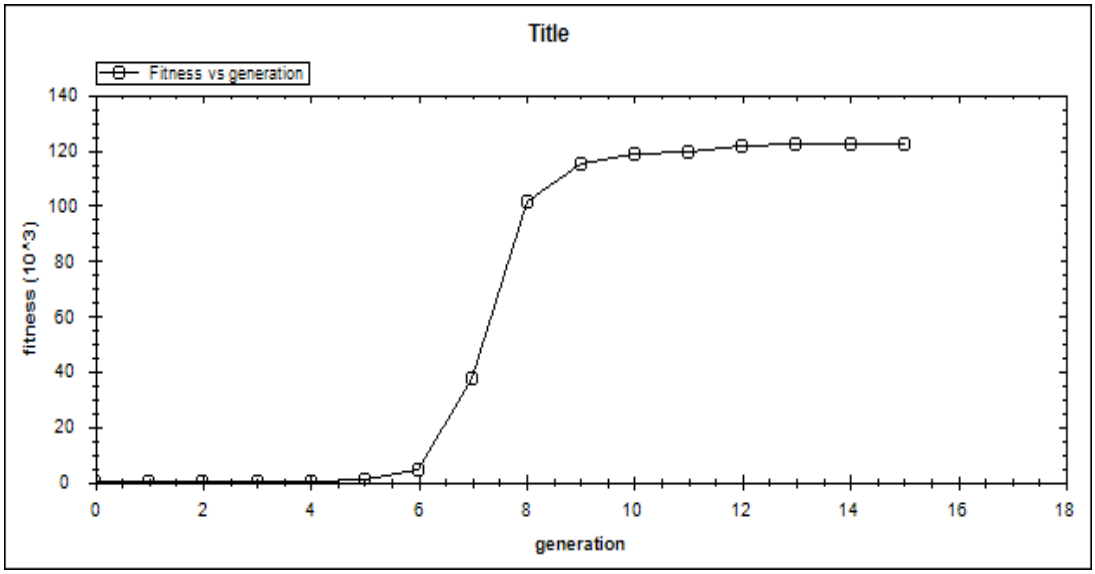


Figure 4.4 Fitness vs Generation Curve for Function 2

The fitness of the population increases from generation to generation. In truth, the global minimum for this function is zero, however, due to the nature of the manipulation of continuous numbers, the value obtained is close to zero, where X and Y are 3 and 2 respectively.

Based on the fitness versus generation curves, it can be deduced that through the fitness evaluation and selection methods, the best solutions are preserved from time to time. Using GA processes such as crossover and mutation, varieties are introduced into the pool of solutions. Variation is important and crucial as it allows GA to sample with numerous solutions. The selection method will not only preserve the best individuals, it will also serve to check that the sampling is done in the correct 'direction', meaning that the search will be guided towards the area that has higher probability of global minimum or maximum. For the application of heat exchangers, the developed GA code will serve as a backbone to solve these problems and hence achieving the prescribed objectives.

4.3. Incropera's Textbook Problem

There is a problem related to the concentric heat exchanger given in Incropera's heat transfer text book [4]. In that particular problem, the objective was to determine the length of the heat exchanger. For the project that the author undertakes, the problem is reversed – in which the values of the inner and outer heat transfer coefficients are sought out, rather than determining the coefficients first, then the geometry.

However, in Incropera's example of calculation, many simplifications were made, especially in the calculation of the overall heat transfer coefficient value, U. Thus, all calculation are made or performed exactly as shown in the example problem.

4.3.1 Results

Table 4.3 Example Problem Results 1

h_o	h_i	fitness
2264	38.42	47.77846
2493	38.36	42.32586
2073	38.48	65.80698
2335	38.4	51.13799

2164	38.45	81.54128
2373	38.39	224.2397
2452	38.37	230.055
2299	38.41	239.0287
2776	38.3	157.9888
2581	38.34	723.5419

The results above are obtained with the terminating criteria set at 0.01, which is the allowable difference between $Q_{GA\text{-}simulated}$ and $Q_{experimental}$. Any combinations of h_o and h_i that yield the difference below than that criteria will result in the termination of the searching process.

In the example, h_i and h_o values are 2250 W/m2.K and 38.4 W/m2.K, respectively. However, a simple back-calculation will reveal that the Q value generated is not 8524 W, but 8518.773611 W, as some values are lost when the round-up of numbers is performed.

If the $Q_{experimental}$ for which the comparison basis is set at is given by the value of 8518.773611 W, the following results are obtained:

Table 4.4 Example Problem Results 2

H_i	H_o	Fitness
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711
2250	38.4	2515711

The results give a clear indication that for the application in this project, GA mainly serves the purpose of data fitting. It does not really recognize which heat transfer coefficients are more superior, in the context of theoretical aspect.

4.4. Application on Heat Exchangers

GA is applied to determine the coefficients associated with the Nusselt number correlations for two geometries: (i) a straight tube concentric heat exchanger and (ii) a brazed plate heat exchanger. The root mean squared (RMS) error is then plotted against various data points obtained from the industrial data, to provide a comparison of the rms error between the industrial-given Nusselt number correlation and GA-generated Nusselt number correlation, as well as the MNCGA-generated Nusselt number correlation.

4.4.1. Straight Tube Concentric Heat Exchanger

Equations (9) and (10) are the industrial-based Nu correlations for the concentric tube heat exchanger:

$$Nu_{Industry,tube-side} = 0.048 \times Re^{0.733} \times Pr^{0.333} \quad (9)$$

$$Nu_{Industry,annulus-side} = 0.0201 \times Re^{0.733} \times Pr^{0.333} \quad (10)$$

a) Application via GA

$$Nu_{GA,tube-side} = 0.0482 \times Re^{0.7326} \times Pr^{0.333} \quad (11)$$

$$Nu_{GA,annulus-side} = 0.02001 \times Re^{0.7332} \times Pr^{0.333} \quad (12)$$

Using the GA-based Nu correlations, the heat transfer rate could be obtained. The RMS error is then the difference between heat transfer rate predicted by the GA-based Nu correlation and industry Nu correlation.

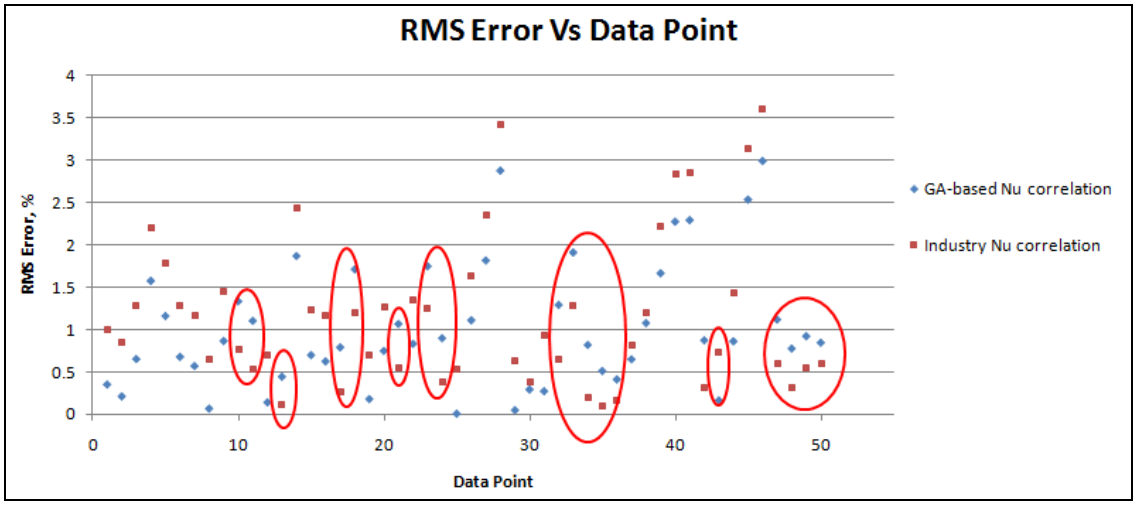


Figure 4.5 RMS error vs data point for straight tube heat exchanger (GA)

b) Application via MNCGA

$$Nu_{GA,tube-side} = 0.04784 \times Re^{0.7325} \times Pr^{0.333} \quad (13)$$

$$Nu_{GA,annulus-side} = 0.02006 \times Re^{0.7325} \times Pr^{0.333} \quad (14)$$

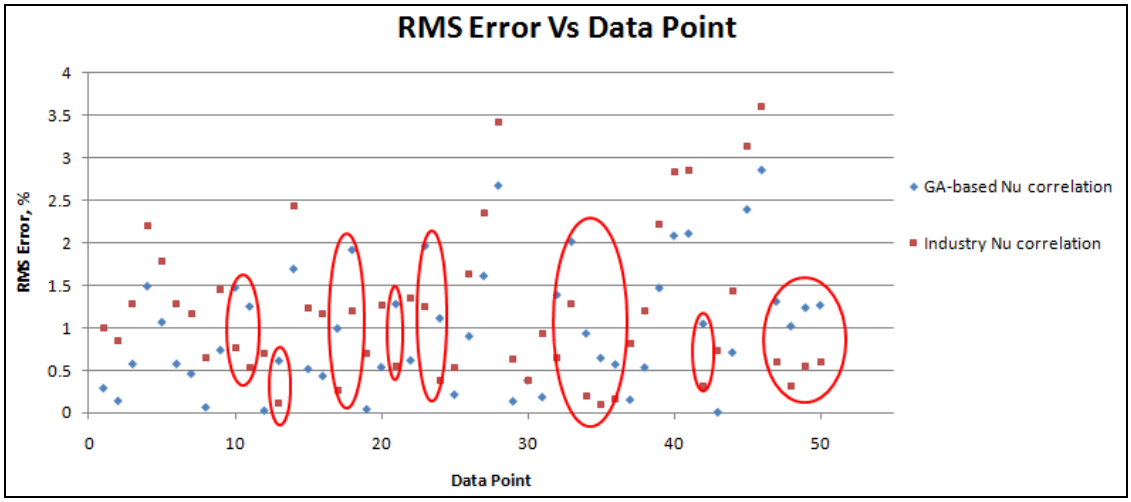


Figure 4.6 RMS error vs data point for straight tube heat exchanger (MNCGA)

Based on the figure 4.5, out of 50 data points, 32 showed an improvement, which implies a reduction in error in predicting the heat transfer rates. The remaining data points are not improved (indicated by red circles), but the error range in which these data points occur is below 2%, which is still well-accepted. In addition, the highest RMS error has also been reduced from 3.7 % to 3 % in the case of the GA-based Nusselt number correlation.

Similarly, the correlation produced as a result of running the MNCGA algorithm generates 32 points which are improved over the industrial Nu correlation. Compared to the GA-based Nu correlation, the highest RMS error for the second application using MNCGA has been reduced from 3.7 % to 2.8 %. Hence, MNCGA is able to find the parameters which are even closer to the global optimum.

In essence, both GA and MNCGA generate correlations which incur less prediction error than the industry-based Nu correlation, in a broad sense. It is rather difficult to have all data points improved, due to the quality of the data. GA cannot account for all experimental errors since what GA really does is reducing the fitting error between the experimental heat transfer rate and GA-simulated heat transfer rate, and in this case, the lowest error is incurred for the improvement for all data points, rather than a few data points.

4.4.2. Brazed Plate Heat Exchanger

Equations (15) and (16) are the industrial-based Nu correlations for the brazed plate heat exchanger:

$$Nu_{Industry,hot-side} = 1.906 \times Re^{0.586} \times Pr^{0.333} \times \frac{\mu_b}{\mu_w}^{0.14} \quad (15)$$

$$Nu_{Industry,cold-side} = 0.579 \times Re^{0.586} \times Pr^{0.333} \times \frac{\mu_b}{\mu_w}^{0.14} \quad (16)$$

a) Application via GA

$$Nu_{GA,hot-side} = 1.757 \times Re^{0.589} \times Pr^{0.333} \times \left(\frac{\mu_b}{\mu_w}\right)^{0.14} \quad (17)$$

$$Nu_{GA,cold-side} = 0.572 \times Re^{0.589} \times Pr^{0.333} \times \left(\frac{\mu_b}{\mu_w}\right)^{0.14} \quad (18)$$

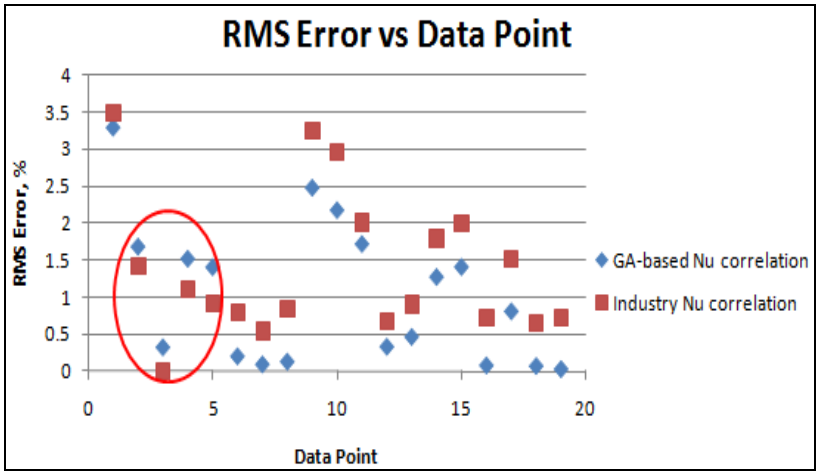


Figure 4.7 RMS error vs data point for brazed plate heat exchanger (GA)

b) Application via MNCGA

$$Nu_{GA,hot-side} = 1.801 \times Re^{0.589} \times Pr^{0.333} \times \left(\frac{\mu_b}{\mu_w}\right)^{0.14} \quad (19)$$

$$Nu_{GA,cold-side} = 0.568 \times Re^{0.589} \times Pr^{0.333} \times \left(\frac{\mu_b}{\mu_w}\right)^{0.14} \quad (20)$$

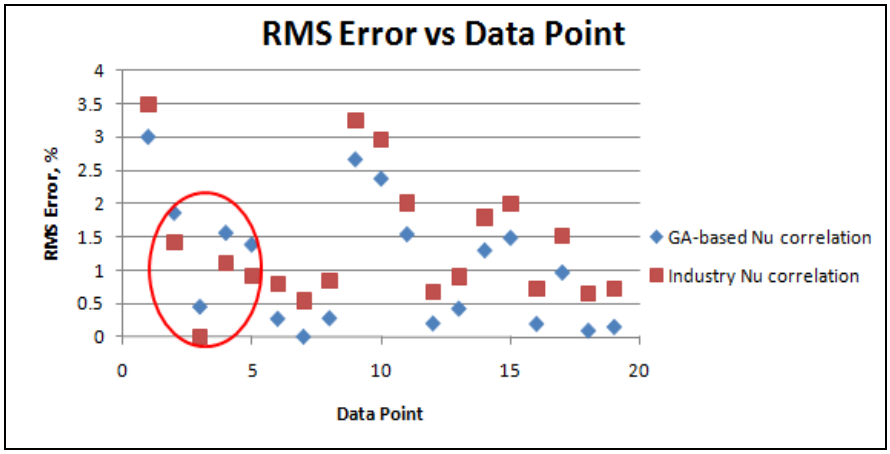


Figure 4.8 RMS error vs data point for brazed plate heat exchanger (MNCGA)

Based on figure 4.7, out of 19 data points, 15 showed improvements in predicting heat transfer rate, whereas 4 were not improved (indicated by red circle). Again, the 4 unimproved data points are still below 2 %, a well-accepted limit. The highest RMS error is reduced from 3.5% to 3.3 %.

Meanwhile, figure 4.8 shows that similar to the previous case (GA application) for the application of MNCGA, 15 showed improvements in predicting heat transfer rate, whereas 4 were not improved. However, the RMS error is reduced from 3.5 % to 3 %, in the case of MNCGA application.

From the results of both heat exchanger applications, GA and MNCGA produce an almost identical performance, with MNCGA slightly better off in the reduction of highest RMS error.

CHAPTER 5
DISCUSSION

5.1. Overview

In this section, the coding structure will be explained and discussed as this is the essence of the work done by the author. It is also hoped that such discussion will provide the readers a rough idea, should they decide to utilize and code GA in the future. The results obtained will also be analyzed and discussed.

5.2. Explanation of the Code/Coding Structure

In this section, some explanations will be given with regards to what is actually happening based on the implemented codes. The algorithm is easily available in all literature, but the implementation via coding is heavily dependent on the user. For instance, it is easy to say that an individual can be eliminated or selected for survival in the selection step. However, how can the actual elimination or selection done, in coding terms? Thus, the explanation can give one an insight into the actual process or translation of those algorithms into real, practical use.

5.2.1. Preprocess

- This step is to produce random values of ho and hi.
- Random values are generated based on

Value = random * (hi – lo) + lo
in which hi = upper bound value, lo = lower bound value, random = a random number between 0 to 1

5.2.2. Roulette Wheel Sorting

- This step is to arrange the various combinations of ho and hi values to form the various sections or ‘pie’ in the roulette wheel.
- First the total fitness will be calculated, based on the contributions from each ho – hi combination.
- The relative fitness will be calculated for each ho – hi combination, by dividing the fitness value with the total fitness value.

- The relative fitness values will be sorted in an ascending manner, with their associated numerical arrangement of ho – hi kept in an array.

5.2.3. Roulette Wheel Configuration

- In this step, the roulette wheel will be formed, based on the numerical arrangement of ho – hi values stored in the array shown in the previous step.

5.2.4. Selection

- Having formed the roulette wheel, the selection in which the survival of which ho – hi values will be determined.
- From the first combination of ho –hi values, a random number between 0 to 1 will be generated.
- If the random number falls between 2 designated values in the roulette wheel, the numerical arrangement of the ho – hi value that results in those 2 designated values will be selected and stored in another array, given as status array for explanation purposes.
- Otherwise, the selection will be ignored and another portion in the roulette wheel will be tested.

5.2.5. Create New Generation

- Like its name implies, a new breed of population will be created, based on the survival mechanism selected in the previous step.
- Using the array, status array as indicated previously, new copies of population will be created using the clone function available in VB.
- The creation of new generation will proceed until the number of population has been exceeded in the iteration.

5.2.6. Choose Mate

- This step is to select mating partners for the individuals in the population.

- First, an array of numbers, based on the number of population, will be created to store the numerical arrangement of each individual. As an example, individual 1, 2, 3 will be stored as 1, 2, 3 in the array.
- A number in the arrangement will be selected and removed from the array. The selection of a number to partner the previously selected number will be done, similarly.
- Thus, a pair of different numerical arrangement can be obtained. By diminishing the numbers in the array, an individual cannot be allowed to mate for more than once.

5.2.7. Crossover

- Based on the pairing of numbers performed in the previous step, each pair of individuals will be selected.
- The crossover site will be generated randomly.
- For the continuous number crossover, the crossover operator given by Haupt will be performed.

5.2.8. Mutation

- A random number between 0 to 1 will be generated. If the specified mutation probability exceeds the random number, mutation will be performed. Otherwise, this step will be skipped.
- A random mutation site is chosen, and a random number based on the user specified range will replace the current number at the mutation site.

5.3. Results Discussion

1. Based on the results of the function minimization problems, the following discussions can be made:

a) GA is a robust search model, in which it can perform search even in the fitness surface with many local optima. The strength in GA lies with the fact that it works on many potential solutions, which are randomly generated. As observed, only solutions which are fit stand a higher chance to survive, and when they are combined

via crossover operator with other solutions, they will improve in terms of fitness from generation to generation. On the other hand, mutation allows the solution to 'jump' out of local optima and this introduces variety to the population.

b) Variation is extremely important in GA. This implies that at the beginning, there should be enough solutions of different values for GA to sample with. High variety would lead to more efficient crossover operations, where else similar individuals or solutions that cross over would result in stagnation, i.e. the solutions do not improve much. In fact, with fitness landscape of many local optima, variation of individuals or solutions is important to ensure that the searching process is not biased towards certain optima only.

c) The Roulette Wheel selection process eliminates weaker individuals, leaving stronger individuals to dominate. However, if the initial population is insufficient or lacking in variety, the selection pressure would leave the GA to concentrate on suboptimal solutions. The role of mutation is therefore important here, to lead the search out of local optima. However, based on the work by Cedeno and Rao[7], a multi-niching Genetic Algorithm can be implemented to allow various local optima to emerge throughout the generations. Towards the end, the algorithm will slowly focus towards the best among the local optima, i.e. Global optima. This is why through this research, the MNCGA will be utilized on the heat exchanger application (Nu correlation) too. There is a good prospect of including elitism, a selection mechanism where the fittest individual will be survive in the next generation, at this stage where the multiple local optima are emerging. Elitism can be used to direct the search towards the global optima among the local optima or hills.

2. Based on the results of the text book example problem, the following discussions can be made:

a) If the target $Q_{experimental}$ is 8524 W, consider two results that have been obtained:

h_o	h_i	fitness
2229	38.43	31.78333

2581	38.34	723.5419

As can be seen, the fitness measure of the first set of ho-hi combination is not as high as compared to the second set, though the ho and hi values might be closer to 2250 and 38.4, the optimal values as provided in the example. Despite the fact that the first set of ho-hi combinations are closer to the optimal values, they might not fit as good as the second set to the Q value of 8524 W.

b) One deduction to this observation would be GA will merely search for ho-hi values that could best-fit the heat transfer value, Q. This is why when 8528.358373 W is chosen as the Q for fitness measure purpose, the following result could be obtained and no other ho-hi combinations can be better than the ho-hi combination resulting from the fit with this Q value data.

h_o	h_i	fitness
2250	38.4	2515711

The difference between 8528.358373 W and 8524 W is about 0.05 %, but it could make a difference when the data fitting of ho-hi combinations to the given Q is concerned. This strengthens the fact that GA could not operate more than what it is expected, as it tries, mathematically to reduce the error between the experimental Q and GA-simulated Q values.

c) Thus, one cannot discount the possibility that the various combinations of ho and hi that result in the lowest error between $Q_{GA\text{-}simulated}$ and $Q_{experimental}$ can be found, but they do not produce good correlation. The major limitation here lies in the experimental heat transfer rate. With lower experimental error, the chances of having better values of heat transfer coefficients for better Nusselt Number correlations are higher.

3. Based on the results of the exponents and coefficients in the Nu correlations of the heat exchangers, the following discussion is made:

a) As mentioned earlier, GA is unable to account for any variations or uncertainties in the experimental data. GA can only process those data in accordance to the fitness measure and undoubtedly, it is not possible for GA to detect any errors in the data. The major limitation here lies in the experimental heat transfer rate. With lower experimental error, the chances of having better values of heat transfer coefficients for better Nusselt Number correlations are higher. Observing the RMS error versus data point graphs, in truth the experimental error is well within the established engineering error limit of 5 %. However, if this could be improved in the future the quality of the data fed to GA will also be enhanced and as a result the correlations could be improved as well.

b) GA provides a framework which can be the basis for searching parameters that would optimize certain functions. However, the manner in which the GA conducts the search is very much dependent upon the user. GA can only find values that would either maximize or minimize certain functions but these values are influenced by the quality of the data given. The errors in the Reynolds number and Prandtl number have a pronounced effect in the prediction accuracy of the heat transfer rate. Then again, this is not something that is within the control of GA.

c) The basic GA and MNCGA produce results that are of different level accuracy. As can be observed from both heat exchanger applications, the highest RMS error has been reduced even more in the case of MNCGA as compared to the basic GA. In fact, out of the improved 32 data points in basic GA and MNCGA, MNCGA could further reduce the prediction error of 22 data points. This implies that MNCGA possesses the ability of being able to locate the solutions even closer to the global optimum. However, the improvement rate is not very significant. Thus, for future problems, basic GA could still be implemented. It is much easier to understand and poses lesser parameters for control as compared to MNCGA.

d) As for the Nu correlations in the straight tube and brazed plate heat exchangers, majority of the data points were improved. For the few data points that could not be improved, the author has attempted to form the correlation based on those data points. However, it is not possible within the feasible and typical search limits as the error criterion value is rather high, eventually after the search, meaning that the exact

opposite occurs where a majority of the data points could not be improved. In the case of a brazed plate heat exchanger, the geometry on adjacent plate channels is identical. This involves the plate wavelength, the amplitude of the plate's waviness and the Chevron angles. Unlike in the case of the fully developed turbulent pipe flow, no well-established correlations are available. Heat transfer coefficients are determined here by holding the flow constant on one side while the velocities and Reynolds numbers are varied over the turbulent regime of flow on the other side. By extrapolating to zero resistance on the side of the variable flow rate, the heat transfer coefficients are determined for the constant flow side. Because of the greater tortuosity of the flow in the plate spaces and the uncertainties in the pattern of flow, the correlations derived have a higher inherent error.

An advantage with the plate-frame exchangers is that, due to the similarity of flow paths on either side, the form of the correlations on both sides could be taken to be identical, with the exponents for each term being identical. However, the heat transfer coefficients are not expected to be the same on either side even at the same flow rate. This is purported to be on account of the difference in the diameter at the entry for the two fluids.

CHAPTER 6
CONCLUSION AND RECOMMENDATION

Through literature review, it can be concluded that GA is indeed a powerful optimization tool. Traditional optimization methods often get trapped in local minima as the local 'hill' is perceived to be the peak and no more improvements with regards to the optimization can be made. GA, on the other hand, relies on operators such as crossover and mutation to get over those local 'hills' in order to achieve global optima. GA does not require derivative information or full-fledged mathematical models, which is extremely flexible since fitness functions evaluation can be obtained from external software. This flexibility makes on-the-fly optimization possible. The advent of more powerful computers enables the searching of GA more efficient and faster. The implicit parallelism of GA can also take advantage of these modern machines as the searches can be distributed or covers a large search space.

This project aims to utilize the capabilities of GA in arriving at the coefficients and exponents associated with the plain tube and brazed plate heat exchangers. Since GA is capable of arriving at the optimum solution, the degree of 'fit' between the experimental heat transfer rate and GA simulated heat transfer rate can be extremely good. This method reduces the errors incurred when using established correlations such as Pethukov-Popov, Sieder-Tate or Gnielinskis's correlations. These correlations encounter error in the form of experimental error. Based on the results, the present correlations used in the industry have been improved over by using GA. This could not go on without the observation that not all the data points could be improved as a result of using GA. Rather, a majority of the data points were improved. This can be attributed to the quality of the data, as the quality of the data fitting (which produces the optimal values of the exponents and coefficients of the Nu correlations) would depend very much on the quality of the data itself. As mentioned earlier, GA does not require full-blown mathematical modeling, yet GA would not be able to tell if the results of the data are good or not, as the only measure of fitness is with respect to the heat exchange rate data.

The recommended future work is to conduct the modeling of the errors involved in the experiment to obtain more accurate results by reducing the error band of the actual heat transfer rate. Error modeling will take the uncertainties and errors of the experiments into account, and present it in the form of a mathematical

correlation. With the correlation, necessary adjustments could be made to the heat transfer with respect to the error it has incurred. With that, the correlation formed will present even lesser uncertainties.

In addition to that, other heat exchanger configurations such as the shell and tube exchanger can be considered and the GA is applied to find or estimate the coefficients and exponents of the Nu correlations. This is to enhance the repertoire of GA in this field of optimization.

Meanwhile, it is found that MNCGA does not differ much from the basic GA, and the basic GA is sufficient to perform the required task. Though not all data points could be improved via GA, it still presents a good opportunity for GA to form a good correlation that could improve a majority of the data points.

CHAPTER 7
REFERENCES

1. Manish, T.C. , Yan Fu. 1999. Optimal Design of Heat Exchangers: A Genetic Algorithm Framework, USA, *Industrial & engineering chemistry research, vol. 38, pp. 456-467*

2. Chambers, L. 2001. *The Practical Handbook of Genetic Algorithms Applications*, USA, Chapman & Hall/CRC

3. R. K. Shah. 1990. Assessment of modified Wilson plot technique for obtaining heat transfer design, *Proc. 9th Int. Heat Transfer Conf. , vol. 5, pp. 51-56*

4. Incropera, P. , DeWitt, D. , 2004. *Fundamentals of Heat Transfer*, New York, Wiley Publishing

5. Arunasalam, P., Seetharamu, K.N. and Ishak, A.A. (2005) Determination of Thermal Compact Model via Evolutionary Genetic Optimization Method, Malaysia, *IEEE Transaction on Components and Packaging Technologies, Vol 28, No. 2*

6. Haupt, R.L. , Haupt, S.E. 1998. *Practical Genetic Algorithms*, USA, John Wiley & Sons

7. Cedeno, W., Vemuri, V.R. 1999. Analysis of Speciation and Niching in the Multi-Niche Crowding GA, *Theoretical Computer Science Archives, Volume. 229, Issue 1-2, pp. 177-197*

8. Jiang Feng, G. , Lin, C., Ming Tian, X. 2009. Optimization design of shell-and-tube heat exchanger by entropy generation minimization and genetic algorithm, China, *Applied thermal engineering, Vol 29, pp. 2954-2960*

9. Jin, C. 2007. Hybrid Genetic Algorithms for Structural Reliability Analysis, China, *Computers and Structures, Vol 85, pp. 1524-1533*

APPENDICES

Microsoft Visual Basic 2008 Code

Normal GA

```
Module Module1
    Public my_ga_parameters As New GA_parameters
    Public matrix_random(,) As Double
    Public my_chromosome() As chromosome
    Public crossover_chromosome() As chromosome
    Public holder_chromomsome() As chromosome
    Public timespan As Integer
    Public list(my_ga_parameters.no_of_populations - 1) As Double
    Public sort_list(my_ga_parameters.no_of_populations - 1) As Double
    Public sorted_list(my_ga_parameters.no_of_populations - 1) As Double
    Public sorted_index(my_ga_parameters.no_of_populations - 1) As Double
    Public list2(my_ga_parameters.no_of_populations - 1) As Double
    Public sort_list2(my_ga_parameters.no_of_populations - 1) As Double
    Public sorted_list2(my_ga_parameters.no_of_populations - 1) As Double
    Public sorted_index2(my_ga_parameters.no_of_populations - 1) As Double
    Public number_list1() As Integer
    Public number_list2() As Integer
    Dim Hi(my_ga_parameters.no_of_parameters - 1) As Double
    Dim Lo(my_ga_parameters.no_of_parameters - 1) As Double
    Public X() As Double
    Public Y() As Double
    Private RndSeed As Long = 1
    Public hi_sd As Integer ' sn = significant digit
    Public ho_sd As Integer
    Public experiment_counter As Integer
    Public myheatx_param As New heatx_params
    Public fitness_level As Double
    Public crossover_numbers As Integer
    Public allowed_difference As Double = 0
    Public error_holder() As Double

    Public Function random() As Double
        Dim probabilityTimeSeeded As New System.Random()
        RndSeed = RndSeed + probabilityTimeSeeded.Next
        Dim maxInt As Integer = Integer.MaxValue
        RndSeed = RndSeed * 2 + 17
        If RndSeed > Integer.MaxValue Then
            RndSeed = RndSeed Mod Integer.MaxValue
        End If
        Dim probabilitySeeded As New System.Random(RndSeed)
        Return probabilitySeeded.NextDouble
    End Function
    Sub Main()
        Dim exit_status As Integer
        Dim crossover_numbers As Integer
        Dim starttime As DateTime = DateTime.Now
```

```
        Dim holding_count As Integer = 0
        For iii As Integer = 0 To 9
            ReDim Preserve holder_chromomsome(iii)
            holder_chromomsome(iii) = New chromosome
            holder_chromomsome(iii).init(my_ga_parameters.no_of_parameters - 1)
        Next
        Dim elite_number As Integer = 0

        Do
            elite_number = 0
            preprocess()
            For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
                ReDim Preserve my_chromosome(i)
                my_chromosome(i) = New chromosome
                my_chromosome(i).init(my_ga_parameters.no_of_parameters - 1)
                For j As Integer = 0 To my_chromosome(i).gene.Length - 1
                    my_chromosome(i).gene(j) = matrix_random(i, j)
                Next
            Next

            For ii As Integer = 0 To 40

                For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
                    heat_transfer_eval(my_chromosome(i).gene(1), my_chromosome(i).gene(4), my_chromosome(i).gene(3),
my_chromosome(i).gene(0), my_chromosome(i).gene(2), my_chromosome(i).gene(5))
                    my_chromosome(i).fitness = fitness_eval()
                    For iii As Integer = 0 To UBound(error_holder)
                        error_holder(iii) = Math.Abs((myheatx_param.Qexp(iii) - myheatx_param.Qga(iii)) / myheatx_param.Qexp(iii)
* 100)
                    Next
                    If error_holder(get_max) < 3.4 And fitness_level < 1.08 And Math.Abs(my_chromosome(i).gene(1) -
my_chromosome(i).gene(4)) < 0.001 Then
                        MsgBox("done")
                        Exit Do
                    End If

                    'If fitness_level < 0.75 Then
                    '    MsgBox("done")
                    '    Exit Do
                    'End If
                Next
                If ii > 0 Then
                    elite_number = 1
                    elitist()
                End If
                crossover_numbers = get_crossover_numbers()
                If Not crossover_numbers = 0 Then
                    choose_mate(crossover_numbers)
                End If
```

```
            exit_status = RW_sorting()

            If exit_status = 1 Then
               MsgBox("done!")
               Exit Sub
            Else
               RW_config()
               selection(my_ga_parameters.no_of_populations - crossover_numbers)
               'crossover_numbers = 0
               If Not crossover_numbers = 0 Then
                  For iii As Integer = 0 To crossover_numbers - 1
                     ReDim Preserve crossover_chromosome(iii)
                     crossover_chromosome(iii) = New chromosome
                     crossover_chromosome(iii).init(my_ga_parameters.no_of_parameters - 1)
                  Next
                  crossover(my_chromosome)
               End If
               my_chromosome = create_new_generation(my_chromosome, crossover_chromosome, _
                                          (my_ga_parameters.no_of_populations - crossover_numbers), elite_number)
               mutation(random)
            End If
            'my_ga_parameters.mutation_rate = my_ga_parameters.mutation_rate * 0.6
            crossover_chromosome = Nothing
         Next

      Loop

      'Dim endtime As DateTime = DateTime.Now
      'Dim Span As TimeSpan = endtime - starttime
      'timespan = Span.TotalMilliseconds

   End Sub
   Function get_crossover_numbers()
      Dim cross_number As Integer
      Dim remainder As Long
      cross_number = random() * my_ga_parameters.no_of_populations
      Math.DivRem(cross_number, 2, remainder)
      If remainder = 1 Then
         cross_number = cross_number - 1
      End If

      Return cross_number
   End Function
   Function findmax(ByVal sent_chromosome() As chromosome)
      Dim counter As Integer = 0
      For i As Integer = 1 To UBound(sent_chromosome)
         If Not sent_chromosome(i).selected_status = -1 Then
            If sent_chromosome(counter).fitness < sent_chromosome(i).fitness Then
               counter = i
            End If
```

```
        End If
    Next
    Return counter
End Function

Function get_max()
    Dim counter As Integer = 0
    For i As Integer = 1 To UBound(error_holder)
        If error_holder(counter) < error_holder(i) Then
            counter = i
        End If
    Next
    Return counter
End Function
Function get_min()
    Dim counter As Integer = 0
    For i As Integer = 1 To UBound(error_holder)
        If error_holder(counter) > error_holder(i) Then
            counter = i
        End If
    Next
    Return counter
End Function
Sub selection(Optional ByVal survivor_numbers As Integer = 0)
    Dim taken As Integer = 0
    Dim rand As Double
    Dim status(my_ga_parameters.no_of_populations-1) As Double
    Do
        For i As Integer = 0 To UBound(my_chromosome)

            If i = 0 Then
                rand = random()
                If rand < my_ga_parameters.wheel(i) Then
                    If taken >= survivor_numbers Then
                        Exit Do
                    End If
                    my_chromosome(sorted_index(i)).survival_numbers = status(sorted_index(i)) + 1
                    status(sorted_index(i)) = status(sorted_index(i)) + 1
                    taken = taken + 1

                End If

            ElseIf Not i = 0 Then
                rand = random() 'Rnd()
                If rand > my_ga_parameters.wheel(i - 1) And rand < my_ga_parameters.wheel(i) Then
                    If taken >= survivor_numbers Then
                        Exit Do
                    End If
                    my_chromosome(sorted_index(i)).survival_numbers = status(sorted_index(i)) + 1
                    status(sorted_index(i)) = status(sorted_index(i)) + 1
```

```
                taken = taken + 1

            End If

        End If

    Next
  Loop Until taken = survivor_numbers

End Sub

Sub RW_config()
  Dim counter As Double = 0

  ReDim my_ga_parameters.wheel(UBound(my_chromosome))

  For i As Integer = 0 To UBound(my_chromosome)

    counter = counter + my_chromosome(sorted_index(i)).RW_value
    my_ga_parameters.wheel(i) = counter

  Next
End Sub

Sub elitist()
  For i As Integer = 0 To UBound(list)
    list2(i) = my_chromosome(i).fitness
    sorted_list2(i) = my_chromosome(i).fitness
  Next

  Array.Sort(sorted_list2)
  Array.Reverse(sorted_list2)

  For i As Integer = 0 To UBound(list2)
    For j As Integer = 0 To UBound(list2)
      If Not sorted_list2(i) = -1 And sorted_list2(i) = list2(j) Then
        sorted_index2(i) = j
        sorted_list2(i) = -1
      End If
    Next
  Next
End Sub
Function RW_sorting()
  Dim sum As Double = 0
  Dim rw_sum As Double = 0
  Dim index As Integer
  Dim counter As Integer = 0
  For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
    sum = sum + my_chromosome(i).fitness
  Next
```

```
If sum = 1 / 0 Then
   index = 1
   Return index
   Exit Function
End If

For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
   'rw_sum = rw_sum + my_chromosome(i).fitness / sum
   my_chromosome(i).RW_value = my_chromosome(i).fitness / sum
Next

For i As Integer = 0 To UBound(list)
   list(i) = my_chromosome(i).RW_value
   sorted_list(i) = my_chromosome(i).RW_value
Next

Array.Sort(sorted_list)
Array.Reverse(sorted_list)

Do
   For i As Integer = 0 To UBound(my_chromosome)
      If counter = my_ga_parameters.no_of_populations Then
         Exit Do
      End If
      If sorted_list(counter) = my_chromosome(i).RW_value And Not my_chromosome(i).selected_status = -1 Then
         my_chromosome(i).selected_status = -1
         sorted_index(counter) = i
         counter = counter + 1
      End If
   Next
Loop Until counter = my_ga_parameters.no_of_populations

'Dim myindex As Integer
'For i As Integer = 0 To UBound(my_chromosome)
'   myindex = findmax(my_chromosome)
'   my_chromosome(myindex).selected_status = -1
'   sorted_index(i) = myindex
'Next

'For i As Integer = 0 To UBound(list)
'   For j As Integer = 0 To UBound(list)
'      If i > 0 Then
'         If Not sorted_list(i) = -1 And sorted_list(i) = list(j) And Not j = sorted_index(i - 1) Then
'            sorted_index(i) = j
'            sorted_list(i) = -1
'         End If
'      Else
'         If Not sorted_list(i) = -1 And sorted_list(i) = list(j) Then
'            sorted_index(i) = j
```

```
'                sorted_list(i) = -1
'            End If
'        End If

'    Next
'Next
    Return index
End Function


Function create_new_generation(ByVal sent_chromosome() As chromosome, ByVal crossover_chrome() As chromosome, _
                    Optional ByVal survival_numbers As Integer = 0, Optional ByVal elitenum As Integer = 0)
    Dim new_chromosome() As chromosome

    If survival_numbers = -1 Then
        elitenum = 0
    End If
    Dim counter As Integer = elitenum
    For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
        ReDim Preserve new_chromosome(i)
        new_chromosome(i) = New chromosome
        new_chromosome(i).init(my_ga_parameters.no_of_parameters - 1)
    Next
    If Not elitenum = 0 And Not survival_numbers = -1 Then
        new_chromosome(0) = my_chromosome(sorted_index2(0)).clone
    End If
    If Not survival_numbers = -1 Then
        For i As Integer = 0 To my_ga_parameters.no_of_populations - 1
            If sent_chromosome(i).survival_numbers > 0 And Not counter = survival_numbers Then
                For j As Integer = 0 To sent_chromosome(i).survival_numbers - 1
                    If counter = survival_numbers Then
                        Exit For
                    Else
                        new_chromosome(counter) = sent_chromosome(i).clone
                        counter = counter + 1
                    End If
                Next
            Else
            End If
        Next
    End If

    If Not survival_numbers = my_ga_parameters.no_of_populations Then
        For i As Integer = 0 To UBound(crossover_chrome)
            new_chromosome(counter) = crossover_chrome(i).clone
            counter = counter + 1
            If counter >= my_ga_parameters.no_of_populations Then
                Exit For
            End If
        Next
```

```
    End If
    counter = 0
    Return new_chromosome
End Function
Sub choose_mate(Optional ByVal cross_over_numbers As Integer = 0)
    Dim selection As Integer = 0
    Dim take1 As Integer
    Dim take2 As Integer
    Dim random1 As Integer
    Dim random2 As Integer
    ' Dim number_list1() As Integer
    ' Dim number_list2() As Integer
    Dim count As Integer = 0
    'ReDim my_matingpool(my_parameters.solution_space - 1)

    Dim a As New List(Of Integer)

    For i As Integer = 0 To UBound(my_chromosome)
        a.Add(i)
    Next

    Do

        random1 = Int((a.Count - 1) * random())
        take1 = a.Item(random1)
        ReDim Preserve number_list1(count)
        number_list1(count) = take1
        a.Remove(take1)

        random2 = Int((a.Count - 1) * random())
        take2 = a.Item(random2)
        ReDim Preserve number_list2(count)
        number_list2(count) = take2
        a.Remove(take2)

        count = count + 1
        my_chromosome(take1).mating_partner = take2

    Loop Until a.Count = my_ga_parameters.no_of_populations - cross_over_numbers

End Sub

Sub crossover(ByVal sent_chromosome() As chromosome)
    Dim store1 As Double
    Dim store2 As Double
    Dim store3 As Double
    Dim store4 As Double
    Dim beta As Double
    Dim crossover_point As Double
    Dim counter As Integer = 0
```

```
For i As Integer = 0 To UBound(number_list1)
   beta = random()
   crossover_point = CInt(((my_ga_parameters.no_of_parameters - 1) * random()) + 1) - 1

   If crossover_point = 0 Then
      store1 = sent_chromosome(number_list1(i)).gene(crossover_point)
      store2 = sent_chromosome(number_list2(i)).gene(crossover_point)
      crossover_chromosome(counter).gene(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 3)
      crossover_chromosome(counter + 1).gene(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 3)
   ElseIf crossover_point = 3 Then
      store1 = sent_chromosome(number_list1(i)).gene(crossover_point)
      store2 = sent_chromosome(number_list2(i)).gene(crossover_point)
      crossover_chromosome(counter).gene(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 3)
      crossover_chromosome(counter + 1).gene(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 3)
   ElseIf crossover_point = 5 Or crossover_point = 2 Then
      store1 = sent_chromosome(number_list1(i)).gene(crossover_point)
      store2 = sent_chromosome(number_list2(i)).gene(crossover_point)
      crossover_chromosome(counter).gene(crossover_point) = (store1 - (beta * (store1 - store2)))
      crossover_chromosome(counter + 1).gene(crossover_point) = (store2 + (beta * (store1 - store2)))
   Else
      store1 = sent_chromosome(number_list1(i)).gene(crossover_point)
      store2 = sent_chromosome(number_list2(i)).gene(crossover_point)
      crossover_chromosome(counter).gene(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 3)
      crossover_chromosome(counter + 1).gene(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 3)
   End If


   If crossover_point = 0 Then
      For ii As Integer = 1 To my_ga_parameters.no_of_parameters - 1
         If ii = 3 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
         ElseIf ii = 5 Or ii = 2 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = store4
            crossover_chromosome(counter + 1).gene(ii) = store3
         Else
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
         End If

      Next
   ElseIf crossover_point = my_ga_parameters.no_of_parameters - 1 Then
      For ii As Integer = my_ga_parameters.no_of_parameters - 2 To 0 Step -1
```

```
        If ii = 3 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
        ElseIf ii = 5 Or ii = 2 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = store4
            crossover_chromosome(counter + 1).gene(ii) = store3
        Else
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
        End If
    Next
Else
    For ii As Integer = crossover_point + 1 To my_ga_parameters.no_of_parameters - 1
        If ii = 3 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
        ElseIf ii = 5 Or ii = 2 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = store4
            crossover_chromosome(counter + 1).gene(ii) = store3
        Else
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store4, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store3, 3)
        End If
    Next
End If
If Not crossover_point = 0 Or crossover_point = my_ga_parameters.no_of_parameters - 1 Then
    For ii As Integer = crossover_point - 1 To 0 Step -1
        If ii = 3 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = Math.Round(store3, 3)
            crossover_chromosome(counter + 1).gene(ii) = Math.Round(store4, 3)
        ElseIf ii = 5 Or ii = 2 Then
            store3 = sent_chromosome(number_list1(i)).gene(ii)
            store4 = sent_chromosome(number_list2(i)).gene(ii)
            crossover_chromosome(counter).gene(ii) = store4
            crossover_chromosome(counter + 1).gene(ii) = store3
        Else
```

```
                store3 = sent_chromosome(number_list1(i)).gene(ii)
                store4 = sent_chromosome(number_list2(i)).gene(ii)
                crossover_chromosome(counter).gene(ii) = Math.Round(store3, 3)
                crossover_chromosome(counter + 1).gene(ii) = Math.Round(store4, 3)
            End If
        Next
    End If
    counter = counter + 2
  Next

End Sub

Sub mutation(ByVal mutation_probability As Double)
  Dim counter As Integer
  Dim take1 As Integer
  Dim mutation_point As Integer
  Dim random1 As Integer
  Dim mutated_gene As Double
  Dim beta As Double
  If my_ga_parameters.mutation_rate > mutation_probability Then

    Dim a As New List(Of Integer)

    For i As Integer = 0 To UBound(my_chromosome)
      a.Add(i)
    Next

    Do
      beta = random()
      mutation_point = CInt(((my_ga_parameters.no_of_parameters - 1) * random()) + 1) - 1
      random1 = Int((a.Count - 1) * Rnd())
      take1 = a.Item(random1)
      mutated_gene = ((Hi(mutation_point) - Lo(mutation_point)) * beta) + Lo(mutation_point)
      If mutation_point = 0 Or mutation_point = 3 Then
        my_chromosome(take1).gene(mutation_point) = Math.Round(mutated_gene, 3)
      ElseIf mutation_point = 2 Or mutation_point = 5 Then
        my_chromosome(take1).gene(mutation_point) = mutated_gene
      Else
        my_chromosome(take1).gene(mutation_point) = Math.Round(mutated_gene, 3)
      End If

      counter = counter + 1

      a.Remove(random1)
    Loop Until counter = my_ga_parameters.mutation_numbers

  End If

End Sub
Sub preprocess()
```

```
    Hi(0) = 2
    Hi(1) = 0.6
    Hi(2) = 1 / 3
    Hi(3) = 2
    Hi(4) = 0.6
    Hi(5) = 1 / 3

    Lo(0) = 0.5
    Lo(1) = 0.5
    Lo(2) = 1 / 3
    Lo(3) = 0.6
    Lo(4) = 0.5
    Lo(5) = 1 / 3

    ReDim matrix_random(my_ga_parameters.no_of_populations - 1, my_ga_parameters.no_of_parameters - 1)

    For i As Integer = 0 To UBound(matrix_random, 1)
        For j As Integer = 0 To UBound(matrix_random, 2)
            If j = 0 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * random() + Lo(j)), 3)
            ElseIf j = 1 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * random() + Lo(j)), 3)
            ElseIf j = 2 Then
                matrix_random(i, j) = ((Hi(j) - Lo(j)) * random() + Lo(j))
            ElseIf j = 3 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * random() + Lo(j)), 3)
            ElseIf j = 4 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * random() + Lo(j)), 3)
            ElseIf j = 5 Then
                matrix_random(i, j) = ((Hi(j) - Lo(j)) * random() + Lo(j))
            End If
        Next
    Next

End Sub
Function significant_digit_counter(ByVal sent_digit As Double)
    Dim counter As Integer
    Dim number As String = sent_digit
    Dim x As Char
    Dim sd As Integer

    Do
        'If Not number(counter) = "." Then
        x = number(counter)
        counter = counter + 1
        'End If

    Loop Until x = "."
```

```
    counter = counter - 1

    If counter = 2 Then
        sd = 2
    ElseIf counter = 3 Then
        sd = 1
    Else
        sd = 0
    End If
    Return sd
End Function

Sub gather_data()
    Dim counter As Integer = 0
    Form1.WorkbookView2.GetLock()
    If Form1.WorkbookView2.ActiveWorksheet.Cells(0, 0).Value = 0 Then
        MsgBox("please key in values first")
    Else
        Do
            If Not Form1.WorkbookView2.ActiveWorksheet.Cells(counter, 0).Value = 0 Then
                counter = counter + 1
            Else
                Exit Do
            End If
        Loop
        experiment_counter = counter
        ReDim myheatx_param.ki(experiment_counter - 1)
        ReDim myheatx_param.ko(experiment_counter - 1)
        ReDim myheatx_param.Pri(experiment_counter - 1)
        ReDim myheatx_param.Pro(experiment_counter - 1)
        ReDim myheatx_param.Qexp(experiment_counter - 1)
        ReDim myheatx_param.Rei(experiment_counter - 1)
        ReDim myheatx_param.Reo(experiment_counter - 1)
        ReDim myheatx_param.Rw(experiment_counter - 1)
        ReDim myheatx_param.LMTD(experiment_counter - 1)
        ReDim myheatx_param.Qga(experiment_counter - 1)
        ReDim error_holder(experiment_counter - 1)

        For j As Integer = 0 To experiment_counter - 1
            myheatx_param.ki(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 0).Value
            myheatx_param.ko(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 1).Value
            myheatx_param.Pri(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 2).Value
            myheatx_param.Pro(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 3).Value
            myheatx_param.Qexp(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 4).Value
            myheatx_param.Rei(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 5).Value
            myheatx_param.Reo(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 6).Value
            myheatx_param.Rw(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 7).Value
            myheatx_param.LMTD(j) = Form1.WorkbookView2.ActiveWorksheet.Cells(j, 8).Value
        Next
        Form1.Button1.Enabled = True
```

```
        End If
        Form1.WorkbookView2.ReleaseLock()
    End Sub

    Function fitness_eval() '(ByVal hi As Double, ByVal ho As Double) As Double
        fitness_eval = 0
        fitness_level = 0
        'Dim temp_holder As Double
        For i As Integer = 0 To experiment_counter - 1
            'temp_holder = Math.Abs(allowed_difference - Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i)))
            fitness_eval = fitness_eval + (Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i)))
'Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i))
            fitness_level = fitness_level + (Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i)))
        Next
        'fitness_level = fitness_level / (experiment_counter)
        fitness_eval = 1 / fitness_eval

        Return fitness_eval
    End Function

    Sub heat_transfer_eval(ByVal Rei_coeff As Double, ByVal Reo_coeff As Double, ByVal cons_coeff_outer As Double, _
                           ByVal cons_coeff_inner As Double, ByVal Pri_coeff As Double, ByVal Pro_coeff As Double) ', ByVal
length As Double)
        Dim U() As Double
        ReDim U(experiment_counter - 1)
        Dim hi As Double
        Dim ho As Double
        'For i As Integer = 0 To UBound(U)
        '    hi = (cons_coeff_inner * (myheatx_param.Rei(i) ^ Rei_coeff) * (myheatx_param.Pri(i) ^ Pri_coeff) *
myheatx_param.ki(i) / 0.01792)
        '    ho = (cons_coeff_outer * (myheatx_param.Reo(i) ^ Reo_coeff) * (myheatx_param.Pro(i) ^ Pro_coeff) *
myheatx_param.ko(i) / 0.004)
        '    U(i) = (1 / (hi * 0.1678336)) + (1 / (ho * 0.208106171)) + myheatx_param.Rw(i)
        '    myheatx_param.Qga(i) = (1 / U(i)) * myheatx_param.LMTD(i)
        'Next
        For i As Integer = 0 To UBound(U)
            hi = (cons_coeff_inner * (myheatx_param.Rei(i) ^ Rei_coeff) * (myheatx_param.Pri(i) ^ Pri_coeff) *
myheatx_param.ki(i) / 0.00186) * (1 / 1000)
            ho = (cons_coeff_outer * (myheatx_param.Reo(i) ^ Reo_coeff) * (myheatx_param.Pro(i) ^ Pro_coeff) *
myheatx_param.ko(i) / 0.00186) * (1 / 1000)
            U(i) = (1 / (hi)) + (1 / (ho)) + myheatx_param.Rw(i)
            myheatx_param.Qga(i) = (1 / U(i)) * myheatx_param.LMTD(i) * 0.189
        Next
    End Sub

    Class GA_parameters
        Public no_of_parameters As Double = 6
        Public no_of_populations As Double = 150
        Public wheel() As Double
        Public mutation_numbers As Double = 16
```

```
    Public mutation_rate As Double = 0.15

  End Class

  Class chromosome
    Implements ICloneable
    Public gene() As Double
    Public fitness As Double
    Public raw_value As Double
    Public U_value As Double
    Public mating_partner As Double
    Public RW_value As Double 'roulette wheel value
    Public survival_numbers As Double
    Public ratio_difference As Double
    Public selected_status As Integer
    Sub New()

    End Sub
    Public Sub New(ByVal old_gene() As Double) ', ByVal old_survival_numbers As Double) ', ByVal old_fitness As
Double)
      gene = old_gene
    End Sub
    Sub init(ByVal number_of_parameters As Double)
      ReDim gene(number_of_parameters)
    End Sub

    Public Function clone() As Object Implements System.ICloneable.Clone
      Dim new_gene() As Double
      'Dim new_fitness As Double
      ReDim new_gene(my_ga_parameters.no_of_parameters - 1)
      'Dim new_survival_numbers As Double
      new_gene = Me.gene.Clone
      Dim instance As New chromosome(new_gene) ', new_survival_numbers) ', new_fitness)
      Return instance

    End Function
  End Class

  Class heatx_params
    Public Reo() As Double
    Public Pro() As Double
    Public ko() As Double
    Public Rw() As Double
    Public Rei() As Double
    Public Pri() As Double
    Public ki() As Double
    Public Qexp() As Double
    Public LMTD() As Double
    Public Qga() As Double
  End Class
```

```
End Module
```

MNCGA

```
Module Module1
    Public my_phenome() As phenome
    Public crowding_group(,) As phenome
    Public crowding_factor_group() As phenome
    Public crowd_phenome() As phenome
    Public my_parameters As New parameters
    Public my_ga_method As New GA_methods
    Public my_genepool() As genepool
    Dim Hi(my_parameters.bits_length) As Double
    Dim Lo(my_parameters.bits_length) As Double
    Public matrix_random(,) As Double
    Public experiment_counter As Integer
    Public myheatx_param As New heatx_params
    Public fitness_level As Double
    Private RndSeed As Long = 1
    Public error_holder() As Double
    Public Function Random() As Double
        Dim probabilityTimeSeeded As New System.Random()
        RndSeed = RndSeed + probabilityTimeSeeded.Next
        Dim maxInt As Integer = Integer.MaxValue
        RndSeed = RndSeed * 2 + 17
        If RndSeed > Integer.MaxValue Then
            RndSeed = RndSeed Mod Integer.MaxValue
        End If
        Dim probabilitySeeded As New System.Random(RndSeed)
        Return probabilitySeeded.NextDouble
    End Function

    Sub Main()
        'create_genepool()
        Do
            initiate()
            For i As Integer = 0 To UBound(my_phenome)
                For j As Integer = 0 To my_parameters.bits_length
                    If my_phenome(i).chromosome(j) = 0 Then
                        MsgBox("!")
                    End If
                Next
            Next
            For i As Integer = 0 To 80
                For ii As Integer = 0 To UBound(my_phenome)
                    my_phenome(ii).decoded_value = my_ga_method.heat_transfer_eval(my_phenome(ii).chromosome(1), _
                                                   my_phenome(ii).chromosome(4), _
                                                   my_phenome(ii).chromosome(3), _
                                                   my_phenome(ii).chromosome(0), _
```

```
                                                my_phenome(ii).chromosome(2), _
                                                my_phenome(ii).chromosome(5))
                my_phenome(ii).fitness = my_ga_method.fitness_eval
                For iii As Integer = 0 To UBound(error_holder)
                    error_holder(iii) = Math.Abs((myheatx_param.Qexp(iii) - myheatx_param.Qga(iii)) / myheatx_param.Qexp(iii)
* 100)
                Next
                If error_holder(get_max) < 2.82 And fitness_level < 1000 Then 'And Math.Abs(my_chromosome(i).gene(1) -
my_chromosome(i).gene(4)) < 0.001 Then
                    MsgBox("done")
                    Exit Do
                End If

                'If fitness_level < 1 Then
                '    MsgBox("done!")
                '    Exit Do
                'End If
            Next
            niching()
            For iii As Integer = 0 To UBound(my_phenome)
                For j As Integer = 0 To my_parameters.bits_length
                    If my_phenome(iii).chromosome(j) = 0 Then
                        MsgBox("!")
                    End If
                Next
            Next
            elitism()
        Next

        'find_values()

    Loop


End Sub

Sub elitism()

    Dim worst_index As Integer
    Dim best_index As Integer

    worst_index = worst_individual(my_phenome)
    best_index = best_individual(my_phenome)

    my_phenome(worst_index) = my_phenome(best_index).clone
    my_phenome(worst_index).tag_number = worst_index
    my_phenome(worst_index).decoded_value =
my_ga_method.heat_transfer_eval(my_phenome(worst_index).chromosome(1), my_phenome(worst_index).chromosome(4), _
                        my_phenome(worst_index).chromosome(3), my_phenome(worst_index).chromosome(0), _
                        my_phenome(worst_index).chromosome(2), my_phenome(worst_index).chromosome(5))
```

```
    End Sub
    Sub create_genepool()
        Dim quotient As Integer

        ReDim my_genepool((2 ^ my_parameters.bits_length) - 2)
        For i As Integer = 0 To UBound(my_genepool)
            my_genepool(i) = New genepool
            my_genepool(i).real_value = i + 1
            quotient = my_genepool(i).real_value
            For j As Integer = UBound(my_genepool(i).chroms) To 0 Step -1
                Math.DivRem(quotient, 2, my_genepool(i).chroms(j))
                If my_genepool(i).chroms(j) = 0 Then
                    quotient = (quotient / 2)
                ElseIf my_genepool(i).chroms(j) = 1 Then
                    quotient = (quotient / 2) - 0.5
                End If
                If quotient = 0 Then
                    GoTo breakpoint
                End If
            Next
breakpoint:
        Next

    End Sub

    Sub niching() 'includes both crowding and worst among most similar operators
        Dim index As Integer
        Dim worse_index As Integer
        Dim offspring As New phenome
        offspring.create(my_parameters.bits_length)
        For i As Integer = 0 To my_parameters.solution_space - 1
            Dim crowd_index(my_parameters.crowding_size - 1) As Integer
            For j As Integer = 0 To UBound(crowd_index)
                crowd_index(j) = Int(UBound(my_phenome) * Random())
            Next
            For ii As Integer = 0 To my_parameters.bits_length
                If my_phenome(i).chromosome(ii) = 0 Then
                    MsgBox("!")
                End If
            Next
            crowding(crowd_index, my_phenome)
            For ii As Integer = 0 To my_parameters.bits_length
                If my_phenome(i).chromosome(ii) = 0 Then
                    MsgBox("!")
                End If
            Next
            index = find_similar_mate_index(crowd_phenome, my_phenome(i))
            For ii As Integer = 0 To my_parameters.bits_length
```

```
            If my_phenome(i).chromosome(ii) = 0 Then
                MsgBox("!")
            End If
        Next
        offspring = do_crossover(my_phenome(i), my_phenome(crowd_phenome(index).tag_number))
        For ii As Integer = 0 To my_parameters.bits_length
            If my_phenome(i).chromosome(ii) = 0 Then
                MsgBox("!")
            End If
        Next
        worse_index = cluster_crowding_groups(my_phenome, offspring)
        For ii As Integer = 0 To my_parameters.bits_length
            If my_phenome(i).chromosome(ii) = 0 Then
                MsgBox("!")
            End If
        Next
        my_phenome(worse_index) = replacement(my_phenome(worse_index), offspring, worse_index)
        For ii As Integer = 0 To my_parameters.bits_length
            If my_phenome(i).chromosome(ii) = 0 Then
                MsgBox("!")
            End If
        Next
    Next

End Sub

Function replacement(ByVal worse_phenome As phenome, ByVal replacement_offspring As phenome, ByVal
worse_fit_index As Integer)

    worse_phenome = replacement_offspring.clone

    worse_phenome.tag_number = worse_fit_index

    worse_phenome.decoded_value = my_ga_method.heat_transfer_eval(worse_phenome.chromosome(1),
worse_phenome.chromosome(4), worse_phenome.chromosome(3), _
                        worse_phenome.chromosome(0), worse_phenome.chromosome(2), worse_phenome.chromosome(5))
    worse_phenome.fitness = my_ga_method.fitness_eval()

    Return worse_phenome
End Function

Public Function do_crossover(ByVal parent_phenome As phenome, ByVal parent_mate As phenome)
    Dim store1 As Double
    Dim store2 As Double
    Dim store3 As Double
    Dim store4 As Double
    Dim beta As Double
    Dim dice As Double = 0.5
    Dim crossover_point As Double
    Dim offspring As New phenome
```

```
offspring.create(UBound(parent_phenome.chromosome))
beta = Random()
crossover_point = CInt(((my_parameters.bits_length) * Random()) + 1) - 1

If crossover_point = 0 Then
   store1 = parent_phenome.chromosome(crossover_point)
   store2 = parent_mate.chromosome(crossover_point)
   If store1 = 0 Then
      MsgBox("!")
   End If
   If dice < Random() Then
      offspring.chromosome(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 5)
   Else
      offspring.chromosome(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 5)
   End If

ElseIf crossover_point = 3 Then
   store1 = parent_phenome.chromosome(crossover_point)
   store2 = parent_mate.chromosome(crossover_point)
   If dice < Random() Then
      offspring.chromosome(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 5)
   Else
      offspring.chromosome(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 5)
   End If
Else
   store1 = parent_phenome.chromosome(crossover_point)
   store2 = parent_mate.chromosome(crossover_point)
   If dice < Random() Then
      offspring.chromosome(crossover_point) = Math.Round((store1 - (beta * (store1 - store2))), 4)
   Else
      offspring.chromosome(crossover_point) = Math.Round((store2 + (beta * (store1 - store2))), 4)
   End If
End If

If crossover_point = 0 Then
   For ii As Integer = 1 To my_parameters.bits_length
      If ii = 3 Then
         store3 = parent_phenome.chromosome(ii)
         store4 = parent_mate.chromosome(ii)
         If dice < Random() Then
            offspring.chromosome(ii) = Math.Round(store4, 5)
         Else
            offspring.chromosome(ii) = Math.Round(store3, 5)
         End If
         If offspring.chromosome(ii) = 0 Then
            MsgBox("!")
         End If
      Else
         store3 = parent_phenome.chromosome(ii)
         store4 = parent_mate.chromosome(ii)
```

```
            If dice < Random() Then
               offspring.chromosome(ii) = Math.Round(store4, 4)
            Else
               offspring.chromosome(ii) = Math.Round(store3, 4)
            End If
            If offspring.chromosome(ii) = 0 Then
               MsgBox("!")
            End If
         End If
      Next

   ElseIf crossover_point = my_parameters.bits_length Then
      For ii As Integer = my_parameters.bits_length - 1 To 0 Step -1
         If ii = 3 Or ii = 0 Then
            store3 = parent_phenome.chromosome(ii)
            store4 = parent_mate.chromosome(ii)

            If dice < Random() Then
               offspring.chromosome(ii) = Math.Round(store4, 5)
               If offspring.chromosome(ii) = 0 Then
                  MsgBox("!")
               End If
            Else
               offspring.chromosome(ii) = Math.Round(store3, 5)
            End If
            If store3 = 0 Or store4 = 0 Then
               MsgBox("!")
            End If
         Else
            store3 = parent_phenome.chromosome(ii)
            store4 = parent_mate.chromosome(ii)
            If dice < Random() Then
               offspring.chromosome(ii) = Math.Round(store4, 4)
            Else
               offspring.chromosome(ii) = Math.Round(store3, 4)
            End If
            If offspring.chromosome(ii) = 0 Then
               MsgBox("!")
            End If
         End If
      Next
   Else
      For iii As Integer = 0 To crossover_point
         If iii = 0 Or iii = 3 Then
            store3 = parent_phenome.chromosome(iii)
            store4 = parent_mate.chromosome(iii)
            If dice < Random() Then
               offspring.chromosome(iii) = Math.Round(store4, 5)
            Else
               offspring.chromosome(iii) = Math.Round(store3, 5)
```

```
            End If
          Else
            store3 = parent_phenome.chromosome(iii)
            store4 = parent_mate.chromosome(iii)
            If dice < Random() Then
              offspring.chromosome(iii) = Math.Round(store4, 4)
            Else
              offspring.chromosome(iii) = Math.Round(store3, 4)
            End If
          End If
        Next
        For ii As Integer = crossover_point + 1 To my_parameters.bits_length
          If ii = 3 Then
            store3 = parent_phenome.chromosome(ii)
            store4 = parent_mate.chromosome(ii)
            If dice < Random() Then
              offspring.chromosome(ii) = Math.Round(store4, 5)
            Else
              offspring.chromosome(ii) = Math.Round(store3, 5)
            End If
          Else
            store3 = parent_phenome.chromosome(ii)
            store4 = parent_mate.chromosome(ii)
            If dice < Random() Then
              offspring.chromosome(ii) = Math.Round(store4, 4)
            Else
              offspring.chromosome(ii) = Math.Round(store3, 4)
            End If
            If offspring.chromosome(ii) = 0 Then
              MsgBox("!")
            End If
          End If
        Next
      End If

      offspring.decoded_value = my_ga_method.heat_transfer_eval(offspring.chromosome(1), offspring.chromosome(4), _
                                          offspring.chromosome(3), offspring.chromosome(0), _
                                          offspring.chromosome(2), offspring.chromosome(5))
      Return offspring
    End Function
    '   Public Function do_mutation(ByVal sent_phenome As phenome)
    '       Dim mutation_rate As Double
    '       Dim random_bit As Integer
    '       Dim mutation_count As Integer = 0

    '       mutation_rate = getRandomNumber()

    '       If mutation_rate > my_parameters.mutation_rate Then
    'return_point:
    '           random_bit = UBound(sent_phenome.chromosome)
```

```
'           If sent_phenome.chromosome(random_bit) = 0 Then
'              sent_phenome.chromosome(random_bit) = 1
'           ElseIf sent_phenome.chromosome(random_bit) = 1 Then
'              sent_phenome.chromosome(random_bit) = 0
'           End If
'           For i As Integer = 0 To UBound(sent_phenome.chromosome)
'              If sent_phenome.chromosome(i) = 0 Then
'                 mutation_count = mutation_count + 1
'              End If
'           Next
'           If mutation_count >= UBound(sent_phenome.chromosome) Then
'              GoTo return_point
'           End If
'       End If

'       sent_phenome.decoded_value = my_ga_method.find_real_number(sent_phenome) / 10

'       Return sent_phenome
'   End Function

Function cluster_crowding_groups(ByVal sent_phenome() As phenome, ByVal offspring_phenome As phenome)
    Dim single_crowding_group() As phenome
    Dim random_index As Integer
    Dim single_crowding_group_counter As Integer

    Dim worse_tag_number As Integer

    ReDim single_crowding_group(my_parameters.crowding_group_size - 1)
    ReDim crowding_group(my_parameters.crowding_groups - 1, my_parameters.crowding_group_size - 1)
    ReDim crowding_factor_group(my_parameters.crowding_groups - 1)

    For i As Integer = 0 To UBound(crowding_factor_group)
        crowding_factor_group(i) = New phenome
    Next

    For i As Integer = 0 To UBound(crowding_group, 1)
        crowding_factor_group(i) = New phenome
        single_crowding_group_counter = 0
        For j As Integer = 0 To UBound(crowding_group, 2)
            random_index = UBound(sent_phenome) * Random()
            crowding_group(i, j) = New phenome
            crowding_group(i, j) = sent_phenome(random_index).clone
            crowding_group(i, j).decoded_value = my_ga_method.heat_transfer_eval(crowding_group(i, j).chromosome(1), _
                                                 crowding_group(i, j).chromosome(4), _
                                                 crowding_group(i, j).chromosome(3), _
                                                 crowding_group(i, j).chromosome(0), _
                                                 crowding_group(i, j).chromosome(2), _
                                                 crowding_group(i, j).chromosome(5))
            '
            'the following code is done as we wish to send only 1D array to the function
```

```
            single_crowding_group(single_crowding_group_counter) = New phenome
            single_crowding_group(single_crowding_group_counter) = crowding_group(i, j).clone
            single_crowding_group(single_crowding_group_counter).decoded_value = my_ga_method.heat_transfer_eval( _
            single_crowding_group(single_crowding_group_counter).chromosome(1),
single_crowding_group(single_crowding_group_counter).chromosome(4), _
            single_crowding_group(single_crowding_group_counter).chromosome(3),
single_crowding_group(single_crowding_group_counter).chromosome(0), _
            single_crowding_group(single_crowding_group_counter).chromosome(2),
single_crowding_group(single_crowding_group_counter).chromosome(5))
            single_crowding_group_counter = single_crowding_group_counter + 1
            'end
        Next
        crowding_factor_group(i) = crowding_select_similar_individuals(single_crowding_group, offspring_phenome)
        crowding_factor_group(i).decoded_value =
my_ga_method.heat_transfer_eval(crowding_factor_group(i).chromosome(1), _
                                                crowding_factor_group(i).chromosome(4), _
                                                crowding_factor_group(i).chromosome(3), _
                                                crowding_factor_group(i).chromosome(0), _
                                                crowding_factor_group(i).chromosome(2), _
                                                crowding_factor_group(i).chromosome(5))
        crowding_factor_group(i).fitness = my_ga_method.fitness_eval()
    Next

    worse_tag_number = crowding_factor_group(worse_individual(crowding_factor_group)).tag_number

    Return worse_tag_number

  End Function
  Function worse_individual(ByVal sent_crowding_factor_group() As phenome)
    Dim worse_index As Integer = 0

    For i As Integer = 1 To UBound(sent_crowding_factor_group)
      If sent_crowding_factor_group(worse_index).fitness > sent_crowding_factor_group(i).fitness Then
        worse_index = i
      End If
    Next

    Return worse_index
  End Function
  Function worst_individual(ByVal sent_phenome() As phenome)
    Dim worst_index As Integer

    For i As Integer = 1 To UBound(sent_phenome)
      If sent_phenome(worst_index).fitness > sent_phenome(i).fitness Then
        worst_index = i
      End If
    Next

    Return worst_index
  End Function
```

```
    Function best_individual(ByVal sent_phenome() As phenome)
        Dim best_index As Integer

        For i As Integer = 1 To UBound(sent_phenome)
            If sent_phenome(best_index).fitness < sent_phenome(i).fitness Then
                best_index = i
            End If
        Next

        Return best_index
    End Function

    Function crowding_select_similar_individuals(ByVal sent_crowding_group() As phenome, _
     ByVal sent_offspring As phenome)
        Dim most_similar_phenome As New phenome
        Dim phenotypic_distance As Double
        Dim most_similar_individual As Integer = 0

        For i As Integer = 1 To UBound(sent_crowding_group)
            phenotypic_distance = Math.Abs(sent_offspring.decoded_value -
sent_crowding_group(most_similar_individual).decoded_value)

            If Math.Abs(sent_offspring.decoded_value - sent_crowding_group(i).decoded_value) < phenotypic_distance Then
                most_similar_individual = i
            End If
        Next
        most_similar_phenome = sent_crowding_group(most_similar_individual).clone

        Return most_similar_phenome
    End Function



    Sub crowding(ByVal sent_crowd_index() As Integer, ByVal sent_phenome() As phenome)

        For i As Integer = 0 To my_parameters.crowding_size - 1
            ReDim Preserve crowd_phenome(i)
            crowd_phenome(i) = New phenome
            crowd_phenome(i) = sent_phenome(sent_crowd_index(i)).clone
            crowd_phenome(i).decoded_value = my_ga_method.heat_transfer_eval(crowd_phenome(i).chromosome(1), _
            crowd_phenome(i).chromosome(4), crowd_phenome(i).chromosome(3), crowd_phenome(i).chromosome(0), _
            crowd_phenome(i).chromosome(2), crowd_phenome(i).chromosome(5))
            For j As Integer = 0 To my_parameters.bits_length
                If crowd_phenome(i).chromosome(j) = 0 Then
                    MsgBox("!")
                End If
            Next
        Next

    End Sub
    Function find_similar_mate_index(ByVal crowd_group() As phenome, ByVal parent As phenome)
```

```
    Dim phenotypic_distance As Double
    phenotypic_distance = 0
    Dim most_similar_index As Integer = 0

    For i As Integer = 1 To UBound(crowd_group)

        phenotypic_distance = Math.Abs(parent.decoded_value - crowd_group(most_similar_index).decoded_value)

        If Math.Abs(parent.decoded_value - crowd_group(i).decoded_value) < phenotypic_distance Then
            most_similar_index = i
        End If
    Next

    Return most_similar_index
End Function
Sub initiate()

    For i As Integer = 0 To my_parameters.solution_space - 1
        ReDim Preserve my_phenome(i)
        my_phenome(i) = New phenome
        my_phenome(i).create(my_parameters.bits_length, i)
    Next

    preprocess()
    find_values()
    'the following code is used if removal of array element is needed
    "2. order Is maintained
    'Dim j As Long, N As Long = 25
    "assumes N is the position In MyArray to remove
    'For j = N To UBound(my_phenome) - 1
    '    my_phenome(j) = my_phenome(j + 1)
    'Next j
    'ReDim Preserve my_phenome(UBound(my_phenome) - 1)

End Sub
'assign random numbers to chromosome
Sub preprocess()
    Hi(0) = 0.05
    Hi(1) = 0.85
    Hi(2) = 1 / 3
    Hi(3) = 0.03
    Hi(4) = 0.85
    Hi(5) = 1 / 3

    Lo(0) = 0.046
    Lo(1) = 0.71
    Lo(2) = 1 / 3
    Lo(3) = 0.02
    Lo(4) = 0.71
    Lo(5) = 1 / 3
```

```
    ReDim matrix_random(my_parameters.solution_space - 1, my_parameters.bits_length)

    For i As Integer = 0 To UBound(matrix_random, 1)
        For j As Integer = 0 To UBound(matrix_random, 2)
            If j = 0 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * Random() + Lo(j)), 5)
            ElseIf j = 1 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * Random() + Lo(j)), 4)
            ElseIf j = 2 Then
                matrix_random(i, j) = ((Hi(j) - Lo(j)) * Random() + Lo(j))
            ElseIf j = 3 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * Random() + Lo(j)), 5)
            ElseIf j = 4 Then
                matrix_random(i, j) = Math.Round(((Hi(j) - Lo(j)) * Random() + Lo(j)), 4)
            ElseIf j = 5 Then
                matrix_random(i, j) = ((Hi(j) - Lo(j)) * Random() + Lo(j))
            End If
        Next
    Next

    For i As Integer = 0 To my_parameters.solution_space - 1
        For j As Integer = 0 To my_parameters.bits_length
            my_phenome(i).chromosome(j) = matrix_random(i, j)
        Next
    Next
End Sub
'Sub take_chromo(ByVal index As Integer, ByVal sent_phenome As phenome)
'    sent_phenome.chromosome = my_genepool(index).chroms
'    my_genepool(index).taken_status = my_genepool(index).taken_status + 1
'End Sub


Sub find_values()

    For i As Integer = 0 To UBound(my_phenome)
        my_phenome(i).decoded_value = my_ga_method.heat_transfer_eval(my_phenome(i).chromosome(1), _
                                                my_phenome(i).chromosome(4), _
                                                my_phenome(i).chromosome(3), _
                                                my_phenome(i).chromosome(0), _
                                                my_phenome(i).chromosome(2), _
                                                my_phenome(i).chromosome(5))
        my_phenome(i).fitness = my_ga_method.fitness_eval()
    Next

End Sub

Sub gather_data()
    Dim counter As Integer = 0
    Form1.WorkbookView1.GetLock()
```

```
    If Form1.WorkbookView1.ActiveWorksheet.Cells(0, 0).Value = 0 Then
        MsgBox("please key in values first")
    Else
        Do
            If Not Form1.WorkbookView1.ActiveWorksheet.Cells(counter, 0).Value = 0 Then
                counter = counter + 1
            Else
                Exit Do
            End If
        Loop
        experiment_counter = counter
        ReDim myheatx_param.ki(experiment_counter - 1)
        ReDim myheatx_param.ko(experiment_counter - 1)
        ReDim myheatx_param.Pri(experiment_counter - 1)
        ReDim myheatx_param.Pro(experiment_counter - 1)
        ReDim myheatx_param.Qexp(experiment_counter - 1)
        ReDim myheatx_param.Rei(experiment_counter - 1)
        ReDim myheatx_param.Reo(experiment_counter - 1)
        ReDim myheatx_param.Rw(experiment_counter - 1)
        ReDim myheatx_param.LMTD(experiment_counter - 1)
        ReDim myheatx_param.Qga(experiment_counter - 1)
        ReDim error_holder(experiment_counter - 1)

        For j As Integer = 0 To experiment_counter - 1
            myheatx_param.ki(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 0).Value
            myheatx_param.ko(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 1).Value
            myheatx_param.Pri(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 2).Value
            myheatx_param.Pro(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 3).Value
            myheatx_param.Qexp(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 4).Value
            myheatx_param.Rei(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 5).Value
            myheatx_param.Reo(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 6).Value
            myheatx_param.Rw(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 7).Value
            myheatx_param.LMTD(j) = Form1.WorkbookView1.ActiveWorksheet.Cells(j, 8).Value
        Next
        Form1.Button2.Enabled = True
    End If
    Form1.WorkbookView1.ReleaseLock()
End Sub
Class phenome
    Implements ICloneable
    Public chromosome() As Double
    Public decoded_value As Double
    Public fitness As Double
    Public tag_number As Integer

    Sub New()

    End Sub
    Sub create(ByVal bits_length As Integer, Optional ByVal index As Integer = 0)
        ReDim chromosome(bits_length)
```

```
        tag_number = index
    End Sub

    Public Sub New(ByVal old_chrome() As Double, ByVal old_tag As Integer)
        chromosome = old_chrome
        tag_number = old_tag
    End Sub

    Public Function clone() As Object Implements System.ICloneable.Clone
        Dim clone_chromosome() As Double
        Dim clone_tag As Integer

        ReDim clone_chromosome(Me.chromosome.Length - 1)
        clone_chromosome = Me.chromosome.Clone

        clone_tag = Me.tag_number

        Dim instance As New phenome(clone_chromosome, clone_tag)

        Return instance
    End Function
End Class

Class GA_methods

    Public Function find_real_number(ByVal myphenome As phenome)

        Dim real_number As Double = 0
        Dim inc As Integer = 0

        For i As Integer = UBound(myphenome.chromosome) To 0 Step -1
            real_number = real_number + myphenome.chromosome(inc) * (2 ^ i)
            inc = inc + 1
        Next

        Return real_number

    End Function

    Public Function objective_function(ByVal parameter As Double)
        Dim objective As Double

        objective = 800 - 62.83 * ((2 * parameter) + (0.91 * parameter ^ -0.2))

        Return objective

    End Function

    Public Function fitness_eval()
        fitness_eval = 0
```

```
        fitness_level = 0
        For i As Integer = 0 To experiment_counter - 1
            fitness_eval = fitness_eval + Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i))
            fitness_level = fitness_level + Math.Abs(myheatx_param.Qexp(i) - myheatx_param.Qga(i))
        Next

        fitness_eval = 1 / fitness_eval
    End Function

    Public Function heat_transfer_eval(ByVal Rei_coeff As Double, ByVal Reo_coeff As Double, ByVal cons_coeff_outer As Double, _
                    ByVal cons_coeff_inner As Double, ByVal Pri_coeff As Double, ByVal Pro_coeff As Double)
        Dim totalQ As Double = 0
        Dim U() As Double
        ReDim U(experiment_counter - 1)
        Dim hi As Double
        Dim ho As Double
        For i As Integer = 0 To UBound(U)
            hi = (cons_coeff_inner * (myheatx_param.Rei(i) ^ Rei_coeff) * (myheatx_param.Pri(i) ^ Pri_coeff) * myheatx_param.ki(i) / 0.01792)
            ho = (cons_coeff_outer * (myheatx_param.Reo(i) ^ Reo_coeff) * (myheatx_param.Pro(i) ^ Pro_coeff) * myheatx_param.ko(i) / 0.004)
            U(i) = (1 / (hi * 0.1678336)) + (1 / (ho * 0.208106171)) + myheatx_param.Rw(i)
            myheatx_param.Qga(i) = (1 / U(i)) * myheatx_param.LMTD(i)
        Next

        For i As Integer = 0 To UBound(myheatx_param.Qga)
            totalQ = totalQ + myheatx_param.Qga(i)
        Next

        Return totalQ

    End Function

End Class
Function get_max()
    Dim counter As Integer = 0
    For i As Integer = 1 To UBound(error_holder)
        If error_holder(counter) < error_holder(i) Then
            counter = i
        End If
    Next
    Return counter
End Function
Class parameters
    Public no_of_generations As Integer = 10
    Public bits_length As Integer = 5
    Public solution_space As Integer = 150
    Public crowding_size As Integer = 10
    Public crowding_groups As Integer = 8 'number of crowding groups
```

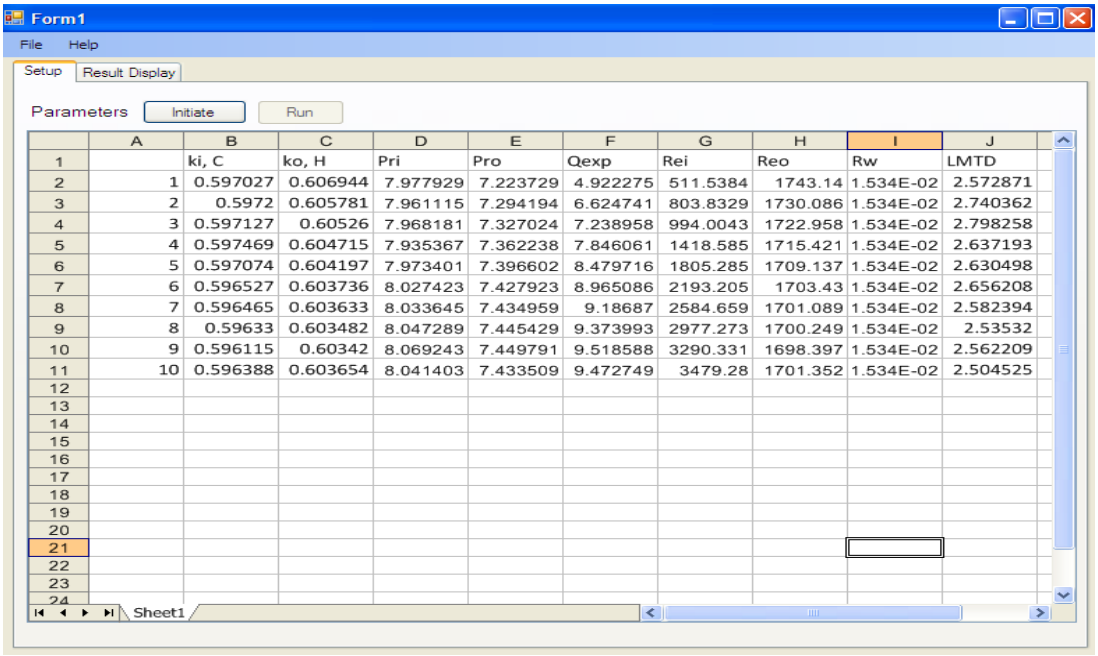
```
        Public crowding_group_size As Integer = 8 'number of individuals per crowding group
        Public mutation_rate As Double = 0.0005

    End Class

    Class genepool
        Public chroms(my_parameters.bits_length - 1) As Integer
        Public real_value As Integer
        Public taken_status As Integer
    End Class

    Class heatx_params
        Public Reo() As Double
        Public Pro() As Double
        Public ko() As Double
        Public Rw() As Double
        Public Rei() As Double
        Public Pri() As Double
        Public ki() As Double
        Public Qexp() As Double
        Public LMTD() As Double
        Public Qga() As Double
    End Class
End Module
```

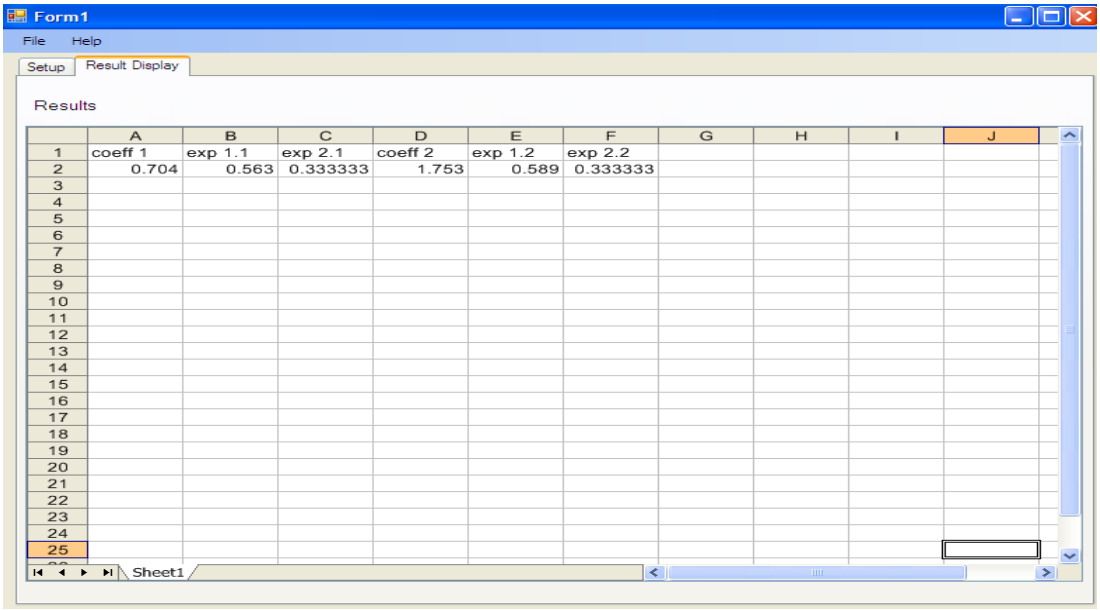
Software Demonstration (Screenshots)



	A	B	C	D	E	F	G	H	I	J
1		ki, C	ko, H	Pri	Pro	Qexp	Rei	Reo	Rw	LMTD
2	1	0.597027	0.606944	7.977929	7.223729	4.922275	511.5384	1743.14	1.534E-02	2.572871
3	2	0.5972	0.605781	7.961115	7.294194	6.624741	803.8329	1730.086	1.534E-02	2.740362
4	3	0.597127	0.60526	7.968181	7.327024	7.238958	994.0043	1722.958	1.534E-02	2.798258
5	4	0.597469	0.604715	7.935367	7.362238	7.846061	1418.585	1715.421	1.534E-02	2.637193
6	5	0.597074	0.604197	7.973401	7.396602	8.479716	1805.285	1709.137	1.534E-02	2.630498
7	6	0.596527	0.603736	8.027423	7.427923	8.965086	2193.205	1703.43	1.534E-02	2.656208
8	7	0.596465	0.603633	8.033645	7.434959	9.18687	2584.659	1701.089	1.534E-02	2.582394
9	8	0.59633	0.603482	8.047289	7.445429	9.373993	2977.273	1700.249	1.534E-02	2.53532
10	9	0.596115	0.60342	8.069243	7.449791	9.518588	3290.331	1698.397	1.534E-02	2.562209
11	10	0.596388	0.603654	8.041403	7.433509	9.472749	3479.28	1701.352	1.534E-02	2.504525

- All the necessary inputs required by GA are inserted in this interface.

- This values will provide the necessary information to obtain the heat transfer rate, based on the $Q=UA\Delta T_{LMTD}$ relationship.



	A	B	C	D	E	F	G	H	I	J
1	coeff 1	exp 1.1	exp 2.1	coeff 2	exp 1.2	exp 2.2				
2	0.704	0.563	0.333333	1.753	0.589	0.333333				

- The “RUN” button is clicked and the results will be shown if the termination criterion is fulfilled.

- In this case, as an example, from the figure above, the Nusselt correlations are $Nu_{in}=0.704Re^{0.563}Pr^{0.333333}$, $Nu_{out}=1.753Re^{0.589}Pr^{0.333333}$