University of Strathclyde Engineering

Department of Mechanical Engineering

Development of an Optimisation Algorithm for Auto-sizing Capacity of Renewable and Low Carbon Energy Systems

Author: Sana Waqas

Supervisor: Dr Paul Strachan

A thesis submitted in partial fulfilment for the requirement of the degree

Master of Science

Sustainable Energy: Renewable Energy Systems and the Environment

2011

Copyright Declaration

To My Parents……

Abstract

Hybrid renewable energy systems can help in improving the economics and environmental sustainability of renewable/low carbon energy systems. For better utilization of renewable/low carbon energy systems, design of hybrid systems in terms of correct size and selection is very important. Finding optimum capacity of each component of hybrid systems in large scale problems will be time taking associated with exhaustive search and it will requires complex computer program/package. The main goal of this study is to research and develop optimization algorithm for auto sizing capacity of renewable/low carbon energy systems to assist in development of hybrid energy systems. The developed algorithm is mainly designed for implementation into MERIT, an energy systems evaluation simulation computer package that supports the analysis of new and renewable energy schemes. Merit lacks this capability, thus this developed algorithm will work as Merit improvement.

Existing literature about auto sizing capacity of renewable/low carbon energy, their match evaluation criteria's and optimization algorithms was reviewed. Genetic algorithms and inequality along with correlation coefficients of supply/demand profile have proved to be most suitable algorithm and main objective function respectively for the particular problem. The particular auto sizing problem is then formulated mathematically and designed in C++ which is integrated development environment of Merit. Particular design of algorithm deals with single objective optimization using genetic algorithm and inequality coefficient as main objective to find several results from which one is selected with maximum correlation coefficient as best optimized solution. Analysis of particular nature of genetic algorithm, optimization methods, and renewable energy match criterias provide the reasons behind the particular design. The developed algorithm is verified by global optimization toolbox of Matlab software.

An actual case study is done with help of Merit by using actual hourly supplies and demand data based on climatic conditions. Results show that developed algorithm can handle large scale problems, works in a reasonable computation time, good in eliminating exhaustive search and can be successfully embedded with Merit. Further work areas are also discussed at end.

Acknowledgements:

First of all, I would like to thank Dr. Paul Strachan and Dr. Jun Hong for spending their time on my project and helping me throughout to reach towards optimized contents of my research work. I really have no words to describe how Dr. Jun Hong assisted me to polish my knowledge and concepts during the course of project and I really enjoyed progressing in an innovative way in the light of his ideas. Meanwhile, supervision of Dr. Paul Strachan has enabled me to learn and analyze about MERIT software during course work and to complete my work on time during thesis research work of my MSc.

I am also thankful to my class mates in MSc who made my time very beautiful in Scotland. The experience I gained by interacting with students from different nationalities also improved my vision.

I want to thank with utmost gratitude Mr. Waqas Khan, my husband, who managed to provide me finance for my MSc, encouragement when I needed, support in domestic work, and above all this, love and kindness. I will never forget his help to reinforce my determination throughout the year of study and he deserves more than words.

At the end, I want to dedicate my thesis to my parents who have made me what I am today by sacrificing their time and money and my each and every success belongs to their initial guidance. Above all, I am thankful to Allah who has given me these loving people and beautiful opportunities in my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction:

1.1. Introduction:

The environmental effects related to energy production have attained global attention since 1970 when the world major industries faced the shortage of Petroleum and worst energy crises. Along with the scarcity problems associated with fossil fuels, they are also related to severe environmental problems which include mainly green house gas emissions and global warming. The world started looking for the ways to get energy from the sources which will be emissions free as well as sustainable i.e. will also be available for the future needs so renewable energy resources started gaining attraction. Researches and development were started in order to achieve their best utilization to find the most efficient conversion technologies to convert them into the form which can be used for useful purposes. The renewable energy resources are present naturally, have huge potential to achieve the goals like diversity in energy production technologies, movement towards clean energy resources, balancing the use of fossil fuel, saving them for other applications in future, and reducing emissions.

Renewable energy sources are highly unpredictable, variable with time and dependant on climatic conditions. Renewable energy system is competitive and feasible for off-grid application; single source renewable usually leads to component over sizing, which increases the operating and life cycle costs (Bagul et al., 1996). Single renewable energy supply do not provide same amount of energy all the time so concept of combined utilization of more than one supply is presented which is known as hybrid energy systems concept. Hybrid energy system concept is becoming popular because of their high efficiency, high load factor, reliability, low emission and acceptable maintenance cost when compared with individual supply options. Design of renewable energy hybrid systems requires correct selection and size of different options available based on appropriate strategy and here comes main background and justification for thesis work which provides the motivation for this report.

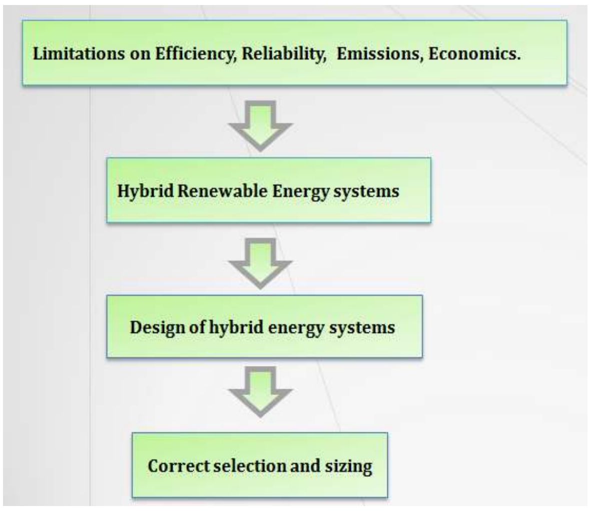

Limitations on Efficiency, Reliability, Emissions, Economics.
Hybrid Renewable Energy systems
Design of hybrid energy systems
Correct selection and sizing

Figure 1: Background for Project Work

Figure 2: Hybrid system (wind/solar)

There are challenges related to find correct capacity for hybrid energy systems. Considering two or three options of supplies to be matched with a single demand can lead towards a simple hit and trial method to find optimum size or capacity of each supply but dealing with finding optimum capacity of 1000's of supplies to satisfy a demand would be difficult and exhaustive with manual search keeping in mind variable nature of renewable energy. For this purpose complex computer programs and optimization algorithms are required to find optimum solution in a reasonable time.

Extensive research has been done in the field of optimizing renewable energy supplies with different objectives for example minimizing cost, maximizing match or reliability. Initial optimization and component sizing methods are based on worst month scenario leads to non-optimal design with excess capacity (Celik, 2003). This report represents a new design of optimization process for finding optimum capacity of renewable/low carbon energy supplies with objective of maximizing electricity match rate between demand and supplies by utilizing concept of genetic algorithms. Genetic algorithms are search heuristic that mimics the process of natural evolution. Genetic Algorithms are selected after reviewing extensive literatures related to optimization algorithms, current researches done in field of optimization of renewable energy, advantages of genetic algorithm and their good match with nature of renewable/low carbon energy. The designed optimization process is made mainly to implement and use in Merit tool. Merit is a computer program developed by

University of Strathclyde to support the development of new and renewable energy schemes and is a quantitative evaluation tool that allows user to determine match between supply and demand in order to make right decisions.

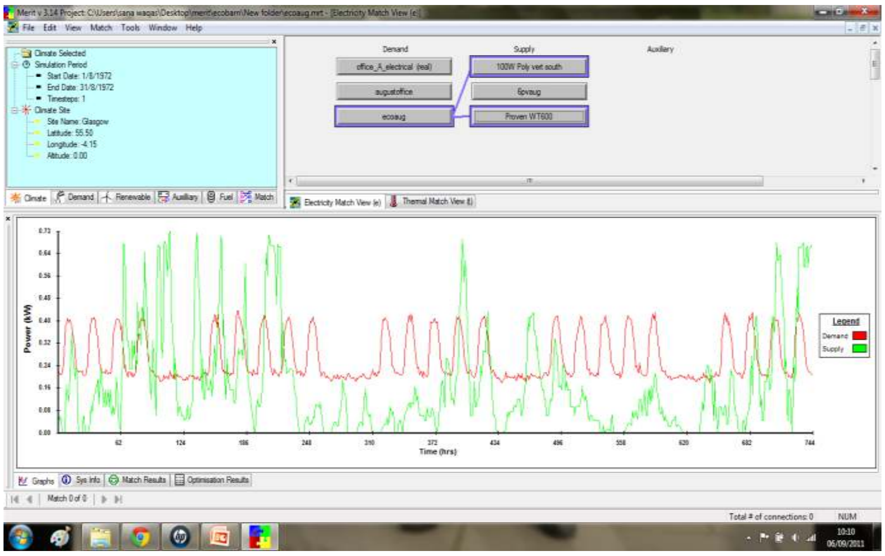

Climate Selected
Simulation Period
Start Date: 1/8/1972
End Date: 31/8/1972
Timesteps: 1
Climate Site
Site Name: Glasgow
Latitude: 55.50
Longitude: -4.15
Altitude: 0.00

Figure 3: Merit Tool environment

Merit tool has already option available for auto searching optimum capacity of a single renewable energy supply to match with a single supply but if more than one supply options are to be matched with a single demand then Merit cannot provide optimum capacity of each supply to be best suited with demand. This thesis is showing work for auto sizing capacity of number of supplies available to match with single demand to overhaul Merit capabilities and hence will prove to be a further improvement to Merit tool.

1.2. <u>Objectives:</u>

The main purpose of this thesis is to research and generate an optimization algorithm for auto sizing capacity of different renewable/low carbon energy supplies to assist creation of new option available for Merit tool which would prove to be useful in analysis of hybrid energy systems.

Particular objectives of thesis include

- To research best suited optimization algorithm which match finely with nature of renewable/ low carbon energy systems.

- To analyse different match evaluation criterias used for renewable/low carbon energy systems for finding best suited objective function for development of algorithm.
- To develop a method of optimization best suited for particular auto sizing problem utilizing selected optimization algorithm and objective functions.
- To develop a C++ programming code using Microsoft visual studio which is integrated environment of Merit tool for selected optimization algorithm with help of appropriate mathematical model.
- To verify, analyse and discuss results obtained from developed optimization method.

1.3. Methodology:

Theses work methodology is provided in figure 4. Report will first present literature review about match evaluation criteria's, optimization algorithms and current researches on particular auto sizing problem. From literature review, best suitable match evaluation criteria and optimization algorithms are selected for use in design of optimization algorithm. Design of optimization algorithm also involves selection of design variables, variable bounds and mathematical formulation of particular auto sizing problem. Then programming codes are produced for Microsoft visual studio C++ and Matlab designed optimization process. C++ with a special library of genetic algorithm named as GAlib is used for designing optimization algorithm because integrated environment of Merit tool is C++ and Matlab is used for verification of designed algorithm. After designing, a detailed analysis is done on the basis of different objective functions, different combinations of objective functions, genetic parameters, genetic nature and mathematical function behaviour. A case study is done by using actual profiles of hourly supply and demand data which are exported from Merit tool to conclude about the behaviour of designed algorithm when it will be implemented in Merit tool. At the end conclusions and further work is presented.

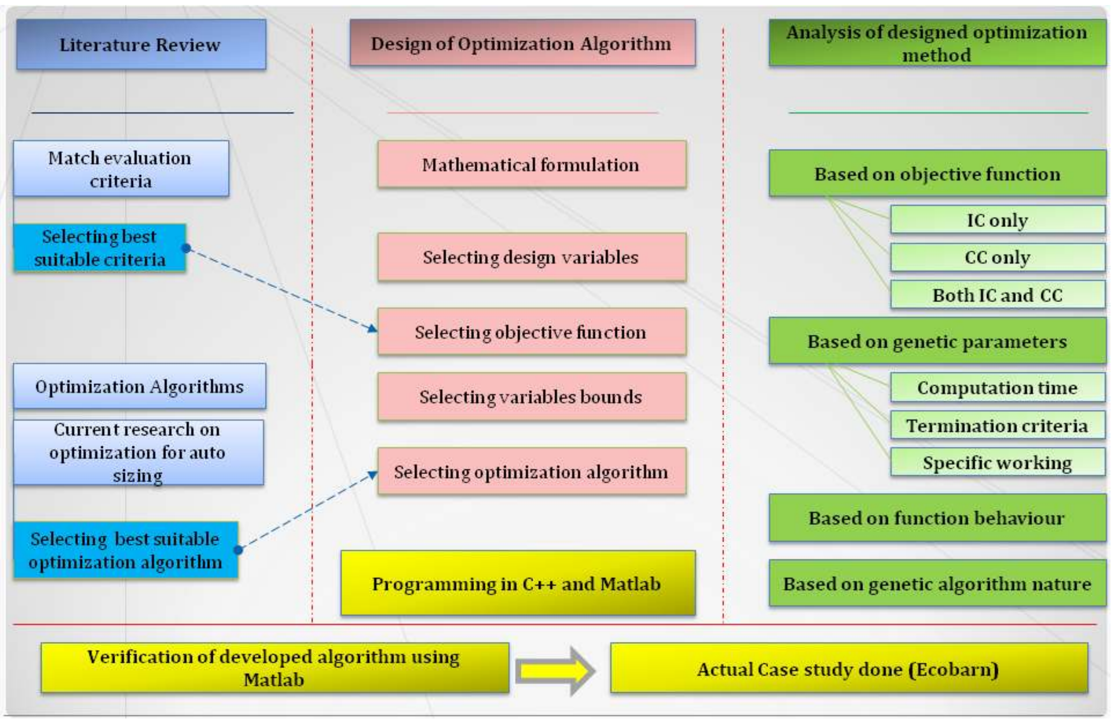

Literature Review
Design of Optimization Algorithm
Analysis of designed optimization method
Match evaluation criteria
Selecting best suitable criteria
Optimization Algorithms
Current research on optimization for auto sizing
Selecting best suitable optimization algorithm
Mathematical formulation
Selecting design variables
Selecting objective function
Selecting variables bounds
Selecting optimization algorithm
Programming in C++ and Matlab
Based on objective function
IC only
CC only
Both IC and CC
Based on genetic parameters
Computation time
Termination criteria
Specific working
Based on function behaviour
Based on genetic algorithm nature
Verification of developed algorithm using Matlab
Actual Case study done (Ecobarn)

Figure 4: Methodology of Thesis Work

1.4. <u>Report Layout:</u>

This work has six chapters. Chapter 1 starts with introduction of thesis work and provide information about background, motivation, objectives and methodology for thesis work along with report layout.

Chapter 2 provides a review of existing literature on optimization process. It includes optimal problem formulation process in detail along with classification of optimization algorithms after which working of genetic algorithms for single and multi objective optimization is presented. Introduction about global optimization toolbox of Matlab as well as of C++ genetic Library 'GAlib' is also given in this chapter.

Chapter 3 is written mainly for understanding of optimization for renewable/low carbon energy auto sizing capacity problem. It first looks on nature of renewable/low carbon energy systems and currently used optimization methods for its auto sizing problem and then move towards reasons for selection of genetic algorithm for this problem. It also includes existing literature on different match evaluation criteria for renewable/low carbon energy systems.

Chapter 4 deals with formation and design of algorithm. Mathematical details of problem are given in detail after which flow charts of designed process and genetic algorithms are presented. Analysis of designed algorithm process is also given in this chapter based on different genetic parameters, different objective functions, nature of genetic algorithm and objective function behaviour. Detail discussion on reason behind design of particular generated design with its benefits is also provided in this chapter.

Chapter 5 deals with the information about verification and case study for the designed algorithm method while Chapter 6 provides conclusion as well as possible further work area for this thesis work.

Chapter 2

2. Background on Optimization Process:

The process of optimization is of concern for man from many ages. Previously there were no defined and scientific rules for optimum conditions. But with the passage of time and advancement in science and technology, everything was considered to be based on certain reason or logic. Mathematical calculations involving process of optimization have become more famous in recent years. True meaning of optimization is to find the best answer for a particular problem. For example, problems dealing with the cost will require the best cost to be as less as possible. On the other hand, problems dealing with profit will see the maximum value as the best answer. So 'Optimum' is the word which is used to demonstrate the meaning of best and the process of finding best solution to a particular problem is known as process of optimization (Antoniou & Lu, 2007) .

2.1. Optimal problem formulation:

To solve an optimization problem, there is a particular optimal formulation procedure. It is not possible to apply single optimal formulation procedure for every design problem because objective functions and the associated parameters in optimization are different for different problems. The main purpose of formulation procedure is to make the mathematical model of optimal design problem which is then solved by using some optimization algorithm. Optimization algorithm accepts the optimization problem in a particular format. Deb (2005) has provided this format in his book which is shown in figure 5.

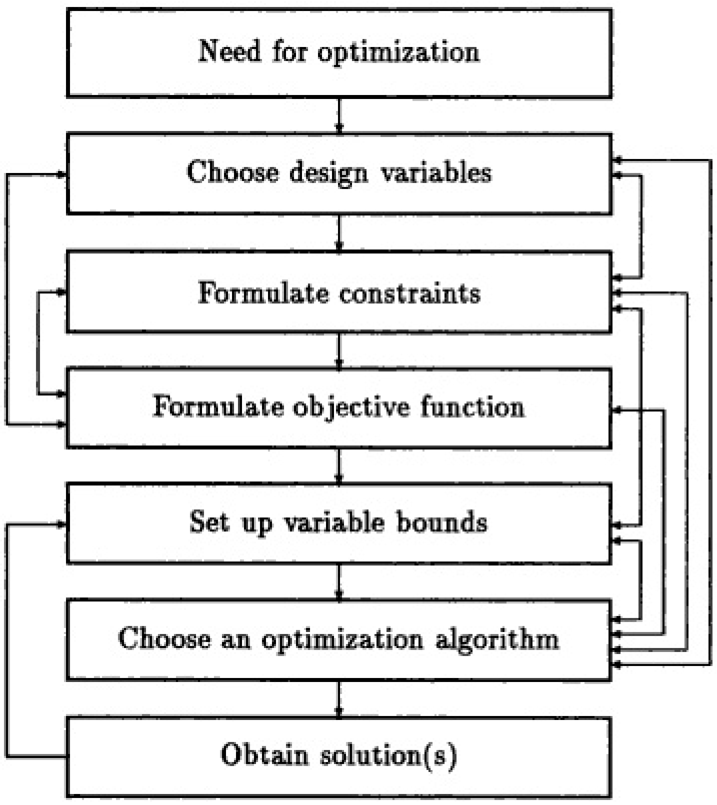


Figure 5: A flow chart of optimal design procedure (Deb, 2005, p.3)

The first step in designing an algorithm is to understand the need and purpose of optimization. After which design variables, constraints, objective function, variable bounds, and algorithms are selected for that particular problem. These steps are explained below.

2.1.1. Design variables:

A design problem is associated with many variables some of which are more sensitive to the problem. Design variables selection is dependent on user. These are the variables which are primarily varied in optimization process. Efficiency and speed of the optimization algorithm is largely dependent on number of design variables. So the efficiency can be improved by selectively choosing design variables. Deb (2005) says that first thumb rule is to select as few design variables as possible then outcome of optimization must be analysed to check that weather number of variables should be increased or decreased.

2.1.2. Constraints:

After selecting design variables, next step is to choose constraints. Constraints show some type of relationship with design variables and other design parameters satisfying some physical phenomena or resource limitations. Some of these considerations include design to stay within static or dynamic equilibrium. For example in many Mechanical engineering problems there are some limitations on stress and deflection. There is no particular format for defining the constraints as they vary from problem to problem and depend on the user.

There are two types of constraints; equality and inequality. Most of the constraints in the design problems are of inequality type. They state a design variable is either smaller than, greater than, or equal to a resource value. Example of inequality constraint is **"$\sigma(x) \leq S_{allowable}$"** which shows that stress developed **"$\sigma(x)$"** in the component anywhere must be smaller than and equal to allowable length **"$S_{allowable}$"**. Other type of constraint is of equality type and they state that functional relationship must exactly match the resource value. Example of equality constraint is **"$\delta(x) = 5$"** which shows that deflection **"$\delta(x)$"** at some point must be exactly equal to 5mm. Equality constraints are more difficult to handle as compared to inequality constraints. Deb (2005) states in his work that second thumb rule in the formulation of optimization problems is to keep number of complex equality constraints as low as possible.

2.1.3. Objective Function:

The third step in optimal problem formulation is to select objective function in terms of design variables and other problem parameters. There can be more than one objective function for real problems of optimization but due to their complex nature high number of objective functions is mostly avoided. Objective functions are of two types. Either they have to be maximized or to be minimized. If algorithm is developed with the objective of minimization then it can be converted to maximization by just simply multiplying the objective function by -1. For example, $f(x) = x^2+1$ is the objective function of maximization then the duality principle suggests that this problem is equivalent to minimization of $f(x) = -(x^2+1)$ but once the solution is obtained then the original function value must be obtained by multiplying solution with -1 (Deb, 2005, p.7). The duality principle is shown in figure 6.

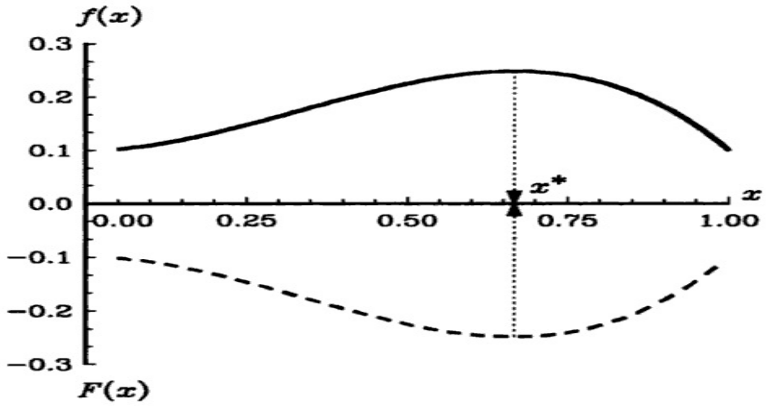


Figure 6: Duality principle show maximum point of f(x) is same point as the minimum of F(x) (Deb, 2005, p.8)

2.1.4. Variable bounds:

To define variable bounds is next step in which minimum and maximum bounds are set on design variables. This information is not necessary in some algorithms but some others need it. Variable bounds describe that solution points must be between them.

After these steps of formulation, optimization algorithms is selected based on above information and problem is written in a special mathematical form known as non linear programming format (NLP) which is then solved. All above written tasks are dependent on each other.

2.2. <u>Review of optimization Algorithms:</u>

An algorithm is assumed to be a sequence of statements which can be readable by computer and has some unambiguous meaning and it always has some input and output. Graphical representation of an algorithm is shown in figure 7. (Rieger & Hartmann, 2002, p.9).

Input
Algorithm
$\mathcal{A}$
Output

Figure 7: Graphical Representation of an algorithm (Rieger & Hartmann, 2002)

In mathematics, the procedure of generating the sequence of solutions for a particular problem is known as iterative method and an algorithm is a specific form of iterative method. An optimization algorithm is the algorithm which is used to define an optimized solution for a particular function. For example, for a function f(x), optimized solution would be the value of x for which f(x) is as small as possible or as large as possible where x has some constraints on it. The value of x could be scalar or a vector consisting of continuous or discrete values.

2.2.1. General classification:

The optimization problems can be classified in various ways and a simple classification is shown in figure 8 (Collette & Siarry, 2004, p.17)

Based on number of decision variable	• Monovariable • Multi variable
Based on type of objective function	• Function is linear with respect to decisions variables: *Linear* • Function is quadratic with respect to decision variables: Q*uadratic* • Function is nonlinear with respect to decision variables:N*onlinear*
Based on form of problem	• With constraints: *Constrained* • Without constraints: *Unconstrained*
Based on type of decision variable	• Continuous real number: *Continuous* • Integer number: *Integer or Discrete* • Permutations on a set of numbers of finite size: *Combinatorial*
Based on properties	• Online optimization: time of optimization is very short of order of milli second • Offline optimization: time of optimization is not given importance

Figure 8: Classification of optimization algorithms

Apart from this classification described above, classification of optimization algorithms can also be given as based on method of operations (Weise, 2009, pp.22-24).

2.2.2. Classification based on method of operations:

For global optimization problems, classification of optimization algorithms according to method of operation is given by Weise (2009) shown in the figure 9.

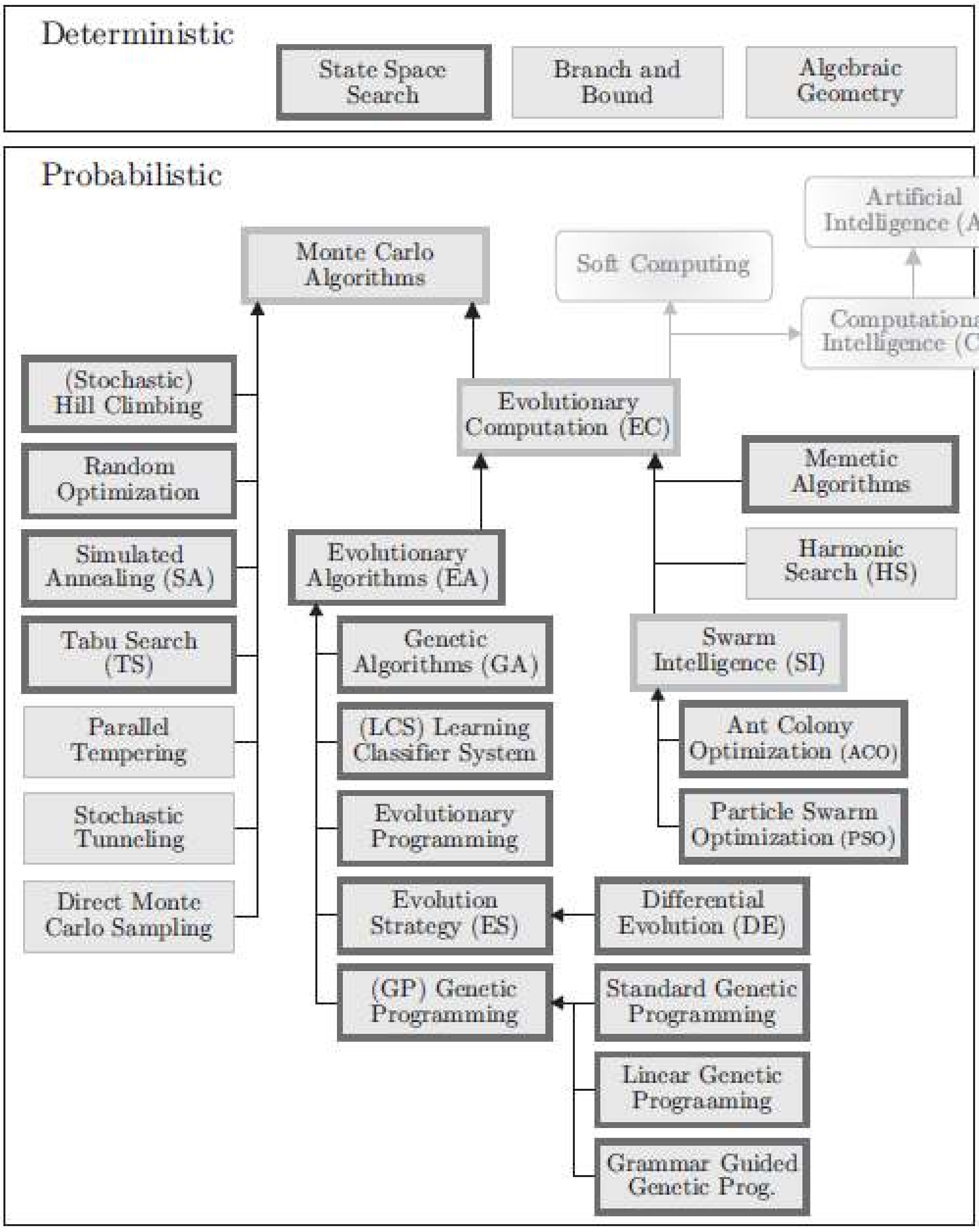

Deterministic
State Space Search
Branch and Bound
Algebraic Geometry
Probabilistic
Artificial Intelligence (AI)
Monte Carlo Algorithms
Soft Computing
Computational Intelligence (CI)
(Stochastic) Hill Climbing
Evolutionary Computation (EC)
Random Optimization
Memetic Algorithms
Simulated Annealing (SA)
Evolutionary Algorithms (EA)
Harmonic Search (HS)
Tabu Search (TS)
Genetic Algorithms (GA)
Swarm Intelligence (SI)
Parallel Tempering
(LCS) Learning Classifier System
Ant Colony Optimization (ACO)
Stochastic Tunneling
Evolutionary Programming
Particle Swarm Optimization (PSO)
Direct Monte Carlo Sampling
Evolution Strategy (ES)
Differential Evolution (DE)
(GP) Genetic Programming
Standard Genetic Programming
Linear Genetic Prograaming
Grammar Guided Genetic Prog.

Figure 9: The taxonomy of Global Optimization Algorithms (Weise, 2009, p.23)

According to this classification, there are two main types of optimization algorithms which are known as Deterministic and Probabilistic Algorithms.

2.2.2.1. Deterministic Algorithms:

If there exists a clear relationship between the characteristics of the possible solutions and their utility for a given problem then most often these algorithms are used. In formal words, it is an algorithm which behaves predictably. For a particular input, it will always produce the same output. The machine under this type of algorithms will undergo same sequence of steps. The existing state of the machine is determining

what would be the nest stage for machine (Wikipedia, the free encyclopedia, 2011). "In each execution step of a deterministic algorithm, there exists at most one way to proceed. If no way to proceed exists, the algorithm has terminated." (Weise, 2009).
One example of deterministic algorithms is hill climbing in which for the same starting point it will follow the same path always when it is run.

2.2.2.2. Probabilistic Algorithms:

If the relationship between solution candidate and its fitness is complicated or not obvious then they cannot be solved deterministically. Then to deal with such type of problems probabilistic algorithms are used. They use some kind of randomness in their logic. They include input which is consisted of uniformly random bits in hope of achieving good performance. Either running time of algorithm or the output or both are the random variables in these algorithms (Wikipedia, the free encyclopedia, 2011). They can be referred to stochastic or randomized algorithms. "A randomized algorithm includes at least one instruction that acts on the basis of random numbers. In other words, a randomized algorithm violates the constraint of determinism." (Weise, 2009).
However these algorithms have the chance to produce incorrect results based on the random inputs by signalling the type of error or by showing its failure to termination but still in many practical problems they are the only way to solve a problem. One example of these algorithms is genetic algorithm in which strings or solutions in the program will be different every time when the program is run. Though their final results do not have much difference but the path of each individual is not exactly repeatable. There are two main types of probabilistic algorithms. One is Las Vegas algorithms and other is Monte Carlo algorithms.

- ***Las Vegas algorithms:*** These algorithms never return a false value. Either they show the failure to proceed or do not return any value at all. If it returns some value then its output is deterministic. Their termination cannot be guaranteed and they usually have an expected runtime limit because their actual execution may be very long (McConnell, 2007). Weise (2009) stated that a Las Vegas algorithm terminates with a positive probability and is (partially) correct.

- ***Monte Carlo algorithms:*** These algorithms are the randomized algorithms whose running time is deterministic but whose output may be incorrect with certain small probability. (Wikipedia, the free encyclopedia, 2011). Weise (2009, p.552) stated that it is an algorithm which terminates always which is its main difference from Las Vegas algorithms.

However, sometimes probabilistic algorithms are classified as heuristic and metaheuristic algorithms (Weise, 2009).

- ***Heuristic:*** A heuristic is a part of an optimization algorithm that uses the information currently gathered by the algorithm to help to decide which solution candidate should be tested next or how the next individual can be produced. They can also be referred as to find and discover solution by trial and error. One example of Heuristics is travelling salesman problem.

- ***Metaheuristic:*** A metaheuristics is a method for solving very general classes of problems. It combines objective functions or heuristics in an abstract and hopefully efficient way, usually without utilizing deeper insight into their structure i.e., by treating them as black-box-procedures. The examples of metaheuristics are hill climbing/greedy search, tabu search, simulated annealing, genetic algorithms and ant colony optimization. It can be applied to all problems as these algorithms do not actually know what problems they are solving for.

There is a third type of algorithm which is the mixture of deterministic and probabilistic algorithms. For example hill climbing with a random restart where the basic idea is to use deterministic algorithm but it starts with different initial points. This hybrid algorithm is obviously having some advantages on simple techniques. However as they include randomness so they are put under probabilistic algorithms (Yang, 2010).

2.3. Evolutionary Algorithms:

"Evolutionary algorithms are population-based metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively" (Weise, 2009). Main advantage of evolutionary algorithms to other optimization method is their “black box” character that makes only few assumptions about the underlying objective function. They are utilizing some phenomena of biological evolution for example reproduction, mutation, recombination and selection. They operate on a population of potential solutions and apply the principle of survival of the fittest to get close to solution gradually. Solution is improved at each generation and the process leads to evolution of population that is best suited for particular environment.

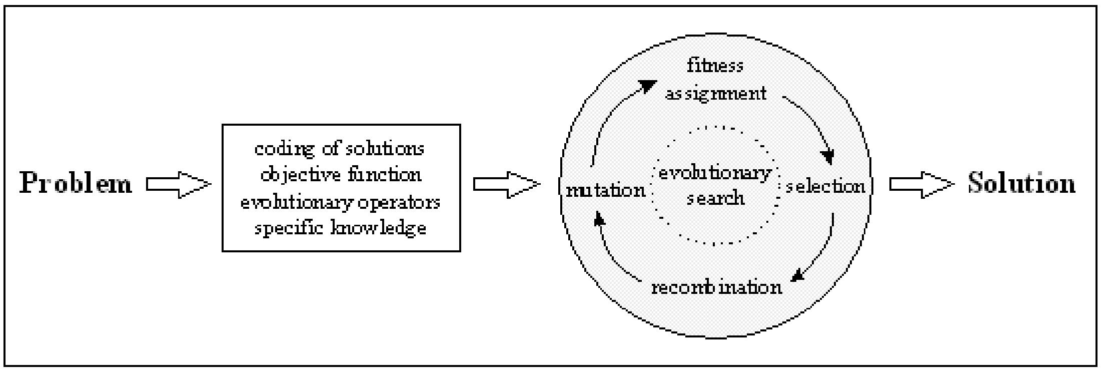


Figure 10: Solution of problem using evolutionary algorithms (Pohlheim, 2006)

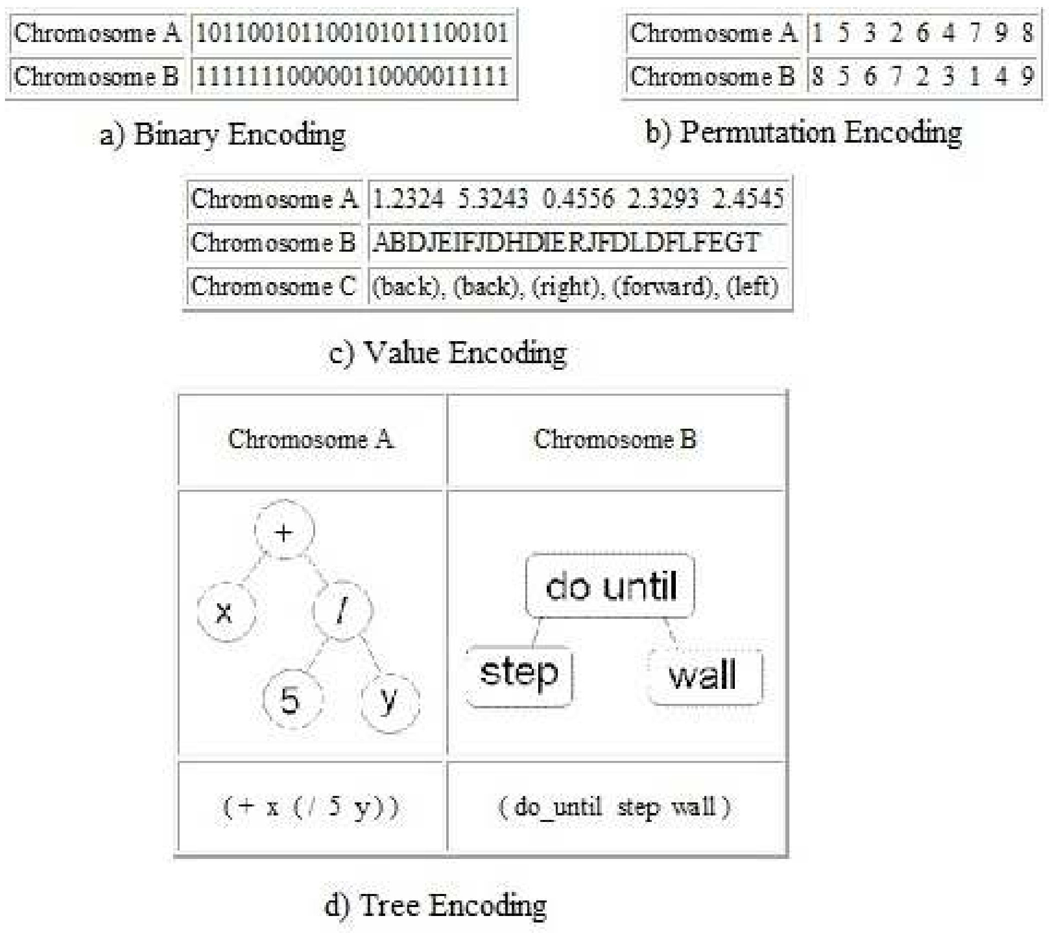
Figure 9 show that there are five subtypes of evolutionary algorithms. The family of evolutionary algorithm include Genetic Programming (GPs), Learning classifier systems (LCS), Evolutionary strategy (ES), Evolutionary programming (EPs) and Genetic algorithms (GAs).

Genetic Programming (GPs) includes all evolutionary algorithms that grow program and algorithms and things like that. Evolutionary programming is an approach that treats the instances of genomes as different species rather than as individuals. Also it has now more or less merged in genetic programming and other evolutionary algorithms. LCS (Learning classifier systems) are online learning approaches that assign output values to the given input values. They internally use a genetic algorithm to find new rules for this mapping. Here it can be stated that GPs (Genetic programming) are well suited for problems that require the determination of a function that can be simply expressed in a function form. Also ES (Evolutionary strategy) and EPs (Evolutionary programming) are well suited for optimizing continuous functions and finally GAs (Genetic algorithms) are well suited for optimizing combinatorial problems (though they have occasionally been applied to continuous problems) (Gray et al., 1997).

2.3.1. <u>Genetic Algorithms (GA):</u>

Genetic algorithms are subtype of evolutionary algorithms and inspired by Darwin's theory about evolution. GAs are adaptive heuristic search based on evolutionary ideas of natural selection and genetics. GAs are intelligent exploitation of random search used in optimization problems. Genetic algorithms start with a set of solutions called population. Solutions from one population are taken and used for to create new population. New population is considered better than old one. Solutions are selected to form new population on basis of their fitness. More is their fitness; more will be the chances to get selected for reproduction. This process is repeated until some termination condition (number of populations, improvement of best solution) is reached.

Before genetic algorithm is used in a real problem, there is need of encoding the solutions in the form which can be processed by computer. There are four types of coding available based on type of representation given in figure 11.



Chromosome A	101100101100101011100101
Chromosome B	111111100000110000011111

a) Binary Encoding

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

b) Permutation Encoding

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

c) Value Encoding

Chromosome A	Chromosome B
(+ x (/ 5 y))	(do_until step wall)

d) Tree Encoding

Figure 11: Types of encoding

Binary encoding is most commonly used in which every chromosome is a string of bits 0 or 1. Its example is Knapsack problem. Permutation encoding is used in ordering problems e.g. travelling sales man problem and in this encoding every chromosome is a string of numbers and numbers are represented in sequence. For problems in which complex values e.g. real number are involved, such as in finding weights of neural network, value encoding is used in which every chromosome is a string of some value which could be characters, real numbers or form numbers. Tree encoding is used for evolving programs, for example finding some value from a function, in which every chromosome is a tree of some object.

Initial population is chosen randomly then following steps are repeated until a termination criterion is met.

i. Evaluate fitness in order to maintain genetic diversity or differentiate between similar individuals raw objective scores are scaled to produce final fitness scores. There are different types of scaling schemes e.g. rank scaling, sigma truncation scaling, linear scaling and sharing (similar sharing).

ii. Prune population (typically all; if not then the worst)

iii. Select pairs to mate from best ranked individuals. There are various selections schemes defined e.g. rank selection, roulette wheel selection, tournament selection, Boltzman selection, and steady state selection.

iv. Replenish population by applying crossover and mutation operators. There are number of techniques available for cross over e.g. one point crossover, two point crossover, multipoint crossover, uniform crossover, arithmetic crossover, and heuristic cross over. All types involve swapping genes, sequences of bits in the strings, between two individuals (or between two strands of a diploid individual). Mutation alters one or more gene values in a chromosome from its initial state. Various types of mutation include Flip bit, boundary, non uniform, uniform and Gaussian etc.

v. Checking for termination criteria which can be number of generations, amount of time, minimum fitness threshold satisfied, fitness has reached a plateau, other.

Further detail of all these steps including parameters about GA can be found in books of Goldberg (1989), Mitchell (1998), Man et al. (1999), Gen & Cheng (1997) and Haupt & Haupt (2004).

2.3.2. Multi Objective Genetic algorithms (MOGA):

When there are more than one objectives involved in optimization then multi objective optimization is used. Fonseca and Fleming (1993) have first introduced the concept of multi objective genetic algorithm which used the non dominated classification of a GA population. MOGA differs from GA in a way fitness is assigned to each solution in the population. The rest of algorithm works in the same way as classical GA. Multi objective optimization provides a Pareto front which is non dominated set of solutions with regard to all objective functions. All solutions in Pareto front are optimal. Consider the plots of two functions in figure 12 which have their minima at point x=-2 and x=2 respectively. But in a multi objective problem, x = -2, x = 2, and any solution in the range -2 <= x <= 2 is equally optimal. The goal of the multi objective genetic algorithm is to find a set of solutions in that range (ideally with a good spread). The set of solutions is also known as a Pareto front (The MathWorks, 1994-2011).

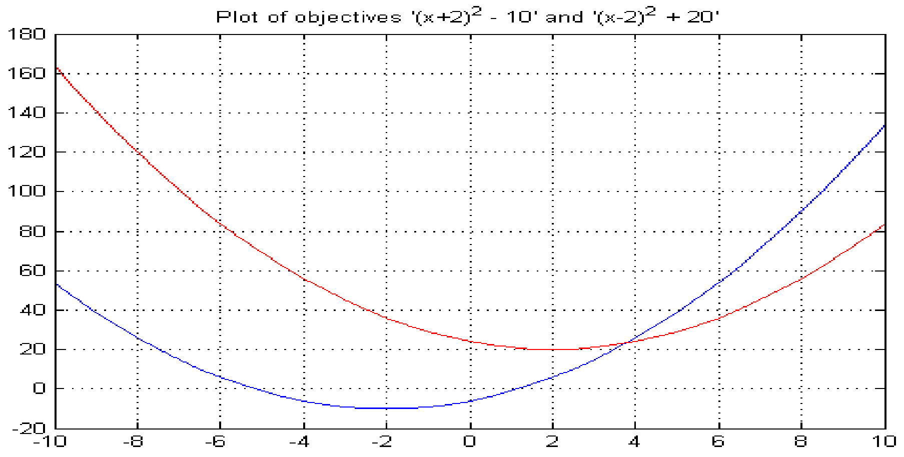


Figure 12: Concept of optimal solution in multi objective optimization

Figure 13 show Pareto front by red line and all points on Pareto front are more efficient as compared to any other point which is not on Pareto front. (Point *C* is not on the Pareto Frontier because it is dominated by both point *A* and point *B*. Points *A* and *B* are not strictly dominated by any other, and hence do lie on the frontier) (Wikipedia, 2011).

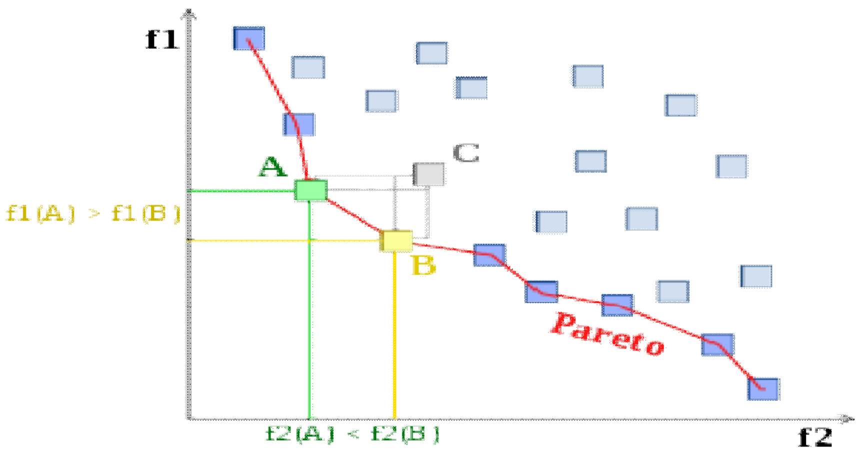


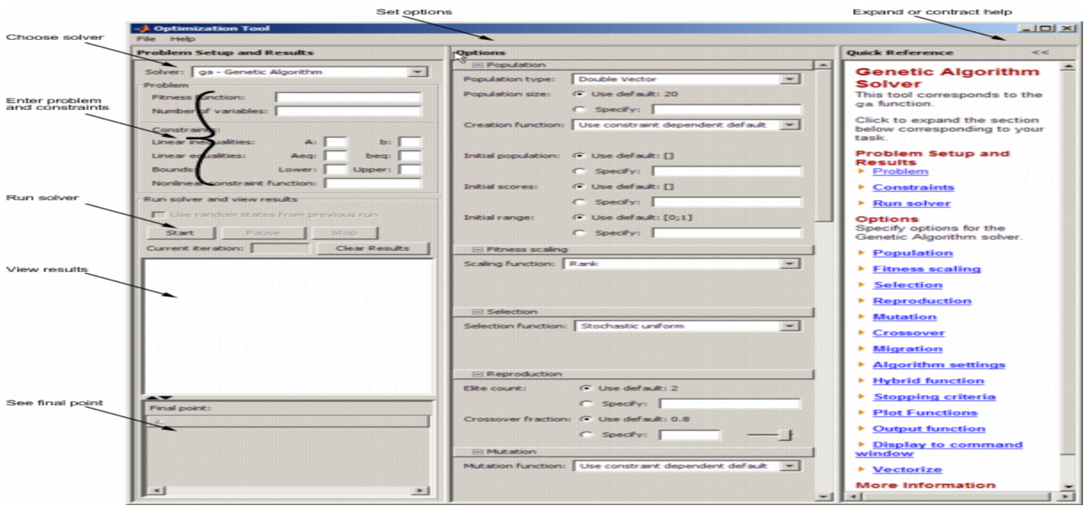
Figure 13: Concept of Pareto Front

2.3.3. <u>Software packages available for GA analysis:</u>

Due to increase in researches and development in field of genetic algorithms, there are many software packages available in market which can be used to analyse genetic algorithms behaviour. List of available GA packages is given in appendix B. Two packages are used for this thesis work as given below in detail.

1. **GA package in Matlab:**

GA package in Matlab is used for purpose of verification of designed algorithm. Global optimization tool box of Matlab (The Mathworks, 1994-2011) provide the solutions for those problems that have multiple maxima and minima involved and need global solutions. Matlab Global Optimization deals with optimizations where objective functions are continuous, discontinuous, stochastic, does not possess derivatives, or includes simulations or black-box functions with undefined values for some parameter settings (The Mathworks, 1994-2011). In Global Optimization tool box, genetic algorithm solver as well as multi objective genetic algorithm solver is used to analyse the behaviour of single objective genetic algorithm and multi objective genetic algorithm. Global Optimization tool box view is shown in figure 14.



Solver: ga - Genetic Algorithm
Problem
Fitness function:
Number of variables:
Constraints:
Linear inequalities: A: b:
Linear equalities: Aeq: beq:
Bounds: Lower: Upper:
Nonlinear constraint function:
Run solver and view results
Use random states from previous run
Current iteration:
Final point:

Population
Population type: Double Vector
Population size: Use default: 20
Creation function: Use constraint dependent default
Initial population: Use default: []
Initial scores: Use default: []
Initial range: Use default: [0;1]
Fitness scaling
Scaling function: Rank
Selection
Selection function: Stochastic uniform
Reproduction
Elite count: Use default: 2
Crossover fraction: Use default: 0.8
Mutation
Mutation function: Use constraint dependent default

Genetic Algorithm Solver
This tool corresponds to the ga function.
Click to expand the section below corresponding to your task.
Problem Setup and Results
- Problem
- Constraints
- Run solver

Options
Specify options for the Genetic Algorithm solver.
- Population
- Fitness scaling
- Selection
- Reproduction
- Mutation
- Crossover
- Migration
- Algorithm settings
- Hybrid function
- Stopping criteria
- Plot Functions
- Output function
- Display to command window
- Vectorize

More Information

Figure 14: Global Optimization tool box of Matlab (The Mathworks, 1994-2011)

The following tables 1 and 2 provide the information of standard genetic algorithm options and standard multi objective genetic algorithm solver options which can be used in global optimization tool box.

Table 1: Option of genetic algorithm available in Matlab (The Mathworks)

Step	**Genetic Algorithm Option**
Creation	Uniform, feasible
Fitness scaling	Rank-based, proportional, top (truncation), shift linear
Selection	Roulette, stochastic uniform selection (SUS), tournament, uniform, remainder
Crossover	Arithmetic, heuristic, intermediate, scattered, single-point, two-point
Mutation	Adaptive feasible, Gaussian, uniform
Plotting	Best fitness, best individual, distance among individuals, diversity of population, expectation of individuals, max constraint, range, selection index, stopping conditions

Table 2: Option of multi objective genetic algorithm solver available in Matlab (The Mathworks)

Step	**Multi objective Genetic Algorithm Option**
Creation	Uniform, feasible
Fitness scaling	Rank-based, proportional, top (truncation), linear scaling, shift
Selection	Tournament
Crossover	Arithmetic, heuristic, intermediate, scattered, single-point, two-point
Mutation	Adaptive feasible, Gaussian, uniform
Plotting	Average Pareto distance, average Pareto spread, distance among individuals, diversity of population, expectation of individuals,

	Pareto front, rank histogram, selection index, stopping conditions

Global Optimization Toolbox can also allow changing population size, number of elite children (not in case of multi objective genetic algorithm), crossover fraction, migration among subpopulations (using ring topology), Pareto front (not in case of genetic algorithm) and bounds, linear, and nonlinear constraints for an optimization problem. Stopping criteria can also be selected based on time, stalling (not in case of multi objective genetic algorithm), fitness limit, or number of generations.

2. GAlib: A C++ library of Genetic Algorithms

GAlib provides object oriented C++ classes and objects to implement genetic algorithms. It basically works with two classes, genome and genetic algorithm where genome represents the single solution to problem and genetic algorithm defines how evaluation must take place. The library contains different types of genomes (GAListGenome, GATreeGenome, GAArrayGenome and GABinaryStringGenome etc.) and different types of genetic algorithm (simple, steady-state, and incremental). It has many built in genetic parameters including elitism, selection strategies and replacement strategies and has the possibility to customize them as well according to need of user. New genetic algorithms can be quickly tested by deriving from base genetic algorithm classes in the library (Fahimuddin, 2003). Built in options available in galib library is listed in table 3. Further detail of GAlib can be found out from its website (Wall, n.d.).

Table 3: Built in options in GAlib

Step	**GALib Genetic Algorithm Option**
Built-in chromosome type	Real number arrays, list, tree, 1D, 2D, and 3D arrays, 1D, 2D, and 3D binary string.
Built-in initialization operators	Uniform random, order-based random, allele-based random, and initialize-to-zero.
Built-in selection methods	Rank, roulette wheel, tournament, stochastic remainder sampling, stochastic uniform sampling, and deterministic sampling.
Built-in crossover operators	arithmetic, blend, partial match, ordered, cycle, single point, two point, even, odd, uniform, node- and subtree-single point
Built-in mutation operators	Random flip, random swap, Gaussian, destructive, swap subtree, swap node.
Type of objective function	Population- or individual-based.

Chapter 3

3. Optimization of Renewable/Low Carbon Energy Systems:

This chapter is presenting literature for currently used optimization methods in field of renewable/low carbon energy and scope of present and future researches for sizing problems of hybrid renewable energy systems. In the light of already employed optimization methods for RE, nature of RE, and type of particular problem of auto sizing, genetic algorithms are selected for developing optimization algorithm for auto sizing problem of renewable/ low carbon energy and benefits of genetic algorithms are provided in detail. Then different available match evaluation criteria (especially available in Merit tool) are discussed for renewable energy supply and demand matching along with their advantages and disadvantages to select best one to be used as objective function in optimization algorithm for auto sizing problem of renewable/ low carbon energy for match evaluation. Inequality coefficients ideally and correlation coefficients to some extents are proved to be good match evaluation criteria in light of literature review.

3.1. Currently used optimization methods for Renewable Energy systems:

Improvement in renewable/low carbon energy technologies is required for sustainable development and to reduce energy problems. Optimization algorithms are a suitable tool for solving problems related with complex nature of renewable/low carbon energy systems. There are several methods available for renewable energy systems using optimization methods e.g. renewable energy models, energy supply models, energy planning models, emission reduction models, energy supply demand models, control models etc (Jebaraj & Iniyan, 2006). Banos et al. (2011) provided list of all researches which have been done related to use of optimization algorithms for design, planning and control problems associated with renewable energy. Initially optimization methods were based on traditional approaches e.g. mixed-integer and interval linear-programming, Lagrangian relaxation, quadratic programming, and Nelder–Mead Simplex search. But in recent years, non traditional approaches i.e. heuristic optimizations have become more famous which include especially genetic algorithms and practical swarm optimization. There are also multi-objective problems related to renewable energy systems using Pareto-optimization techniques. It can be concluded that the use of heuristic approaches, Pareto-based multi-objective optimization and parallel processing are promising research areas in the field of renewable and sustainable energy (Banos et al., 2011).

The researches about sizing problems related to renewable energy include most of work related to wind and solar energy systems. For sizing standalone photovoltaic's, grid connected photovoltaic's and for photovoltaic-wind hybrid systems, Mellit (2009) studies performance of artificial intelligence techniques. ANN and Genetic Algorithms were used by Mellit et al (2010) for sizing problems of photovoltaics. Thiaux et al. (2010) applied NSGAII which is a fast multi objective GA to optimize stand-alone photovoltaic systems while Kornelakis and Marinakis (2010) applied PSO to such problems. Anagnostopoulos and Papantonis (2007) utilized a stochastic EA for the optimal sizing of a small hydropower plant with objective of maximizing the economic benefit and the energy produced. A GA was used for optimal sizing of stand-alone photovoltaic-wind generator systems, which selects the optimal number and type of units to minimize the cost subject to the constraint that the load energy requirements are completely covered (Koutroulis et al., 2006). GA are also employed for optimal sizing to optimize the configurations of a hybrid solar–wind system employing battery banks, where the decision variables are the number of photovoltaic modules, wind turbines and batteries, the photovoltaic module slope angle and wind turbine installation height (Yang et al., 2008). Bilal et al. (2010) used multi objective GA for sizing a hybrid solar–wind-battery system with objective of minimizing the annualized cost system and the loss of power supply probability while Moghaddas-Tafreshi (2009) used PSO for sizing problems. Papantonis (2008) combined an evaluation algorithm that simulates in detail the plant operation and an automated optimization software based on EA for optimum sizing of the various components of a reversible hydraulic system, i.e. turbine size, the size and the number of the pumps, the penstock diameter and thickness, the capacity of the reservoirs and some financial parameters.

3.2. Genetic algorithm & auto sizing of Renewable/low carbon Energy:

Literature review on classification of optimization algorithms is analysed with particular nature of auto sizing problem of renewable/low carbon energy systems. The particular problem lies under

- Multi variable optimization as problem deals with finding optimum capacity of more than one supply options available.
- Non linear optimization as renewable energy supply and demand profiles are non linear involving many maxima and minima.
- Constrained optimization as auto sizing of renewable/low carbon energy supply deals with finding capacity in some particular constraint e.g. cost.
- Dynamic optimization as renewable/low carbon energy profiles change their value at each time step and hence can be represented as a function of time
- Discrete as the considered sizes of each supply can take only discrete values.
- Combinatorial optimization as sizes of particular supplies can only take specific values and sizing problem is to deal with only finite number of possible values.
- Probabilistic optimization as renewable/ low carbon energy is variable and stochastic in nature because of their high dependency on climatic conditions.

It is clear that problem lies under probabilistic optimization. Under probabilistic algorithms, metaheuristics produce high quality results as compared to heuristic and as described by Yang (2010) that metaheuristics are suitable for global optimization problems. Also under probabilistic algorithm, Monte Carlo algorithms are better than Las Vegas because they always terminate and produce some results. Evolutionary algorithms are type of metaheuristics as well as of Monte Carlo algorithms. Evolutionary algorithms further consisted of five types from which genetic algorithm are the one which are well suited for combinatorial problems (Gray et al., 1997). So selection of algorithms well suited for renewable energy problems is given in following figure.

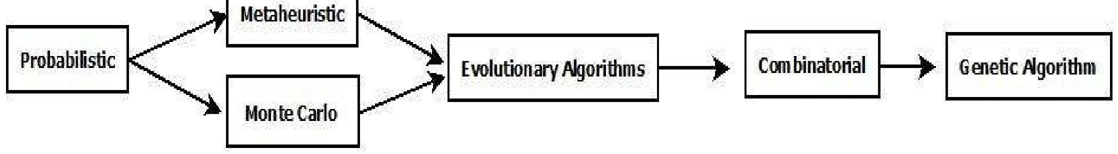


Figure 15: Step by step selection of algorithm

It can be noted from figure 9 that there are also other algorithms along with evolutionary algorithm which comes under Monte Carlo and Meta heuristics for example Hill climbing, simulated annealing, tabu search and swarm intelligence etc. The reasons behind selection of GA which comes under evolutionary algorithms are provided here. Hill climbing simulated annealing and tabu search deal with a single solution rather than with populations of solutions and therefore can't explore the neighbourhood of the whole population. They are also termed as local search methods and their main disadvantage is that they get stuck in local optima.

GA is population based, inspired by nature and search globally. However swarm intelligence algorithms (PSO and ACO) are also inspired by nature and proved good for optimization problems but researches show that GA approach is superior to PSO approach in terms of its computational time/efforts (Jones, 2005). Artificial neural network (ANN) is also effected by problems of local optima and to get good results from them they must be used combine with other methods such as GA.

Also current researches in field of renewable energy optimization problems related to sizing are mostly about evolutionary algorithms (genetic algorithms and multi objective genetic algorithms). Initially, applications of genetic algorithm were mainly theoretical. With the increase in research, growth of computing power and development of internet, they moved in commercial sector. Now they are solving problems of every day interest and found wide applications in areas of acoustics, aerospace engineering, astronomy and astrophysics, chemistry, electrical engineering, financial markets, game playing, geophysics, materials engineering, mathematics and algorithmics, Military and law enforcement, molecular biology, pattern recognition and data mining, robotics, routing and scheduling and systems engineering.

Further benefits of genetic algorithm explained by Haupt & Haupt (2004, p.23) and Marczyk (The talk origin archives, 2004) which are in favour of particular auto sizing problems include

- It can deal with problems with continuous as well as discrete variables. Some experimental works (Water et al., 1998) show that for some class of problems which deal with highly discrete variables, GA , because of their inherent discrete nature, can be more accurate than other algorithms built originally for continuous variables (Solomatine, 1998).

- They can perform well on a wide variety of test functions, including noisy, discontinuous, and multimodal search landscapes (Goldberg, 1989, p.107).

- Compared with traditional methods (the direct exhaustive search method and the gradient-directed search method) for function optimization, one of the main advantages of the GA is that it is generally robust in finding global optimal solutions, particularly in multimodal and multi-objective optimization problems (Yang et al., 2008). However, even if a GA does not always deliver a provably perfect solution to a problem, it can almost always deliver at least a very good solution (Marczyk, 2004).

- Genetic algorithms are intrinsically parallel. Instead of exploring solution space in one direction, they explore in multiple direction at once as they have multiple offsprings. If one path is turn out to be dead, they eliminate that one and move towards other direction which is more promising in finding optimal solution in each run. More over by evaluating one particular string, they sample each of its spaces to which it belongs due to which they build up an increasingly accurate value for the *average* fitness of each of these spaces, each of which has many members. Therefore, a GA that explicitly evaluates a small number of individuals is implicitly evaluating a much larger group of individuals. This phenomenon is known as schema theorem and is main advantage over other problem solving techniques (Goldberg, 1989).

- They are suitable for solving problems in which search space is really huge which is usually the case of non linear problems. In case of linear problems, improvement in one part will improve the systems as a whole while in case od non linear problems, improvement at one point is not having much effect on the entire system. Search space of non linear systems is require search 1000 of times more than that of the linear systems resulting in exhaustive search. But GA due to producing multiple schemas at once and parallelism complete such tasks in reasonable amount of time. So they can optimize variables with extremely complex cost surfaces.

- GA has ability to deal with many parameters simultaneously (Forrest, 1993). So they behave very well for multi objective problems.

- Another quality of genetic algorithm is that they solve the problem about which they never know before. They start with random solutions and use fitness function to find improvement which means *all* possible search pathways are theoretically open to a GA.
- It does not need the objective functions to be differentiable or continuous and can deal with a large number of variables.
- It provides a list of optimum number instead of single solution and can encode the variables so that the optimization is done with encoded variables.
- It works with numerically generated data, experimental data, or analytical functions and can solve different types of problems including bound-constrained and general optimization problems.

Based on above description, GA's are considered to be most suitable algorithms in terms of global optimization, particular nature of renewable energy and particular problem of auto sizing.

"Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime." - Salvatore Mangano Computer Design, May 1995

3.3. <u>Match Evaluation Methods for Renewable/Low carbon Energy:</u>

Objective function in case of Renewable/ Low carbon energy systems is to maximize match between supplies and demands i.e. to ensure that period of generation is matched with period of consumption. In particular case of renewable/low carbon energy systems, the accuracy of match is dependent on supply and demand profiles. Born in his work says that match is also effected by data resolution. For example a data obtained at very high frequency that is order of seconds can lead to poor match while time averaged over half hourly intervals can give improved results. Mathematical formula is required for computing objective function for its use in optimization algorithm (Born, 2001). Many match evaluation methods along with their mathematical relationships are given below.
'A building's self-sustenance is the ratio of the demand displaced by on-site generation, to the demand without generation. The optimal value for self-sustenance

is taken to be unity, whereby all the sites demand could be displaced by on site generation' (Mahdavi et al., 1999). The self sustenance index is given by the following equation.

$$ESS_x = \frac{\sum_{t=1}^{n} disp_{x,t}}{\sum_{t=1}^{n} dem_{x,t}}$$

-------------EQ 1

Where ESS_x = site energy self sustenance for energy type x, $disp_{x,t}$ = energy displacement by generation system, for energy type x, at time t, $dem_{x,t}$ = energy demand of building without generation, for energy type x, at time t, n = total number of time steps

Shared area between supply and demand profiles can also be used for match evaluation. This value can be approximated by evaluating the area between the x-axis and the lowest value between supply and demand for every time step. For a perfect match %SA would equal 100 where shared area can be evaluated by following equation.

$$\%SA = \frac{\left(\int_0^n D(t)\,dt\right) U \left(\int_0^n S(t)\,dt\right)}{\int_0^n D(t)\,dt}$$

-------------EQ 2

Where D (t) = demand profile, S (t) = supply profile, n = time period.

Excess supply is another criterion to describe match rate and must be 0 for perfect match where it can be given by following relationship.

$$\%ES = \frac{\int_0^n S(t)\,dt - \left[\left(\int_0^n D(t)\,dt\right) U \left(\int_0^n S(t)\,dt\right)\right]}{\int_0^n S(t)\,dt}$$

-------------EQ 3

The residual, r (t), of two profiles can be used to represent the combined profile and can be obtained by subtracting the supply at each time step from the demand.

$$r(t) = D(t) - S(t)$$

-------------EQ 4

‘The least-squares approach can be used to quantify the magnitude of deviation between two sets of data variables’ (Born, 2001). The answer will always be a positive value, with a lower limit of zero indicating a perfect match and without an upper limit. The following equation defines this method.

$$LS = \sum_{t=0}^{n} (D_t - S_t)^2$$

-------------EQ 5

Spearman’s Rank Correlation Coefficient describes the correlation between any pair of variables by calculating the degree to which the variables fall on the same least square line (Scheaffer & McClave, 1982). Its value varies from ‘-1’ to ‘1’. ‘1’ shows the perfect positive correlation and ‘-1’ show the perfect negative match where ‘0’ represents no match. Correlation coefficient can be given by the following equation.

$$CC = \frac{\sum_{t=0}^{n} (D_t - d)\cdot(S_t - s)}{\sqrt{\sum_{t=0}^{n} (D_t - d)^2 \cdot \sum_{t=0}^{n} (S_t - s)^2}}$$

-------------EQ 6

Where D_t = demand at time t, S_t = supply at time t, d = mean demand over time period n, s = mean supply over time period n

‘The Inequality Coefficient, IC, describes the inequality in a time-series due to three sources: unequal tendency (mean), unequal variation (variance) and imperfect co-variation (co-variance)’ (Born, 2001). The value of Inequality coefficient varies from 0 to 1 where 0 shows good match and 1 show no match.

$$IC = \frac{\sqrt{\frac{1}{n}\cdot\sum_{t=0}^{n} (D_t - S_t)^2}}{\sqrt{\frac{1}{n}\cdot\sum_{t=0}^{n} (D_t)^2} + \sqrt{\frac{1}{n}\cdot\sum_{t=0}^{n} (S_t)^2}}$$

-------------EQ 7

Born (2001) in his work has used an example to show different merits and demerits for these match criteria. Energy self-sustenance index accounts for displaced energy but neglects excess energy production which can also affect the quality of match. When comparing scenarios where one of the profiles is common, i.e. a demand profile matched to a number of supplies or vice-versa, the shared area can be used to compare the individual matches. However Shared area can provide information about the demand portion satisfied by renewable energy but cannot give any indication of excess energy. Shared area and excess energy both are very important to describe a match. For a perfect match %SA would equal 100 and %ES zero and addition of these terms yields an optimum value of 100, however non-perfect match values could range above and below this figure, making judicious comparisons difficult. Similarly, residual cannot describe match between supply and demand accurately. Due to lack of an upper limit in case of least square method quality of match is difficult to find. Where numerous profile pairs are to be compared, bands defining the quality of match are useful in processing various possibilities, although establishing such bands is extremely difficult where a worst case cannot be defined. Correlation coefficient is describing trend between two data sets but ignores relative magnitude of the two. For example if two profiles are exactly in phase but of very different magnitude would result in perfect correlation. For a perfect match, magnitude and phase, both must be considered. Still importance of CC cannot be ignored as it provides ‘a measure of the potential match that could exist given changes to the relative capacities, i.e. through energy efficiency or altering the size of the RE system’. However, Inequality coefficient is the one which can be ideally used to describe match rate. Smaller the inequality, larger will be the match. Values of Inequality Coefficient (IC) between 0-0.4 represent good matches and values above 0.5 show bad matches.

To sum up, for defining match between supply and demand profiles of renewable energy systems, inequality coefficient can prove to be best criteria and correlation coefficient is also good but up to some extent.

Chapter 4

4. Formation of Optimization Algorithm for Auto sizing:

4.1. Introduction:

This chapter deals with the development of optimization algorithm for auto sizing capacity of renewable/low carbon energy systems. The auto sizing problem is first mathematically formulated and most suitable genetic parameters are selected. And then analysis of objective function IC and CC, optimization method and working of algorithm leads towards a particular design for particular auto sizing problem.

4.2. Developed optimization algorithm

An optimization algorithm is developed for auto sizing capacity of renewable/ low carbon energy systems. The optimization algorithm is developed for the purpose of finding optimized capacities of each supply while there is n number of supplies available to satisfy a single demand. Optimization algorithm is designed and run in C++ computer language utilizing genetic algorithms. Flow chart of algorithm is shown in figure 16.

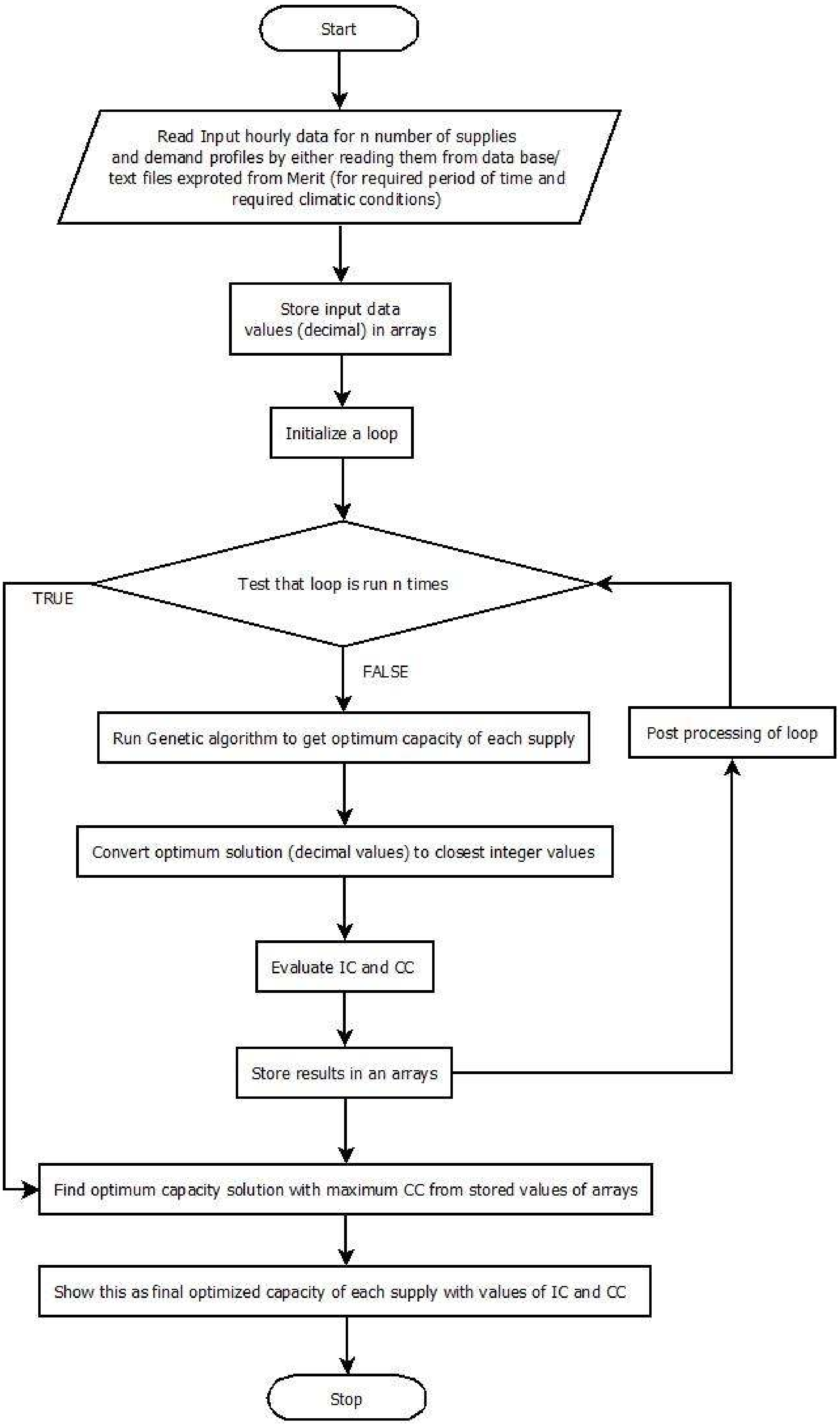

Start
Read Input hourly data for n number of supplies and demand profiles by either reading them from data base/ text files exproted from Merit (for required period of time and required climatic conditions)
Store input data values (decimal) in arrays
Initialize a loop
Test that loop is run n times
TRUE
FALSE
Run Genetic algorithm to get optimum capacity of each supply
Post processing of loop
Convert optimum solution (decimal values) to closest integer values
Evaluate IC and CC
Store results in an arrays
Find optimum capacity solution with maximum CC from stored values of arrays
Show this as final optimized capacity of each supply with values of IC and CC
Stop

Figure 16: Flow chart for optimization Process

The designed program starts with taking input values of hourly data for different time spans e.g. days, weeks, months or year etc. of demand and each renewable energy supply and storing them in form of arrays. Data base for hourly supply and demand profiles for different climates are exported from Merit. Then a for loop is run in program to call genetic algorithm several times with objective of minimizing inequality coefficient (IC). The results of optimum capacity generated by GA in decimal values are then converted to closest integer values in order to get whole value of each supply capacity. Optimum capacity results along with corresponding IC and CC value are recorded for each GA run and stored in arrays. From all stored results obtained, one result is selected as best optimized one which has maximum correlation coefficient.

The design of algorithms first needs optimal design formulation process (Deb, 2005) which is described in detail under section 4.2.1 and provides mathematical details and non linear programming format of auto sizing problem according to figure 5. Details of genetic algorithm (GA) used and selection of its parameters are given under section 4.2.2. Reasons for using inequality coefficient and correlation coefficient as objectives for match evaluation in a particular order as given in flow chart along with how many times GA is called/run is discussed in detail under section 4.2.4.

4.2.1. <u>Optimal design of auto sizing Problem:</u>

According to problem formulation procedure provided by Deb (2005), Particular problem for auto sizing of renewable/low carbon energy systems is defined first as shown graphically in figure 17.

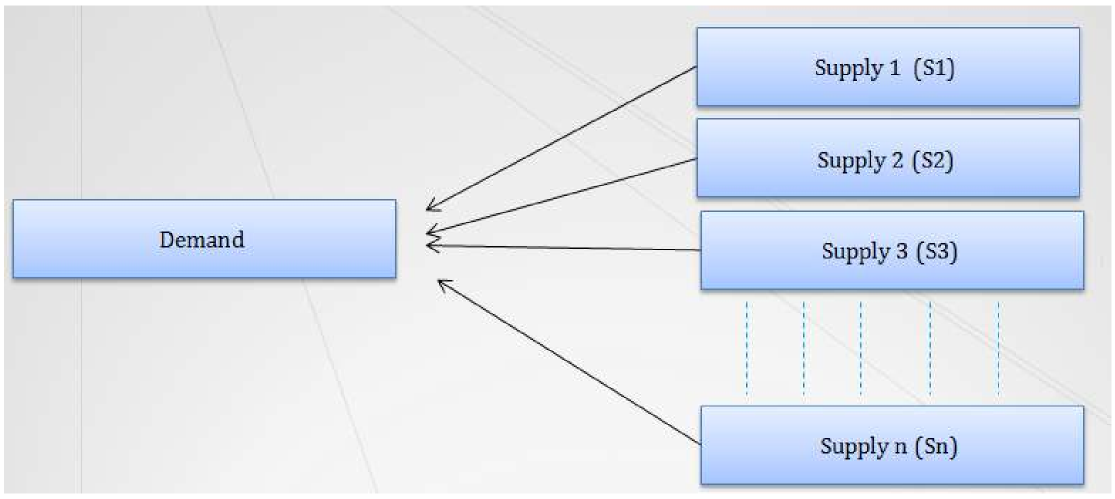


Figure 17: Graphical representation of auto sizing problem

Demand (D) is required to be matched with different supplies in a way that Resultant Supply (RS) = n_1 (S1) + n_2 (S2) + n_3 (S3) + ---------------+ n_n (Sn) where main objective of required auto sizing algorithm is to find the optimized values of "n_1, n_2, n_3, -----, n_n" to maximise match between demand and resultant supply where S1, S2, S3… Sn represents unit capacities of supplies. After defining problem, next step is to choose design variables. In this case, design variables are n_1, n_2, n_3, -----, n_n which describes how many number/units of different supplies of particular capacities are needed. When there are five supplies available to be matched with a demand then n=5 meaning that there are five design variables (n1, n2, n3, n4, n5).

After defining design variables, variable bounds are selected. Variable bounds include minimum and maximum limits on a particular supply capacity in order to generate a solution (size of each capacity) between certain ranges. There must be utilization of each considered renewable energy supply at least once so minimum value (lower bound) of design variables (n_1, n_2, n_3, ---, n_n) is selected as 1. For case of maximum bound on design variables, there is a flexibility which varies with case to case. If demand is in 10,000 of watts then maximum value of design variables (number of supplies) could be in 1000's. However in this particular auto sizing algorithm, a simple procedure is adopted to find upper bound on each design variables which depends on supply and demand hourly profiles data. Since data base of hourly demand and supplies profiles are stored in form of arrays so upper bound of design variables are found by dividing maximum value in demand array with minimum value of each supply array.

Non linear programming format of problem which is suitable for solving by utilizing some kind of optimization algorithm is given as below.

$$Minimize\ IC\ which\ is\ now\ = \frac{\surd[\frac{1}{n}.\sum_{t=0}^{n}\ \{Dt-(\mathrm{n1.St1+n2.St2+n3.St3+\cdots+nn.Stn})\}^2}{\surd[\frac{1}{n}.\sum_{t=0}^{n}\ (Dt)^2+\surd[\frac{1}{n}.\sum_{t=0}^{n}\ (\mathrm{n1.St1+n2.St2+n3.St3+\cdots+nn.Stn})^2}$$

-------------EQ 8

Where if time span considered is n then D_t = Demand at time t and S_t1, S_t2, S_t3, … , S_tn are values of S1, S2, S3, … , Sn at time t

$1 \leq n1 \leq up1$ Where $up1 = \frac{\max(\mathrm{D})}{\min(\mathrm{S1})}$ -------------EQ 9

$1 \leq n2 \leq up2$ Where $up2 = \frac{\max(\mathrm{D})}{\min(\mathrm{S2})}$ -------------EQ 10

$1 \leq n3 \leq up3$ Where $up3 = \frac{\max(D)}{\min(S3)}$ -------------EQ 11

…………

………

$1 \leq nn \leq upn$ Where $upn = \frac{\max(D)}{\min(Sn)}$ -------------EQ 12

Where max (D) is the maximum value of demand over considered time period and min (S1), min (S2), min (S3)… min (Sn) are the minimum values over considered time period of S1, S2, S3 … Sn respectively.

IC and CC are computed as follow

$$IC = \frac{\sqrt{[\frac{1}{n}\sum_{t=0}^{n} \{Dt-(n1.St1+n2.St2+n3.St3+\cdots+nn.Stn)\}^2}}{\sqrt{[\frac{1}{n}\sum_{t=0}^{n} (Dt)^2}+\sqrt{[\frac{1}{n}\sum_{t=0}^{n} (n1.St1+n2.St2+n3.St3+\cdots+nn.Stn)^2}} \quad \text{-------------EQ 13}$$

$$CC = \frac{\sum_{t=0}^{n}(Dt-d).\{(n1.St1+n2.St2+n3.St3+\cdots+nn.Stn)-s\}}{\sqrt{[\sum_{t=0}^{n} (Dt-d)^2+\sum_{t=0}^{n} \{(n1.St1+n2.St2+n3.St3+\cdots+nn.Stn)-s\}^2]}} \quad \text{-------------EQ14}$$

Where s=mean supply over time period n and d= mean demand over time period n

Also s and d are computed as follow

$$s = \frac{1}{n}\sum_{t=0}^{n}\{(n1.St1 + n2.St2 + n3.St3 + \cdots + nn.Stn)\} \quad \text{-------------EQ 15}$$

$$d = \frac{1}{n}\sum_{t=0}^{n}(Dt) \quad \text{-------------EQ 16}$$

4.2.2. <u>Details of genetic algorithm used:</u>

Genetic algorithm is used for optimization purposes and body of genetic algorithm is given in flow chart in figure 18.

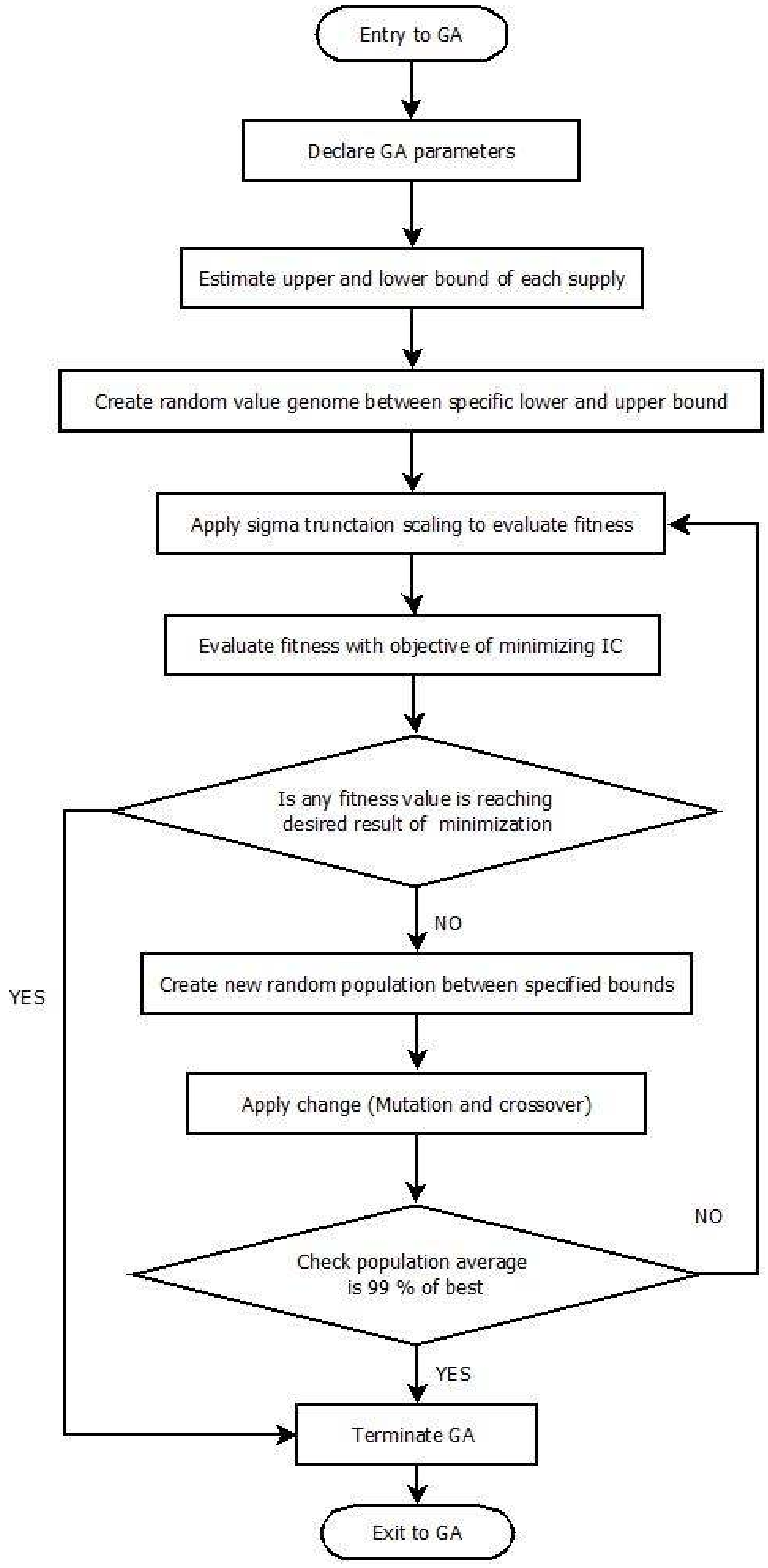

Entry to GA
Declare GA parameters
Estimate upper and lower bound of each supply
Create random value genome between specific lower and upper bound
Apply sigma trunctaion scaling to evaluate fitness
Evaluate fitness with objective of minimizing IC
Is any fitness value is reaching desired result of minimization
NO
YES
Create new random population between specified bounds
Apply change (Mutation and crossover)
Check population average is 99 % of best
NO
YES
Terminate GA
Exit to GA

Figure 18: Flow chart of genetic algorithm

The body of genetic algorithm starts with initializing population by randomly generating genes within particular range between lower and upper limit on design variables determined by a method as described in mathematical details under section 4.2.1. Fitness value of each gene is evaluated using specific sigma truncation scaling scheme based on objective function (inequality coefficient as given in equation 7). If any fitness value reaches to desired results of population convergence then genetic algorithms is exited otherwise a new population is generated and crossover and mutation are applied until termination criteria is met.

Microsoft visual studio C++ 2008 was used for writing code of this algorithm by installing a special library of genetic algorithm named as Galib (Wall, n.d.) to utilize specific functions of genetic algorithm. In Galib, there are three kinds of genetic algorithms which are built in based on Genitor model, Goldberg's work and DeJong's method. Goldberg's work is based on simple GA while Genitor model and DeJong's method is based on steady state GA (SSGA). In a SSGA only one or two individuals are replaced in a population at each iteration. These new individuals become part of the population and are now available for selection. This is in contrast to the standard GA where the entire population (with the possible exception of an elite group of individuals carried over from the previous generation) is replaced each iteration (referred to as a generation in this case) (Parker & Parker, n.d.). So SSGA deals with overlapping populations while simple GA deals with non overlapping populations. The selection of type of GA for desired algorithm is done by analysing computation time of program with these algorithms. The computation time with different GA is given in table 4.

Table 4: Computation time based on different types of GA

Sr No.	Type of GA	Computation time in seconds
1	Simple GA based on Goldberg's work	3.65
2	SSGA based on Genitor method	1.30
3	SSGA based on DeJong's method	1.65

It has been seen that both SSGA (DeJong's method and Genitor model) have lower computation time as compared to simple GA based on Goldberg's work. Also other researches for optimization problems have proved that simple GA has poor performance as compared to steady state GA (SSGA) (Gordon & Whitely, n.d.). Two

SSGA methods though have more or less same computation time but it has been analysed by results obtained that SSGA based on Genitor method converges prematurely at a local optimum. They require large pool size and many trials to ensure the best solution is found (Parker & Parker, n.d.). So finally SSGA based on DeJong's method is selected for design of required optimization algorithm.

Genetic algorithm from GAlib named as "GASteadyStateGA" (Wall, 1996, p.32) is selected according to DeJong's method. It uses overlapping populations with a user-specifiable amount of overlap. The algorithm creates a population of individuals by cloning the genome or population that is passed to it. Each generation the algorithm creates a temporary population of individuals, adds these to the previous population, then removes the worst individuals in order to return the population to its original size.

Type of encoding selected is binary encoding. It is considered as natural for many problems (Obitko, 1998).This type of encoding can provide many no of chromosomes with small number of alleles and Binary GAs are preferred when the problem consists of discrete variables which is the case with renewable/low carbon energy optimization problems.

In Galib, GABin2DecGenome (Wall, 1996, p.54) is used for purpose of creating genome/initial population. This genome uses conventional methods of converting binary strings to decimal values and vice versa. This genome is selected because supply and demand profiles of renewable/low carbon energy are consisted of decimal values. A phenotype is made before instantiating this genome where phenotype defines how bits should map in decimal values and vice versa. Number of bits and minimum/maximum limit for initialization of genome are user defined and for purpose of this algorithm, program reads upper and lower bound itself for each supply and numbers of bits used are 16. This genome first encode input decimal value to binary values and after terminating GA, binary values are decoded to decimal values to display results.

Sigma truncation scaling scheme is used because it eliminates negative fitness scores during GA run which is most common problem in minimization problems where other scaling method fail in dealing with them. Sigma truncation scaling utilizes population mean and standard deviation to set negative results arbitrarily to zero. Objective scores are converted to fitness scores using the following relation (Wall, 1996, p.75).

$f = obj - (obj\_ave - c \bullet obj\_dev)$ -------------EQ 17

Roulette wheel selection method is used to select chromosomes to become parent for crossover. It is most commonly used for GAs and in this selection; wheel is spin N times to get N individuals. Better fitness chromosomes are given more chances to be selected.

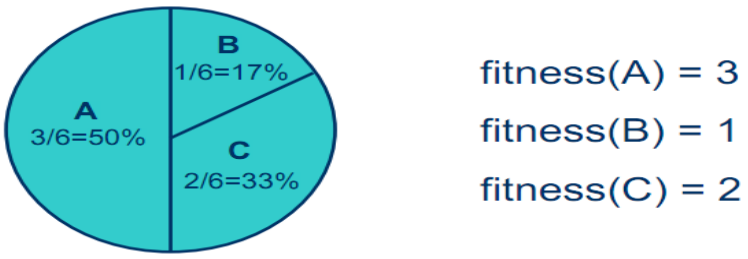


Figure 19: Roulette wheel selection

Genetic algorithm termination criteria settings vary from problem to problem. For this particular auto sizing problem of highly non linear and random nature, main objective of optimization is to reach towards global optimum. A simple analysis is done by comparing four different termination criteria i.e. number of generations, diversity, population convergence and population standard deviation. Termination of GA when specific numbers of generations are achieved has some disadvantages. It is difficult to know whether global minimum is achieved or not even when specified numbers of generations for termination are reached. Also since GA can run until many numbers of generations to search and reach towards global optimum in case of different data with different search spaces so one single general value of number of generation for stopping can't be defined for every type of data. A better usual way of termination is to stop when population diversity drops below a specified threshold. Again best threshold can be different for different data profiles considered (for supply and demand match) so it is difficult to design a single value. Termination can also be done by using the population's standard deviation as the stopping criterion. This type of termination is good when GA run time or cost control is of more importance than to reach towards global optimum so it is also not appropriate for auto sizing problem where there is a need of best global solution and user is willing to wait for finding best optimum match. Another good way to terminate is to find out that entire population has converged to good score and this can be accomplished by comparing the average score in the current population with the score of the best individual in the current population. However, the optimization algorithm is tested by using all above termination criteria in term of their computation time and results are given in table 5 showing that there is not a big difference in computation time for all of these termination criteria.

Table 5: Computation time of algorithm with different termination criteria

Termination criteria	Computation Time in seconds
Diversity	3.7
Generation	3.9
ScoreConvergence	3.6
Standard Deviation	3.8

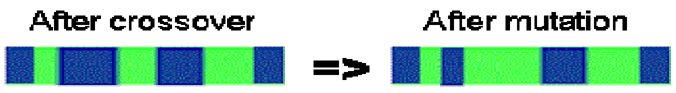
So the suitable stopping criteria selected for this particular optimization problem is the termination based on score convergence. The desired ratio for population convergence is selected as 0.99 which means to stop when population average is 99% of best.
The efficiency of GA is largely dependent on its parameters. For the choice of parameters, either standard parameters defined by different authors can be used or parameters can be customized according to specific problem. There are several recommended settings for these parameters but genetic parameters are selected according to DeJong's standard settings because type of genetic algorithm used is also based on DeJong's work. DeJong's settings (De Jong & Spears, 1990) are the de facto standard for most GAs and DeJong stated that his defined parameters (Pop size, no of generations, mutation rate, mutation type, crossover rate, crossover type) setting work very good for many GAs used for specifically function optimizations. Other parameters which are not available from DeJong's work are selected according to their most common use. The parameters of genetic algorithm which were used for defined optimization algorithm are given in table 6 with detail.

Table 6: Detail of selected genetic parameters for design of algorithm

Genetic Parameter name	Selected parameter values
Population size	50
Number of generations	1000
Encoding	Binary
Mutation rate	0.001

Mutation type	Flip bit
Crossover probability	0.6
Crossover type	Single point crossover
Elitism	Yes
No of generations for convergence test	50
Convergence %age	0.99
Selection method	Roulette wheel selection
Scaling scheme	Sigma truncation scaling
Termination criteria	Population diversity
Genetic Algorithm	Steady state GA
Genome	Decimal values

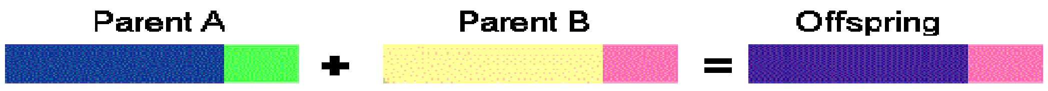
Parameters such as population size, number of generations, mutation rate, mutation type, crossover rate and crossover type are selected according to DeJong's defined standards. Very big population size does not improve efficiency of GA. Usually from many researches 50-100 is reported as best (Obitko, 1998). In this optimized algorithm, stopping criteria is not based on specific number of generations so number of generations will not have much effect on solution improvement. Mutation rate must be as low as possible otherwise it will alter the solution from originality. Cross over rate must be very high as compared to mutation rate and usually 0.6 is reported best. Flip bit mutation is used and is specific for binary operators in which value of chosen genes are simply inverted.



11001001 => 10001001

Figure 20: Flip bit mutation

Single point crossover is used which is very common for binary encoding. In single point crossover, everything after the crossover point is taken from other parent.



11001011+11011**111** = **11001111**

Figure 21: Single point cross over

Elitism is selected in order to eliminate the chances of loosing best chromosome. Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution (Obitko, 1998). Elite count is selected as 2.

4.2.3. <u>Details of objective functions used:</u>

Literature review on match evaluation criteria for renewable energy has shown that inequality coefficient (IC) has proved to be best match evaluation function so far and it must be as low as possible with minimum value of 0. Lower is inequality means larger is the match between supply and demand. However values of IC between 0-0.4 shows good match between supply and demand. It must be stated that importance of correlation coefficient (CC) in renewable energy match evaluation cannot be ignored and it must be as large as possible with maximum value of 1. Flow chart shown in figure 16 describes genetic algorithm with objective of minimizing IC only. CC is also involved in designed optimization algorithm but not as an objective function of genetic algorithm but as a second filter on the results obtained from genetic algorithms with objective of IC. For providing reasons for this method, an analysis is done with different objective functions for optimization along with their merits and demerits. For purpose of this analysis, a simple example is used with three number of supplies which means that n=3 i.e. design variables are n1, n2 and n3. Analysis includes different cases as given below.

<u>Genetic algorithm using IC only (Single objective):</u>

An optimization algorithm was written in C++ language using single objective genetic algorithm with IC, as objective function to be minimized. Five time steps data was assumed for supplies and demand as given below in equations 18 to21.

$D = \{10.0, 20.0, 30.0, 40.0, 50.0\}$ -------------EQ 18

$S1 = \{2.0, 5.0, 4.0, 6.0, 7.0\}$ -------------EQ 19

$S2 = \{3.0, 7.0, 3.5, 4.5, 5.0\}$ -------------EQ 20

$S3 = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ -------------EQ 21

Results from genetic algorithm produced an optimum capacity (n1, n2, n3) of unit supplies (S1, S2, S3) at (1, 1, 7) which show that 1 unit of supply S1 and S2 each along with 7 units of S3 will be optimum combination for satisfying demand D. By using these optimum capacities of supplies, inequality coefficient comes out to be 0.048, correlation coefficient is 0.984 and resultant supply (RS) is evaluated at each time step which comes out to be

$RS = \{12.0, 26.0, 29.0, 39.2, 48.0\}$ -------------EQ 22

IC value is less than 0.3 and there is very low difference at each time step between demand D and resultant supply RS in terms of magnitude as well which shows a good match between demand and resultant supply. CC value is also very high which tells about good correlation between resultant supply and demand. Hence IC has proved to be a good objective function for match evaluation purposes of renewable/ low carbon energy profiles.

Genetic algorithm using CC only (Single objective):

When same program is run with CC only, as objective function to be maximized then genetic algorithm produced optimum capacity (n1, n2, n3) of unit supplies (S1, S2, S3) at (1, 1, 98) which describes 1 unit of S1 and S2 each along with 98 units of S3. By using this optimum capacities correlation coefficients come out to be 0.99985, value of IC is 0.8194 and resultant supply at each time step is

$RS = \{104.0, 209.0, 302.0, 403.0, 503.0\}$ -------------EQ 23

So by maximizing CC as objective function, although good value of correlation coefficient is achieved but value of inequality is very high and magnitude of resultant supply (RS) at each time step is showing large difference with demand D depicting very bad match. It can be seen that correlation coefficient deals with trend matching but magnitude match is not guaranteed. Good matches must deal with matching of magnitude as well as trend both. Hence, it can be stated that CC alone will not prove as good match evaluation criteria for renewable/low carbon energy systems

Genetic algorithm using IC and CC both (Multi objective genetic algorithm):

In order to involve both IC and CC in an optimization algorithm for match evaluation, multi objective genetic algorithm is analysed by considering two objective functions; IC and CC. (IC to be minimized and CC to be maximized).

This method is analysed by using same data as given in equations 18 to21. The results obtained for optimal capacities of S1, S2 and S3 from multi objective optimization algorithm i.e. Pareto front is recorded in the table 7 which show 16 optimal solutions on Pareto front i.e. 16 optimum capacity combinations for 3 supplies along with values of IC and CC for them. If this Pareto front is plotted for IC and CC then a curve is obtained shown in figure 22.

Table 7: Pareto front/optimal solutions obtained from multi objective optimization

S.No	f1(IC)	f2(CC)	S1	S2	S3
1	0.088278	0.98465	1.013309	1	7.15625
2	0.741079	0.99919	1.013309	1	35.79447
3	0.680111	0.99874	1.013309	1	28.42777
4	0.429439	0.99563	1.013309	1	14.65315
5	0.549349	0.99739	1.013309	1	19.34065
6	0.460176	0.99611	1.013309	1.003906	15.65315
7	0.89221	0.99987	1.013309	1	89.95592
8	0.239589	0.99161	1.013309	1	10.1875
9	0.60388	0.99802	1.013309	1	22.40315
10	0.759733	0.99931	1.013309	1	38.79447
11	0.162758	0.98934	1.013309	1	8.875
12	0.175685	0.98951	1.013309	1.015625	9.0625
13	0.89221	0.99987	1.013309	1	89.95592
14	0.876895	0.99982	1.013309	1.015625	78.39342
15	0.088278	0.98465	1.013309	1	7.15625
16	0.088287	0.98474	1.013309	1.000977	7.1875

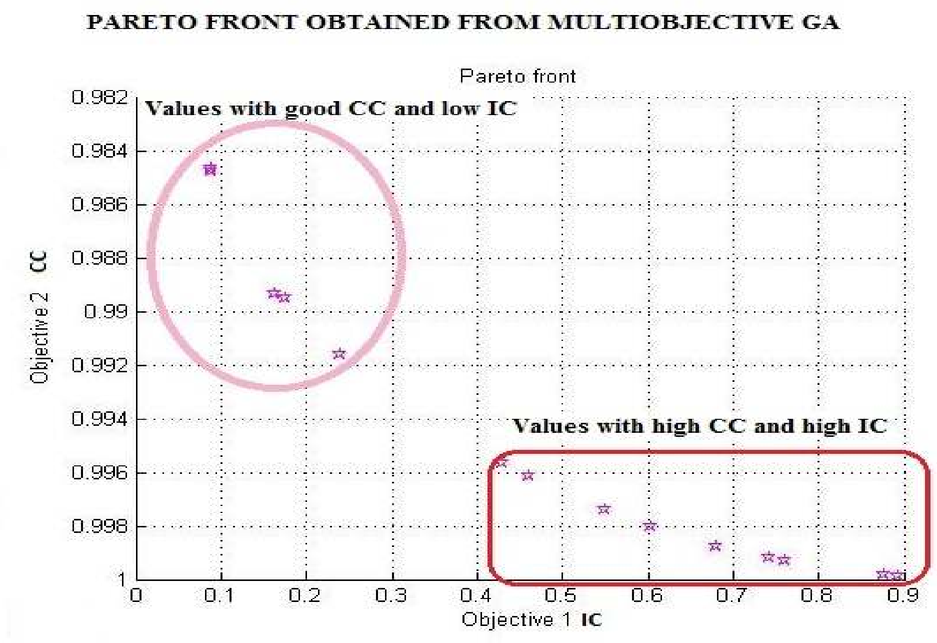


Figure 22: Pareto front obtained in multi objective genetic algorithm

It is clear from figure 22 that the Pareto front includes those points for which IC value is reaching to 0.9. IC must be as low as possible for being first preference in match evaluation and its values more than 0.4 are not acceptable for providing good match. The optimal solutions with high correlation coefficient reaching to 1 enclosed in red line are not associated with low values of IC. On the other hand, optimal solutions enclosed in pink circle are dealing with low IC values i.e. less than 0.3 and still have good correlation coefficients. Results show that multi objective genetic algorithm is providing equal importance to both of its objective function and there is a need of another filter which would remove solutions with high IC values. So it can be stated that multi objective optimization using IC and CC both as its objectives has not proved to be good for development of auto sizing algorithm because correlation coefficients can't be given equal or more importance when compared with IC in match evaluation.

4.2.4. <u>Discussion on developed algorithm:</u>

As above analysis show that IC must be used as main objective for match evaluation but CC cannot be ignored completely in defining optimization algorithm, so there must a method which can involve both objectives i.e. IC and CC but IC must be given first and main preference. Optimization algorithm shown in flow chart of figure 16 is designed for particular auto sizing problem in which main objective function of genetic algorithm is IC (Inequality Coefficient) that is to be minimized and then CC acts as a second filter on results which are already satisfying criteria of minimum IC. This method is designed to avoid problems with single objective genetic algorithm (using IC only as objective function and CC only as objective function) and multi objective genetic algorithms (using IC and CC both as objective functions together).
This method is also based on some other important facts which must be considered in finding global minima. First of all sometimes, there are some functions which have their extrema (maxima/minima) occurring at more than one point. For example, the graph of cosine function i.e. f(x) =Cos(x) as shown in figure 23 has its extrema (absolute and relative) that occurs at many points. Maximum value of 1 is at $x = \ldots -4\pi, -2\pi, 0, 2\pi, 4\pi, \ldots$ and minimum value of -1 at $x = \ldots -3\pi, -\pi, \pi, 3\pi, \ldots$.

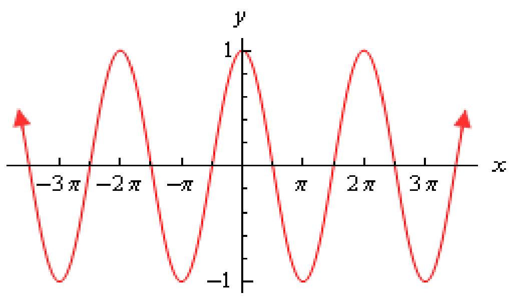


Figure 23: Graph of cosine function with more than one extrema

In case of renewable/low carbon energy systems that deal with highly non linear supplies and demand profiles, there is also a possibility that there a number of same absolute/global minima or maxima which in turn can result in more than one optimized solution which means that good match criteria could be satisfied by different optimized combinations of capacities. Also considering nature of genetic algorithm which selects random number from its search space every time when it is run, it can be possible to get any of these different optimized results on different runs of GA.

Another fact related to genetic algorithm is that it can converge sometimes on local minima but its convergence at local minima can be avoided by careful selection of genetic parameters. So although rare but still there are chances of getting such results. Consider an example in which a very simple data is assumed for three supplies and a demand for five time steps as given below in equations 24 to 27.

$D = \{11.0, 11.0, 11.0, 11.0, 11.0\}$ -------------EQ 24

$S1 = \{1.0, 1.0, 1.0, 1.0, 1.0\}$ -------------EQ 25

$S2 = \{2.0, 2.0, 2.0, 2.0, 2.0\}$ -------------EQ 26

$S3 = \{4.0, 4.0, 4.0, 4.0, 4.0\}$ -------------EQ 27

When this data is provided to genetic algorithm with objective of minimizing inequality coefficient then it is analysed that genetic algorithm provided different results in different runs of program though many of these results were same. The program is run for 35 times and 35 results are recorded given in table 9. The results for which IC value is minimum of all i.e. 0 are (n1, n2, n3) =(3, 2, 1), (5, 1, 1), (1, 3, 1) and (1, 1, 2) which shows that global minima occurs at 0 for four different optimized capacity solutions which is the case of multiple absolute/global extrema with same extreme value just like cosine function. All other results are dealing with

IC values which are very close to 0 but not 0 which shows that genetic algorithm converged early before reaching to global optimum. Now two questions arise here; first is how to filter all results to find global minimum and second is how to find single best result in case when multiple global extrema occur. For answer of first question, in order to find a global minimum, Program has to search for result which gives minimum value of IC. But as far as second question is concerned, there will be a need of another evaluation or filter criteria where correlation coefficient can help. Considering these reasons, two further methods can be analysed for making of renewable energy auto sizing algorithm.

1. One method can be designed which first look for global minima from all results and if situation of multiple global minima arises then one result with maximum correlation coefficient can be selected as best optimized one. However, in renewable/ low carbon energy problem which deals with highly stochastic, random and non linear profiles, chances of multiple global minima are very rare. So when there is only one global minimum then after searching it, this method is left with nothing to involve correlation coefficient.

2. Second possible method can be to filter all results on basis of correlation coefficient (to find one with maximum CC) without trying to find global minima. It is clear that all results are obtained with objective of minimizing IC meaning that every results will try to reach towards minimum value of IC i.e. 0 and values of IC between 0-0.4 can provide good supply and demand match. So even if global minimum is not found, results are already dealing with very low value of IC which is pretty acceptable. This method seems to be better than first method because it always involve correlation coefficient in its working.

Thus, finally designed optimization method as shown in flow chart of figure 16 utilizes single objective genetic algorithm with minimization of IC as objective function thus fulfilling main criteria for match evaluation. GA is called n number of times to get several results and one result with maximum correlation coefficient is selected as best optimized one to display.

The question is still there for how many times GA must be called to look for maximum correlation coefficient. By analysing some actual profiles of renewable energy cases in designed algorithm, it is noted that 10-20 runs of GA are enough for the method. No of GA runs must not be too small e.g. 1-5 to avoid losing chance of analysing every possibility and must not be too large e.g. 100-1000 to avoid slow speed and high computation time of overall algorithm working. However by analysing some actual profiles of renewable energy cases in designed algorithm, it is also noted that most of the time, optimum solutions are coming out to be same for all number of GA runs describing single global/absolute extrema and chances of different solution in different runs of GA are very rare. This designed optimization algorithm method has following advantages.

- It is using minimization of IC as main objective function which is most importantly desirable.
- It is not completely ignoring CC.
- Although it is involving CC but is not providing equal importance to CC as that of IC thus eliminating problems arising in multi objective optimization described above.
- It is calling GA for n number of times with different random numbers and checks for all possible solutions which can fulfil criteria of minimum IC from which only one solution i.e. single best optimized capacity is obtained at the end of algorithm.

Chapter 5

5. Verification and Case Study for auto sizing Algorithm:

5.1. Verification of developed algorithm:

For the purpose of verification of optimization algorithm made, Matlab software global optimization tool box with genetic algorithm solver as described in section 2.3.3 is used. Optimization algorithm is run with help of C++ program written in Microsoft Visual Studio 2008 to obtain results which are then compared with results obtained from the Matlab global optimization toolbox genetic algorithm solver. Programming codes generated for C++ language with Microsoft Visual Studio 2008 as well as for Matlab Global Optimization Toolbox genetic algorithm solver are given in appendix A and appendix D respectively. In both software (Matlab and C++), same genetic algorithm parameters (as explained in table 6) are selected and IC is set as the objective function to be minimized. For simplification purposes, single demand and three supplies are assumed to be consisted of 5 time steps as given in equations 18-21. GA from both software is run for 35 times and the results as given in table 8 which came out to be similar for all 35 runs representing absolute minimum at this point. This verifies correct working of optimization algorithm. Graphs representing behaviour of GA i.e. fitness value at each number of generation and best individual of three variables S1, S2 and S3) are plotted with help of Matlab shown in figure 24.

Table 8: Result of 35 GA runs using supply and demand data from equations 18-21

No	Optimized Algorithm made	Matlab genetic algorithm
Same result for 35 runs	n1,n2,n3=1,1,7.18	n1,n2,n3=1,1,7.2

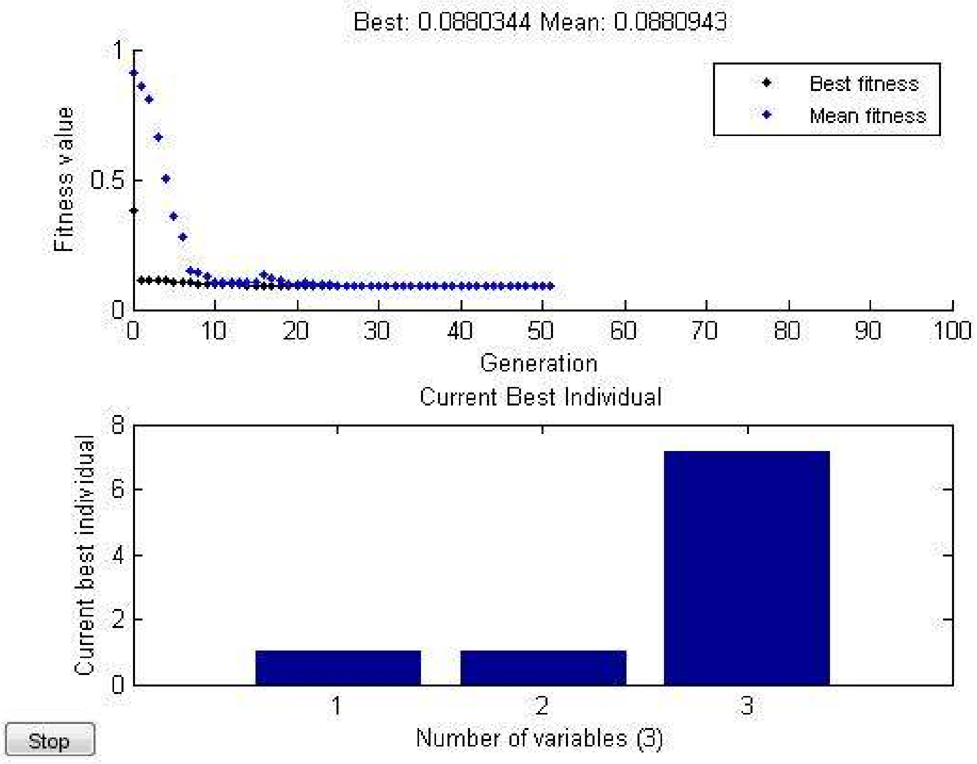


Figure 24: Result of GA behaviour from Matlab

A second example is considered by assuming data of demand and supplies as given in equations 24 to 27. Results are recorded in form of table 9 for 35 runs of genetic algorithm using optimization algorithm made in C++ and in Matlab. It is visible that optimized combinations are coming out to be similar from both verifying correct working of generated optimization algorithm code in C++. Results show that optimized solutions are repeated many times but specific order of appearance for different results in different runs is unpredictable. From all 35 runs, minimum value of IC i.e. 0 is coming out for similar combinations which are (3, 2, 1), (5, 1, 1), (1, 3, 1) and (1, 1, 2) of three supplies S1, S2 and S3 representing absolute minima at theses points. It is also noted that all possible results have been appeared in 20-25 runs and all are associated with very low value of IC i.e. very close to 0.

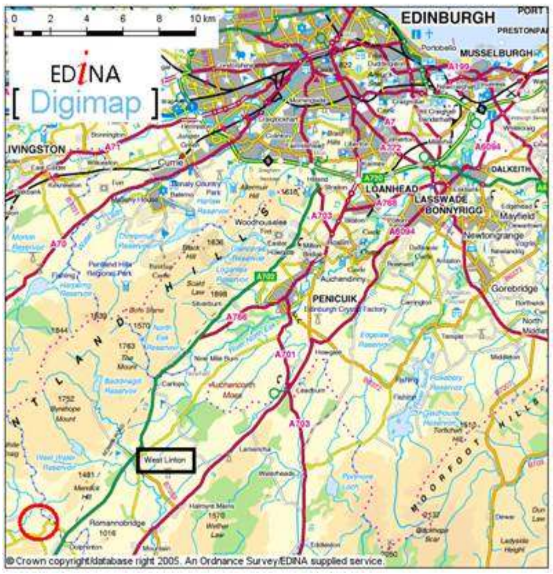
Table 9: Result of 35 GA runs using supply and demand data from equations 24-27

Serial	Optimization Algorithm made in C++				Matlab Software			
No	n1	n2	n3	IC	n1	n2	n3	IC
1	2	2	1	0.048	1	1	2	0.000
2	3	2	1	0.000	1	1	2	0.000
3	2	2	1	0.048	1	1	2	0.000

4	2	2	2	0.120	2	1	2	0.043
5	5	1	1	0.000	3	2	1	0.000
6	3	1	1	0.100	2	2	1	0.048
7	2	2	1	0.048	2	1	2	0.043
8	4	1	1	0.048	1	2	1	0.100
9	2	2	1	0.048	1	1	2	0.000
10	1	2	1	0.100	1	1	2	0.000
11	3	1	1	0.100	1	1	2	0.000
12	2	2	1	0.048	3	1	1	0.100
13	4	1	1	0.048	1	3	1	0.000
14	4	1	1	0.048	1	3	1	0.000
15	1	2	2	0.083	3	2	1	0.000
16	3	1	1	0.100	2	1	1	0.158
17	3	2	1	0.000	1	2	1	0.100
18	3	1	1	0.100	1	2	2	0.083
19	2	2	1	0.048	4	1	1	0.048
20	2	2	2	0.120	1	3	1	0.000
21	1	3	1	0.000	1	1	2	0.000
22	2	1	2	0.043	2	2	1	0.048
23	2	1	2	0.043	1	2	1	0.100
24	2	1	2	0.043	1	3	1	0.000
25	1	2	2	0.083	2	2	1	0.048
26	2	1	2	0.043	1	1	2	0.000
27	3	1	2	0.083	5	1	1	0.000
28	2	2	1	0.048	2	2	1	0.048
29	4	2	1	0.043	1	1	2	0.000
30	2	3	1	0.043	3	2	1	0.000
31	3	1	2	0.083	2	3	1	0.043
32	1	3	1	0.000	1	1	2	0.000
33	4	1	1	0.048	3	2	1	0.000
34	2	2	1	0.048	2	2	1	0.048
35	2	2	2	0.120	1	1	2	0.000

5.2. Case Study done with Merit:

After designing and verification of optimization algorithm, a real case study is done by running optimization using actual data of supplies and demands database which varies with climatic conditions. The case considered is located in Garvaled House Estate, five miles from West Linton in the Scottish Borders. Some area surrounding Garvald house is subjected for a construction project of sustainable building or Eco barn for which electricity is assumed to be supplied by off grid Wind/Solar hybrid system. Developed optimization algorithm is utilized to find optimum capacity of wind turbines and solar panels.



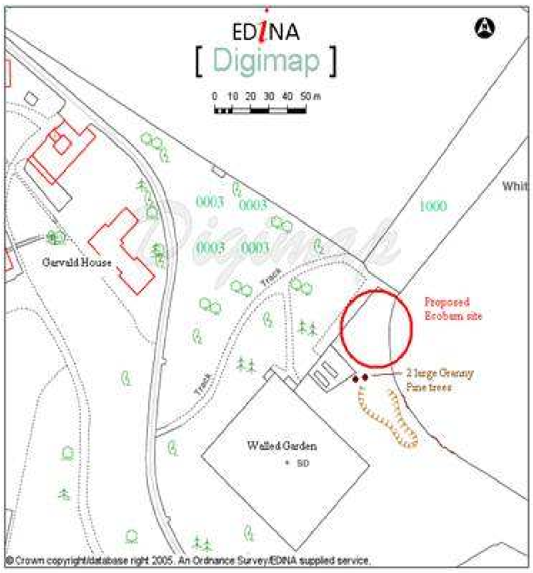

EDINA
Digimap
Garvald House
Walled Garden
Proposed Ecobarn site
2 large Granny Pine trees
Track
©Crown copyright/database right 2005. An Ordnance Survey/EDINA supplied service.

Figure 26: The location of Garvald House, to the South-West of West Linton.

Figure 25: Proposed Ecobarn site

The database of demand and supplies over different periods (over months of four different seasons and over whole year) for climatic conditions of Glasgow are exported from Merit Tool. Hourly energy demand profile named 'office_A_electrical' in the demand database is scaled down to the predefined peak load (around 0.44kW) for the Ecobarn building for use in case study. Supply options include 600W wind turbine manufactured by Proven, 100W poly-crystalline PV panels manufactured by Siemens (tilt angle @40deg, orientation facing south).

Discussion and Analysis:

The optimum capacity of two supplies to fulfil Ecobarn demand is found by using designed optimization algorithm. One month hourly data (30 days case: 720 time steps and 31 days case: 744 time steps) for four different seasons (winter-January, Spring-April, Summer-July and Autumn-October) and yearly hourly data (8760 time steps) for supplies and demand is used for optimization. Results of IC values obtained for resulting optimum capacities of two different supplies are displayed in form of a graph as shown in figure 27 .Optimized capacities of supplies for all months and year are coming out with very low value inequality coefficients i.e. less than 0.4 depicting good match. So results with low inequality coefficient which was main objective of developed algorithm represent fine working of optimization algorithm for different time spans (months or years).

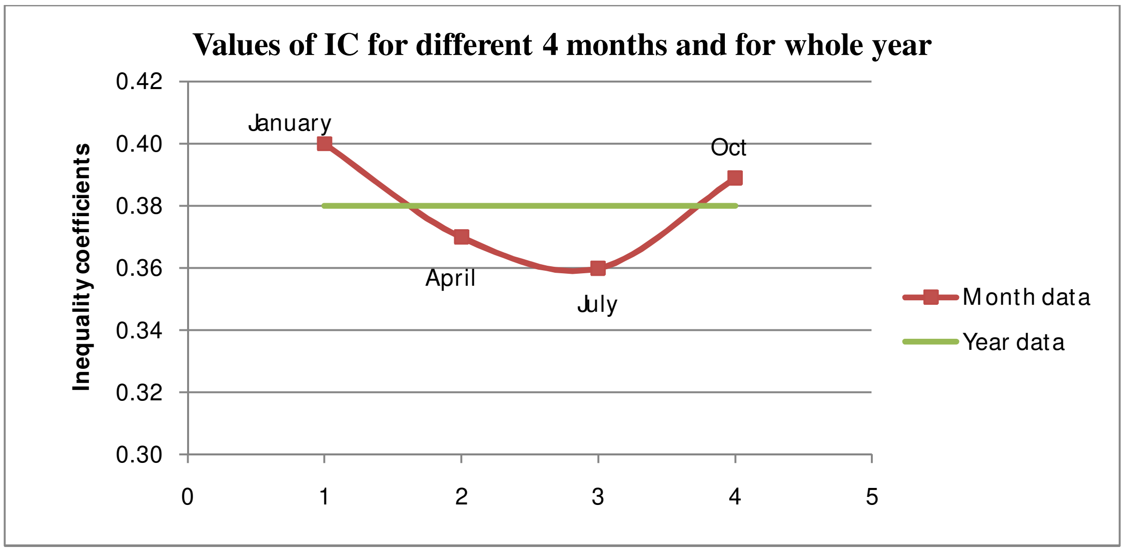


Figure 27: IC values for optimized capacities measured using week/month/year data

Correlation coefficients must be as close to 1 as possible for perfect match. The designed optimization algorithm is mainly based on minimization of IC, but still resulting optimum capacities for all months and year have positive values of correlation coefficients as shown in figure 28 providing information of positive correlation of resultant optimum supply with demand.

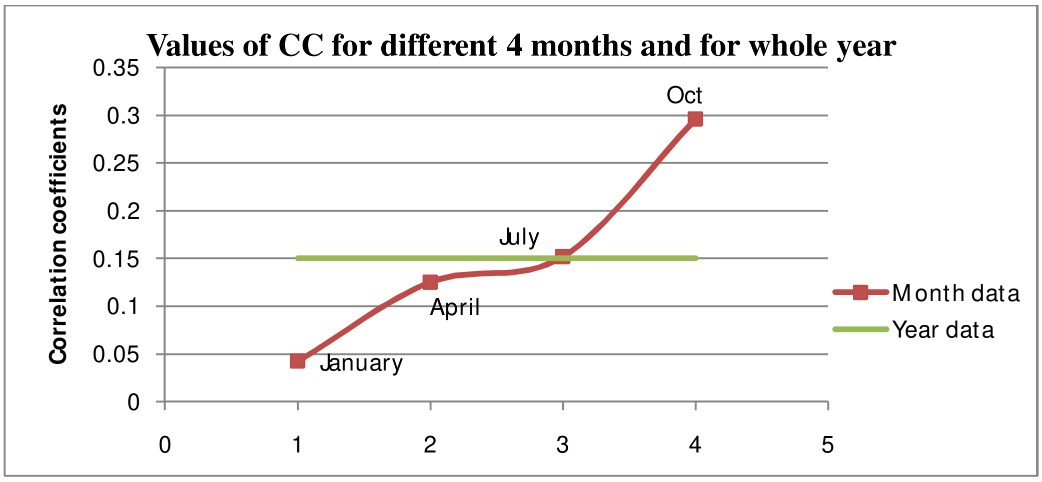


Figure 28: CC values for optimized capacities measured using week/month/year data

Optimized capacities results which are obtained for all months and year from optimization algorithm are used in Merit tool against above described demand profile to analyse results produced by match in merit. Values of IC and CC for these optimized capacities from Merit tool are coming out to be exactly same as obtained from designed optimization algorithm shown in figure 27 and 28 depicting proper working of optimization algorithm results in Merit. Match rates for all optimized capacities are close to 60 % or more than 60 % resulting in reasonable match. Results for match rates are given in table 10.

Table 10: Match rate for optimized capacities

Climate Site Dundee 1980		Optimized capacity (Wind T/b, PV)	Match rate	Match type
Months data	Jan	(1, 15)	59.53	5/10
	Apr	(1, 1)	63.16	6/10
	July	(1,8)	63.8	6/10
	Oct	(1, 10)	61.37	6/10
Yearly data	--	(1, 4)	61.61	6/10

Merit tool itself is utilizing inequality coefficient as base for finding match rates. Since these optimized capacities are also mainly found with the objective function of minimizing IC, so they are providing good match rates when they are checked in Merit tool. Also as C++ is integrated development environment of Merit tool so it can be stated that designed optimization algorithm can be embedded successfully with Merit.

Optimized capacity is given as (Wind t/b, PV) with some number, for example in case of yearly data, optimized capacity is (1, 4) which means that 1 unit of 600W Wind turbines and 4 panels of 100W PV (Total requirement is 600 Watts capacity of wind turbines and 400 Watts capacity of PV panels) are required to satisfy demand over whole year.

By considering summer case of July only form table 9, there are 8 PV panels appearing with 1 wind t/b as optimum capacity and number of PV panels are 15 when a winter month of January is considered. This might be because of the fact that in summer, there are more solar direct and diffuse radiations as compared to winter so in winter more PV panels are required to satisfy demand while as wind speed remains equally variable in summer and winter so number of wind turbines required for both months are same. This represents logical results obtained from designed optimization algorithm. Graph of solar direct and diffuse radiation along with wind speed is shown in figure 29 and 30.

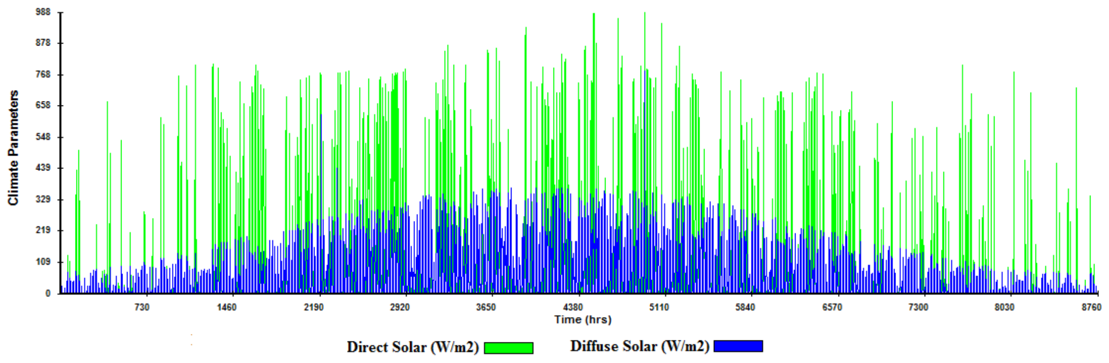


Figure 29: Variation of solar radiations over whole year

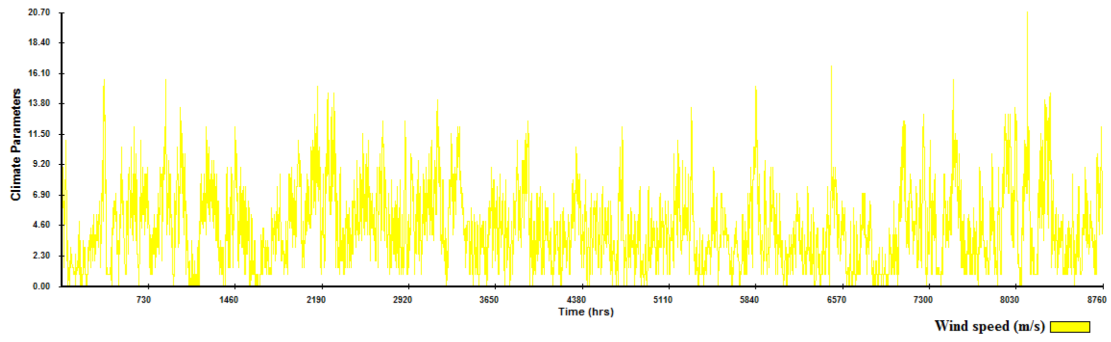


Figure 30: Variation of wind speed over whole year

In this case study 20 numbers of runs of genetic algorithm are set to obtain 20 resulting solutions and their CC and IC values are computed. However, when months of January and April are considered then all 20 results produced from 20 runs of genetic algorithm were same as provided in table 9. When month of July is considered then optimized algorithm results for 20 runs of GA are listed in table 11.

Table 11: Result for 20 runs of genetic algorithm for month of July

No of GA run	Optimum capacity obtained (Wind t/b, PV)	IC	CC	No of GA run	Optimum capacity obtained (Wind t/b, PV)	IC	CC
1	1,8	0.3622	0.152	11	1,8	0.3622	0.152
2	1,7	0.3628	0.138	12	1,7	0.3628	0.138
3	1,8	0.3622	0.152	13	1,8	0.3622	0.152

4	1,8	0.3622	0.152	14	1,8	0.3622	0.152
5	1,8	0.3622	0.152	15	1,7	0.3628	0.138
6	1,7	0.3628	0.138	16	1,7	0.3628	0.138
7	1,8	0.3622	0.152	17	1,7	0.3628	0.138
8	1,7	0.3628	0.138	18	1,8	0.3622	0.152
9	1,8	0.3622	0.152	19	1,8	0.3622	0.152
10	1,8	0.3622	0.152	20	1,7	0.3628	0.138

From table it is clear that sometimes GA produced result with optimizes capacity of 1 t/b, 8 PV and sometimes with 1 t/b, 7 PV. Both of these solutions are resulted by using minimization of IC as objective. This is the situation when CC value comes in action then from these two solutions one is selected as better one with maximum CC value which is optimum capacity of 1 t/b, 8 PV. This show that designed algorithm works nicely in case when GA random nature produce different results however chances of such cases are very less.

Optimum capacity obtained from designed algorithm is 1 wind turbine and 4 PV panels over period of whole year as given in table 10. In order to analyse validity of optimum capacity generated with help of designed algorithm, a simple analysis is done for this case study using Merit by recording results of different number of PV panels with 1 wind turbine to satisfy demand over whole year. Values of IC and CC are displayed in form of a graph in figure 31 with 16 supply combinations (1 wind t/b and number of PV panels varying from 1 to 16). The values of IC show that minimum IC value which is 0.38 is appeared when 4 PV panels are used with wind turbine. Graph of CC values represent that correlation coefficients increase with increase number of PV panels and maximum CC is when 16 PV panels are used with wind turbine but this capacity is associated with high inequality which was one reason of not using CC as main objective function.

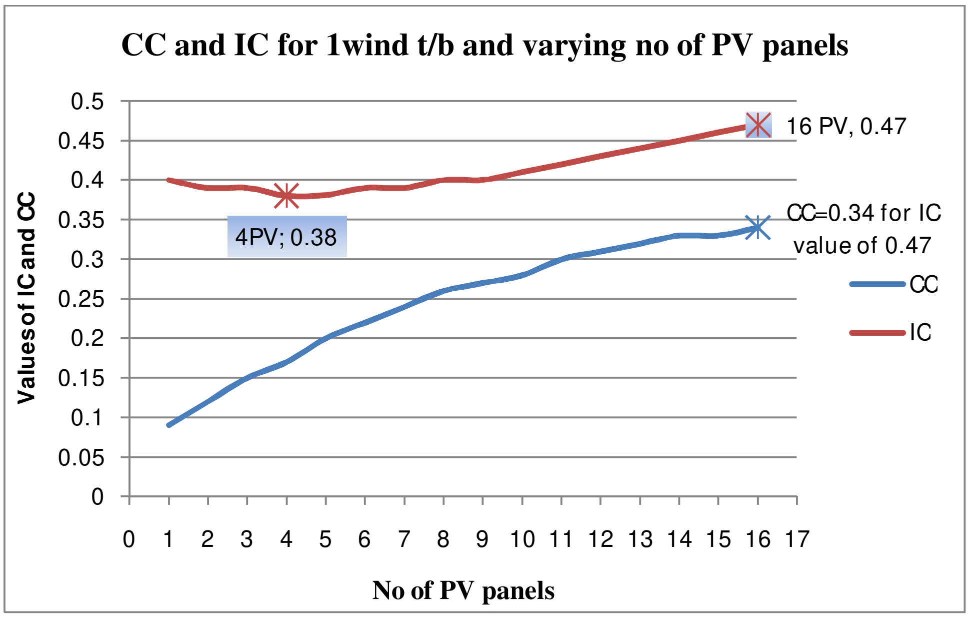


Figure 31: Values of IC and CC for 16 supply combination using 1wind t/b and varying number of PV panel from 1-16.

Match percentage for all these optimum combinations (1 wind t/b and varying number of PV panels from 1 to16) are displayed in form of graph in figure 32 and % age match is maximum for this optimum combination of 1 wind t/b and 4 PV panels.

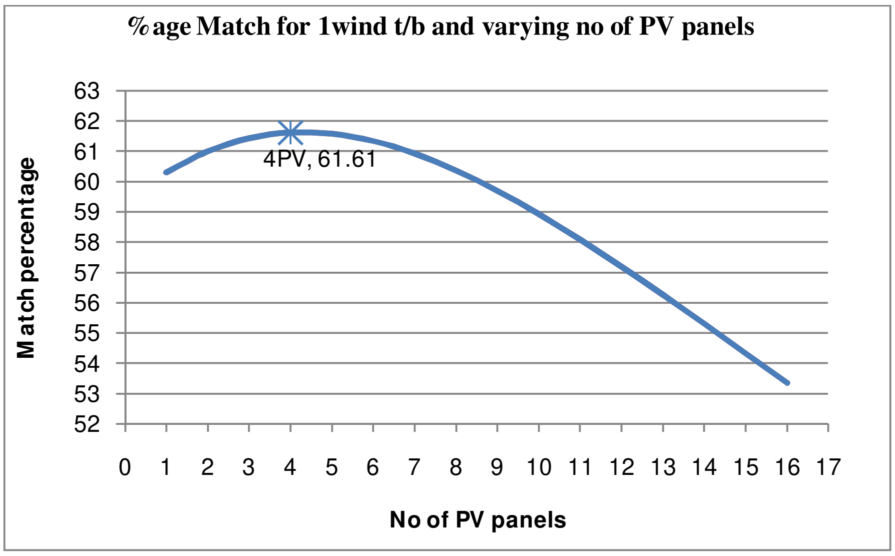


Figure 32: Values of Match %age for 16 supply combination using 1wind t/b and varying number of PV panel from 1-16.

Graph of surplus and deficit energy is displayed in figure 33 for all combination of these two supplies (1 wind t/b and varying number of PV panels from 1 to16). The graph shows that curve of energy surplus and energy deficit cut each other at a point with supply of 4 PV panels with 1 wind turbine representing an optimum of energy deficit and energy surplus. This optimum point i.e. optimum capacity is same as generated by developed optimization algorithm.

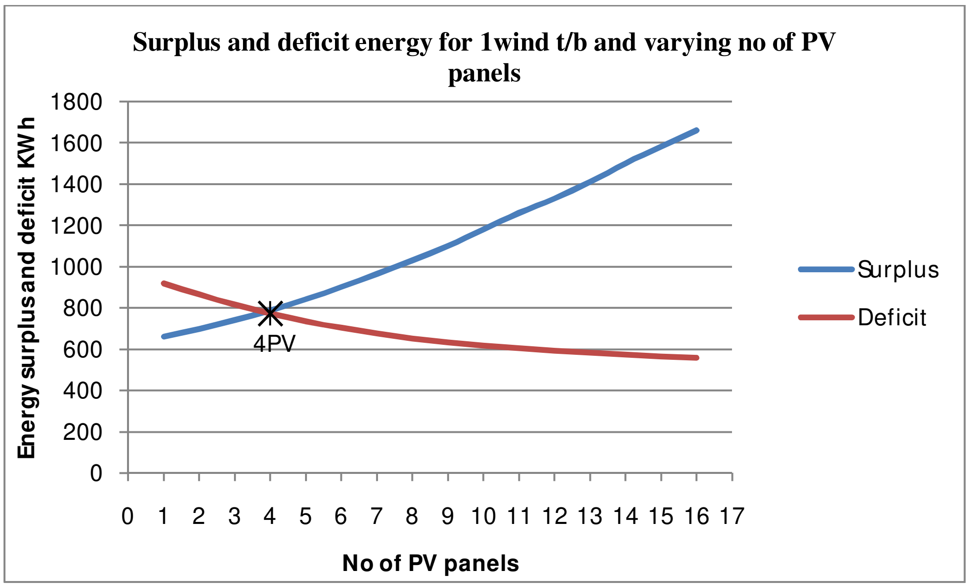


Figure 33: Values of surplus and deficit energy for 16 combination using 1wind t/b and varying number of PV panel from 1-16.

Surplus and deficit energy can be managed and match %age can be increased further by using auxiliaries. No of auxiliaries are not optimized in this designed algorithm. However after finding optimum capacity from algorithm, analysis of different number of auxiliaries with optimum capacity can be done to achieve better results. The residual graphs with optimum capacity (1 wind t/b and 4 PV panels) for Ecobarn obtained from algorithm considering period of whole year are analysed by adding 4, 8 and 16 batteries (battery reserve sizes with the capacity of 215Ah at 12V). Residual graphs are shown in figure 34 to 36 and represents that residual is decreased to great extents when number of batteries is increased. Residual is very less when 16 number of batteries are used.

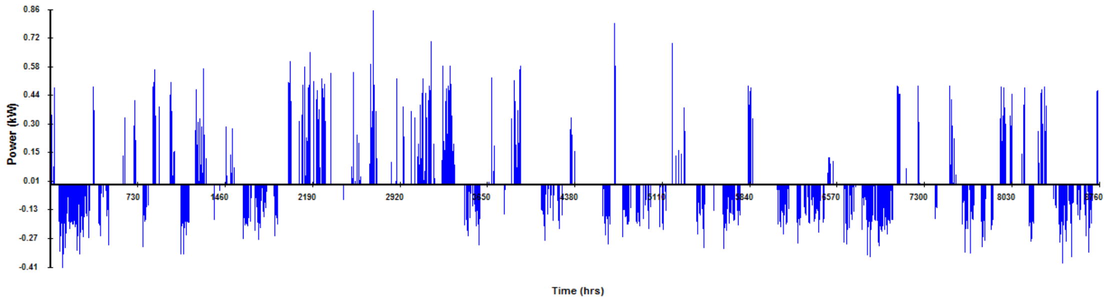


Figure 34: Graph of residual with optimum capacity for whole year with 4 batteries

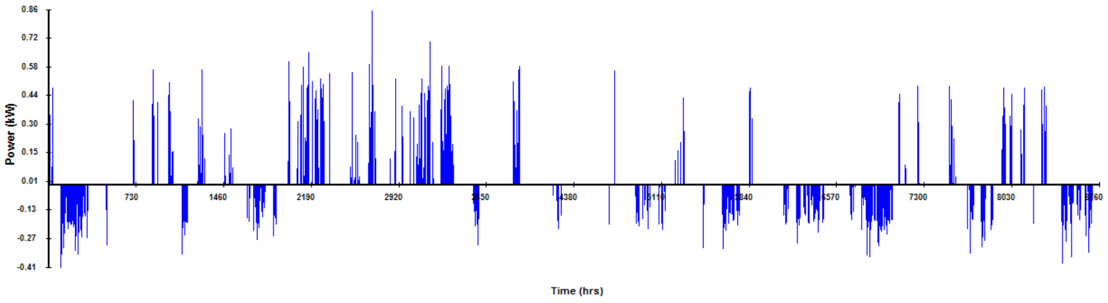


Figure 35: Graph of residual with optimum capacity for whole year with 8 batteries

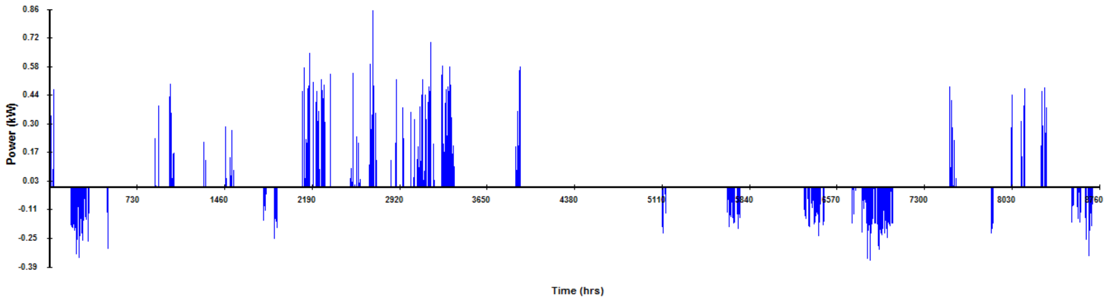


Figure 36: Graph of residual with optimum capacity for whole year with 16 batteries

Results proved that generated optimum capacity by designed algorithm is good for many criteria including match percentage, deficit and surplus energy, inequality coefficient and up to some extent correlation coefficient etc. Algorithm is avoiding exhaustive search of analysing different capacity combinations which become more ridiculous when there are more than 2 renewable energy supply options and help in reducing time for finding optimum match.

As results can be generated by considering different periods of time (weeks, months or year), so there can be different optimum capacities for different periods of time because supply from renewable energy is variable with seasons and climatic conditions. Designed algorithm can provide benefit for better energy utilization for hybrid energy systems. Different supplies could be turn off and on in different seasons according to their optimum requirement to get the best match and better energy saving. If fixed numbers of supplies are to be installed to satisfy some demand over whole year then optimum capacity obtained by considering whole year data can be installed. There is a need a further analysis in order to know which optimum capacity is best suited for a particular demand depending on further requirements of demand which may include cost, economics, payback time, connection to grid, reliability and accommodation of any future increases in energy demand. However, this can be stated that optimum capacities generated from designed algorithm can prove to be very useful base for any further analysis. Hence developed algorithm can provide good benefits to renewable energy analysts.

Chapter 6

6. Conclusion and further work:

6.1. Conclusion:

Current concerns related to energy security and climate change leads toward a lot of research and development in renewable energy. Hybrid energy systems can address the limitations of cost, reliability, efficiency and emission on individual renewable energy supply options for its better utilization. Design of hybrid energy systems need correct selection and sizing of renewable energy systems to reduce variability in supply and demand. MERIT provides a suitable platform for auto sizing a single renewable energy supply source which can be extended to any n number of renewable energy supplies to match with a single demand.

In this thesis, Optimisation Algorithm for Auto-sizing Capacity of Renewable and Low Carbon Energy Systems is developed utilizing principles of genetic algorithm. Genetic algorithm is selected based on its match with nature of renewable energy auto sizing problem. Also existing literature of current research in sizing problems of hybrid energy has shown an increasing trend towards evolutionary algorithms (genetic algorithms). Genetic algorithms are very good in finding global minima but sometimes they converge early at local optimum. However careful selection of genetic parameters can overcome this problem. Selection of genetic algorithm and genetic parameters for designed algorithm is based on DeJong's work on genetic algorithms and comparison of computation time.

Literature for renewable energy supply/demand match evaluation criteria has provided two most useful match evaluation criterias which are inequality coefficient(IC) and correlation coefficient (CC). For getting good match between supply and demand IC must be as low as possible. Analysis is done with single and multi objective optimization methods using these two coefficients as objective function.

Single objective optimization method using IC only provided good match results but ignored correlation coefficient completely. On the other hand, single objective optimization method using CC only showed that it deals with trend matching only between supply and demand and magnitude match can't be guaranteed. Multi objective optimization using both coefficients (IC to be minimized and CC to be maximized) deals with results which were dealing with very high value of IC because multi objective optimization provide equal weightage to all of its objectives. Some results with high CC may not deal with low IC values. So single objective optimization (using IC or CC) as well as multi objective optimization (using IC and CC both) have not proved to be suitable for designing auto sizing algorithm for renewable/low carbon energy problems. IC has proved to be more importantly desirable as compared to CC and must be main objective of optimization.

Thus, the designed algorithm is based on single objective genetic algorithm with objective of minimizing inequality coefficient only and then GA is called several times to achieve several results from which one solution is selected with maximum correlation coefficient. Work also represented that 10-20 times GA must be called to avoid chances of missing global solution and for reasonable computation time of program. This design algorithm is overcoming problems of early convergence of GA and of situation of getting more than one solution in case when energy profile deals with multiple absolute extrema. Designed optimization method will work for maximizing match b/w supply and demand and utilizes hourly data of supply and demand profiles as its input.

The developed algorithm is successfully verified by Matlab genetic algorithm solver. A case study done has shown designed algorithm works nicely for different time spans considered (weeks, months or years) with low values of resulting inequality coefficients and positive correlation coefficient values. It can provide optimum capacity of as many numbers of supplies as required to match with a single demand so it can handle large scale design problems. Case study also showed that results of optimum capacities obtained from designed algorithm when used in Merit to counter check them then they provided exactly same values of IC and CC which are obtained from designed algorithm. Also optimum capacity achieved from developed algorithm is providing good match rates and deals with good balance of surplus and deficit energy when checked with Merit. As the code of developed algorithm is made in C++ which is integrated development environment of Merit and results are same when counterchecked with Merit so developed algorithm can be embedded successfully with Merit thus will overhaul Merit capabilities. Developed algorithm is useful for better energy utilization, good for eliminating exhaustive search and work in reasonable computation time. It can also provide useful basis for analysts in further requirements e.g. economics, payback time, connection to grid, reliability and accommodation of any future increases in energy demand.

6.2. <u>Further work:</u>

The further work may include

- The developed optimization algorithm for auto sizing capacity of renewable/low carbon energy deals with searching optimum capacity combination of different supply options available to match with one single demand. However there is an area of further improvement in which there is more than one demand options are to be matched with different supply options and one has to find which supply optimum capacity is best matched with which demand option.

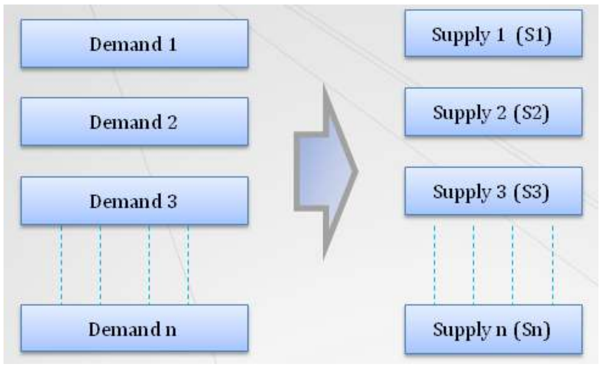


Figure 37: More demand options with more supplies options

- Auxiliaries are of great importance in any renewable energy design problem however in this developed algorithms, number of auxiliaries are not optimized. So in order to improve this algorithm, there could be an addition of auxiliaries' capacity optimization.

- The developed optimization algorithm search for an optimum point on the basis of maximizing match rate and completely ignores the economics and cost. So another possible improvement can be to insert cost as a constraint in the working of genetic algorithm.

- Also performance of genetic algorithms can be improved by using the concepts of hybrid genetic algorithms, global elitism and dynamic adaptability of crossover probabilities, mutation probabilities and other genetic parameters. So the performance of developed algorithm can be further analysed by involving different types of improvement in genetic algorithm.

- The optimization algorithm is based on maximizing electricity match between supply and demand. However work can be extended to match 'heat' or 'heat and supply both' for supplies and demand with a CHP.

- Finally after finding good results of developed algorithm from case study done in Merit, further work also includes its embedment with merit.

Appendix:

A. Program code generated for Matlab

Matlab global optimization toolbox.

- Code written in Matlab for example 2 section ____.

```
function y = project_fitness(x,d,a,b,c)
d = [11.0,11.0,11.0,11.0,11.0];
a = [1.0,1.0,1.0,1.0,1.0];
b = [2.0,2.0,2.0,2.0,2.0];
c = [4.0,4.0,4.0,4.0,4.0];
y=((dot((d-(x(1)*a+x(2)*b+x(3)*c)),(d-
(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5/(((dot(d,d))/5)^0.5+((dot((d-
(x(1)*a+x(2)*b+x(3)*c)),(d-(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5);
```

- Code written in Matlab for example 1 section

```
function y = project_fitness(x,d,a,b,c)
d = [10.0,20.0,30.0,40.0,50.0];
a = [2.0,5.0,4.0,6.0,7.0];
b = [3.0,7.0,3.5,4.5,5.0];
c = [1.0,2.0,3.0,4.0,5.0];
y=((dot((d-(x(1)*a+x(2)*b+x(3)*c)),(d-
(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5/(((dot(d,d))/5)^0.5+((dot((d-
(x(1)*a+x(2)*b+x(3)*c)),(d-(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5);
```

- Code written in Matlab for multiobjective optimization

```
function y =simple_multiobjective(x,d,a,b,c)
d = [10.0,20.0,30.0,40.0,50.0];
a = [2.0,5.0,4.0,6.0,7.0];
b = [3.0,7.0,3.5,4.5,5.0];
c = [1.0,2.0,3.0,4.0,5.0];
y(1)=((dot((d-(x(1)*a+x(2)*b+x(3)*c)),(d-
(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5/(((dot(d,d))/5)^0.5+((dot((d-
(x(1)*a+x(2)*b+x(3)*c)),(d-(x(1)*a+x(2)*b+x(3)*c))))/5)^0.5);
y(2)=-(dot((d-((sum(d))/5)),((x(1)*a+x(2)*b+x(3)*c)-
((sum(x(1)*a+x(2)*b+x(3)*c))/5))))/(dot((d-((sum(d))/5)),(d-
((sum(d))/5)))*dot(((x(1)*a+x(2)*b+x(3)*c)-
((sum(x(1)*a+x(2)*b+x(3)*c))/5)),((x(1)*a+x(2)*b+x(3)*c)-
((sum(x(1)*a+x(2)*b+x(3)*c))/5))))^0.5;
```

B. List of available GA Software Packages

Sr No.	Name of GA Package	Type
1	EO Evolutionary Computation Framework by Geneura Team	A C++ genetic algorithm library
2	GAlib by Matthew Wall	A C++ genetic algorithm library
3	GAGS by J. J. Merelo	A C++ genetic algorithm library
4	GAJIT - A Simple Java Genetic Algorithms by Matthew Faupel	A java genetic algorithm library
5	GA Playground by Ariel Dolan	A java genetic algorithm library
6	PGAPack Parallel Genetic Algorithm Library by David Levine	A Ansi C genetic algorithm library
7	GAUL by Stewart Adcock	A Ansi C genetic algorithm library
8	Sugal 2.1 Genetic Algorithms Simulator by Andrew Hunter	A Ansi C genetic algorithm library
9	GENOCOP III By Zbigniew Michalewicz	Genetic Algorithm for constrained problems in C
10	DE by Rainer storn	Differential Evolution Genetic Algorithm in C and Matlab
11	PGAPack from Argonne National Laboratory	Parallel genetic algorithm in Fortran and C
12	PIKAIA by Charbonneau, Knapp an d Miller	Genetic Algorithm in Fortran 77/90
13	GAGA by Ian Poole	Genetic algorithm for general application in C
14	GAS by Jelasity and Dombi	Genetic Algorithm in C++
15	Genetic algorithm in Matlab by Michael B. Gordy	N/A
16	GADS from Mathworks	Genetic Algorithm and Direct search Toolbox in Matlab
17	GEATbx by Hartmut Pohlheim	Genetic and Evolutionary algorithm for Matlab
18	GAOT by Jeffrey Joines	Genetic Algorithms Optimization toolbox in Matlab
19	Genetic Algorithm for global optimization by Mathworks	Global Optimization Toolbox in Matlab

C. Results for case study with 1 WT and different numbers of PV panels from Merit

No of PV panels	%age Match	Surplus energy	Deficit Energy	CC	IC
1PV	60.3	661.55	918.55	0.09	0.4
2PV	60.99	699.76	864.71	0.12	0.39
3PV	61.42	741.94	815.84	0.15	0.39
4PV	61.61	789.31	772.43	0.17	0.38
5PV	61.57	842.97	734.33	0.2	0.381
6PV	61.33	901.24	702.75	0.22	0.39
7PV	60.92	965.2	675.26	0.24	0.39
8PV	60.36	1030	651.96	0.26	0.4
9PV	59.69	1100	632.74	0.27	0.4
10PV	58.92	1180	616.55	0.28	0.41
11PV	58.08	1260	603.38	0.3	0.42
12PV	57.19	1330	592.12	0.31	0.43
13PV	56.26	1410	581.58	0.32	0.44
14PV	55.3	1500	572.5	0.33	0.45
15PV	54.33	1580	564.46	0.33	0.46
16PV	53.36	1660	557.4	0.34	0.47

D. **Example of code generated in C++ for designed algorithm (with week data)**

```
/*---------------------------------------------------------------------
------
This is the program finding the number of supplies of particular
capacity
using genetic algorithm with the objective function of minimizing
Inequality
coefficient. Further it take 100 results for the optimum combinations
by
running Genetic algorithm 100 times for minimum Inequality and then
look for
the one optimum combination which is having maximum correlation
coefficient.
---------------------------------------------------------------------
-------*/

#include <stdio.h>
#include <iostream>
#include <fstream>
#include <ga/ga.h>
#include <math.h>
#define cout STD_COUT
using namespace std;
void   geneticalg(unsigned  int   &,int   &,int   &,double   &,double
&);//seed,sn1,sn2,Ic,cc
float objective(GAGenome &);  //Declation of objective function

int main(int argc, char **argv)
{

        int Totalsupplies=2;                  // Showing the number of
provided supplies
      double IC;
        double CC;
        int Sn1;
        int Sn2;

        unsigned int seed = 0;
        float ICarray[100];
        float CCarray[100];
        float n1array[100];
        float n2array[100];
        for(int i=0;i<5;i++)// Number of runs of ga are selected here
          {

          for(int i=1; i<argc; i++) // Code for selecting random seed
          {
          if(strcmp(argv[i++],"seed") == 0)
            seed = atoi(argv[i]);
          }
              geneticalg(seed,Sn1,Sn2,IC,CC);//Calling  galib  genetic
algorithm
```

```
              //Filling the arrays
            n1array[i]=Sn1;
            n2array[i]=Sn2;
            ICarray[i]=IC;
            CCarray[i]=CC;
              cout<<"the       n1       n2       and       IC       are
"<<Sn1<<","<<Sn2<<","<<IC<<".\n";
          }
        //Finding maximum element position in  CC array
         int maxIndex =0;
           int size=100;
         for (int j=1; j<size; j++)
             {
           if (CCarray[j] > CCarray[maxIndex])
               maxIndex = j;
           }
         cout<<" The optimized combination of 2 supplies PV and wind
is ("<<n1array[maxIndex]<<" and "<<n2array[maxIndex]<<").";
           cout<<"\n For which IC is "<<ICarray[maxIndex]<<"and CC is
"<<CCarray[maxIndex];

         return 0;
}

void geneticalg(unsigned int &seed,int &n1,int &n2,double &IC,double
&CC)
{
      int S=2;
      int n=192;
    /*Supply and demand detas*/
double
supply2[]={0.115,0.213,0.213,0.145,0.145,0.115,0.213,0.366,0.366,0.54
8,0.548,0.548,0.666,0.682,0.666,0.548,0.366,0.548,0.548,0.424,0.25,0.
213,0.145,0.175,0.145,0.025,0.087,0.06,0.06,0.087,0.115,0.175,0.25,0.
213,0.175,0.25,0.213,0.115,0.115,0.213,0.115,0.145,0.145,0.087,0.043,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.025,0.025,0.025,0.06,0.087,0.175,0.06
,0.06,0.025,0.06,0.043,0.115,0.115,0.115,0.06,0.025,0.087,0.115,0.087
,0.06,0.043,0.087,0.175,0.213,0.25,0.303,0.424,0.548,0.482,0.424,0.30
3,0.213,0.06,0.025,0,0.025,0,0,0,0,0,0,0.087,0,0.213,0.303,0.25,0.145
,0.145,0.25,0.613,0.548,0.482,0.366,0.115,0.06,0.087,0.06,0,0.115,0.1
15,0.06,0.06,0.087,0.115,0.087,0.175,0.213,0.424,0.366,0.548,0.672,0.
672,0.482,0.482,0.482,0.482,0.613,0.613,0.482,0.613,0.175,0.115,0.213
,0.25,0.175,0.213,0.175,0.06,0.06,0.087,0,0.043,0.025,0.087,0.175,0.2
13,0.303,0.25,0.303,0.366,0.482,0.213,0.213,0.115,0.115,0.175,0.175,0
.06,0,0.025,0.043,0,0,0,0,0.087,0.175,0.175,0.25,0.303,0.25,0.303,0.1
75,0.213,0.175,0.303,0.145,0.115,0.043,0.087,0.06};
double
supply1[]={0,0,0,0,0.001,0.002,0.008,0.016,0.018,0.027,0.042,0.045,0.
046,0.044,0.044,0.034,0.035,0.027,0.018,0.019,0.012,0,0,0,0,0,0,0,0.0
06,0.012,0.023,0.029,0.034,0.034,0.035,0.038,0.038,0.035,0.029,0.012,
0.009,0.009,0.006,0.004,0.002,0,0,0,0,0,0,0,0.002,0.007,0.017,0.022,0
.024,0.023,0.022,0.023,0.027,0.025,0.027,0.019,0.008,0.008,0.013,0.01
```

```
4,0.003,0,0,0,0,0,0,0,0.002,0.006,0.011,0.009,0.01,0.013,0.014,0.026,
0.04,0.031,0.036,0.035,0.029,0.013,0.01,0.01,0.004,0,0,0,0,0,0,0,0.00
1,0.001,0.004,0.01,0.008,0.005,0.006,0.006,0.01,0.007,0.015,0.031,0.0
16,0.007,0.006,0.003,0.011,0,0,0,0,0,0,0,0.002,0.003,0.007,0.005,0.00
3,0.007,0.017,0.023,0.018,0.022,0.023,0.013,0.02,0.027,0.012,0.01,0.0
04,0,0,0,0,0,0,0,0,0.001,0.003,0.004,0.006,0.004,0.006,0.007,0.008,0.
008,0.007,0.008,0.003,0.005,0.009,0.006,0.003,0,0,0,0,0,0,0,0.002,0.0
04,0.006,0.009,0.009,0.007,0.01,0.016,0.031,0.036,0.036,0.032,0.029,0
.024,0.019,0.007,0.002,0,0,0};
double
Demand[]={14.065,13.96,13.615,14.44,13.295,14.135,12.93,13.795,13.835
,14.145,13.985,14.615,14.675,14.435,14.86,13.685,13.91,13.96,13.795,1
3.89,13.895,14.125,14.145,14.205,14.815,13.955,13.91,13.6,13.335,13.1
7,13.3,14.095,13.87,14.025,14.935,14.555,13.935,13.52,14.115,14.345,1
4.05,13.775,13.3,13.36,14.39,13.875,14.355,14.28,14.8,14.995,14.73,14
.515,14.82,17.74,23.815,26.24,27.66,29.98,28.23,28.745,28.8,25.99,25.
995,22.175,21.45,16.64,16.045,15.535,14.315,15.5,15.235,15.765,14.58,
14.88,14.985,14.96,16.275,18.235,24.74,27.14,27.415,28.495,28.49,27.6
15,29.02,27.185,26.53,23.415,19.655,17.375,16.3,15.545,15.955,15.43,1
6.885,14.42,15.055,14.49,14.735,14.99,15.45,18.09,24.035,25.99,27.02,
29.335,30.14,27.555,26.85,27.305,25.695,23.51,20.91,17.15,16.825,15.2
35,14.65,15.59,14.68,15.54,14.82,15.685,14.9,14.545,16.07,19.215,22.6
45,26.48,27.125,29.35,27.88,28.745,28.12,27.3,26.69,23.96,20.585,17.4
75,17.545,15.625,15.175,15.32,14.915,15.38,15.13,14.9,14.515,14.465,1
5.45,18.88,21.82,26.42,28.325,28.9,28.2,28.335,27.755,26.68,25.545,23
.675,21.465,17.28,16.405,15.705,15.595,15.29,15.365,14.335,13.9,13.23
5,13.87,13.29,14.135,13.3,13.2,13.4,13.95,14.24,13.795,14.315,14.645,
14.235,14.425,14.66,14.525,13.705,13.54,13.54,14.37,14.725,14.82,14.4
4};

         /*----Declare variables for the GA parameters----*/

       int popsize  = 50;
       int ngen     = 1000;
       float pmut   = 0.01;
       float pcross = 0.6;
       float pconv  = 0.99;          //  threshhold  for  when  we  have
converged
       int nconv    = 50;            //  how  many  generations  back  to
look


      /*Create a phenotype for two variables.  The number of bits you
can use to
      represent any number is limited by the type of computer you are
using.  In
      this case, we use 16 bits to represent a floating point number
whose value
      can range from 1 to 100, inclusive.  The bounds on supplies can
be applied
      here and/or in the objective function.*/

      GABin2DecPhenotype map;
      for (int i=0;i<S;i++)
      map.add(16,1,100);


     GABin2DecGenome genome(map, objective);                  //Create
the  template  genome  using  the  phenotype  map  we  just  made  and
objective function.

     /*--create the GA using the genome and run it.--*/

     GASteadyStateGA ga(genome);
       GASigmaTruncationScaling                                scaling;
//Reference to page 78 of documentation of galib
     ga.minimize();                                          //Code to
show that objective function is to be minimized
     ga.populationSize(popsize);
     ga.nGenerations(ngen);
     ga.pMutation(pmut);
     ga.pCrossover(pcross);
     ga.scaling(scaling);
     ga.scoreFilename("bog.dat");
     ga.scoreFrequency(10);                                //Reference
to page 25-27 of documentation of galib
     ga.flushFrequency(50);
     ga.evolve(seed);


     /*-------Saving Results as integer-----------*/

     genome = ga.statistics().bestIndividual();
       n1=static_cast<int>(genome.phenotype(0)+0.5);
//Converting decimal points to closest Integers
       n2=static_cast<int>(genome.phenotype(1)+0.5);

     /*----Finding  Inequality coefficient-------*/
     double x=0;
     double y=0;
     double z=0;
       double aa;
     double bb;
     double cc;
     for(int  j=0;  j<n;  j++)   //summation of supplies and difference
at all n time steps
       {
         double a;
       double b;
       a=supply1[j]*n1+supply2[j]*n2;
       b=Demand[j]-(supply1[j]*n1+supply2[j]*n2);
       x += a*a;
         y += b*b;
       }
     for (int i=0;i<n ;i++)                                //Summation
of demand at all n time steps
```

```
       {
         z += Demand[i]*Demand[i];
       }
     aa=y/n;
     bb=z/n;
     cc=x/n;
     IC=sqrt(aa)/(sqrt(bb)+sqrt(cc));

     /*----Finding  correlation coefficient-------*/
       double xCC=0;
       double yCC=0;
       double zCC=0;
       double meandemand=0;
       double meansupply=0;
       for(int i=0;i<n;i++)
        meandemand=Demand[i]+meandemand;
        meandemand=meandemand/n;
       for(int j=0;j<n;j++)

meansupply=supply1[j]*genome.phenotype(0)+supply2[j]*genome.phenotype
(1)+meansupply;
        meansupply=meansupply/n;
     for(int j=0; j<n; j++)                          //summation
of products(Dt-d)(St-s)
        {
        xCC                          +=                       (Demand[j]-
meandemand)*(supply1[j]*genome.phenotype(0)+supply2[j]*genome.phenoty
pe(1)-meansupply);
        }
     for (int i=0;i<n ;i++)                          //Summation
of demand at all n time steps
        {
        double f;
        double t;

f=supply1[i]*genome.phenotype(0)+supply2[i]*genome.phenotype(1)-
meansupply;
        t=(Demand[i]-meandemand);
      yCC += (f*f);
        zCC += (t*t);
      }
       double aCC;
       double bCC;
       aCC=zCC*yCC;
       bCC=sqrt(aCC);
       CC=(xCC/bCC);
}

float objective(GAGenome & c)
{
    GABin2DecGenome & genome = (GABin2DecGenome &)c;
      int S=2;
      int n=192;
    /*Supply and demand detas*/
double
supply2[]={0.115,0.213,0.213,0.145,0.145,0.115,0.213,0.366,0.366,0.54
8,0.548,0.548,0.666,0.682,0.666,0.548,0.366,0.548,0.548,0.424,0.25,0.
213,0.145,0.175,0.145,0.025,0.087,0.06,0.06,0.087,0.115,0.175,0.25,0.
213,0.175,0.25,0.213,0.115,0.115,0.213,0.115,0.145,0.145,0.087,0.043,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.025,0.025,0.025,0.06,0.087,0.175,0.06
,0.06,0.025,0.06,0.043,0.115,0.115,0.115,0.06,0.025,0.087,0.115,0.087
,0.06,0.043,0.087,0.175,0.213,0.25,0.303,0.424,0.548,0.482,0.424,0.30
3,0.213,0.06,0.025,0,0.025,0,0,0,0,0,0,0.087,0,0.213,0.303,0.25,0.145
,0.145,0.25,0.613,0.548,0.482,0.366,0.115,0.06,0.087,0.06,0,0.115,0.1
15,0.06,0.06,0.087,0.115,0.087,0.175,0.213,0.424,0.366,0.548,0.672,0.
672,0.482,0.482,0.482,0.482,0.613,0.613,0.482,0.613,0.175,0.115,0.213
,0.25,0.175,0.213,0.175,0.06,0.06,0.087,0,0.043,0.025,0.087,0.175,0.2
13,0.303,0.25,0.303,0.366,0.482,0.213,0.213,0.115,0.115,0.175,0.175,0
.06,0,0.025,0.043,0,0,0,0,0.087,0.175,0.175,0.25,0.303,0.25,0.303,0.1
75,0.213,0.175,0.303,0.145,0.115,0.043,0.087,0.06};
double
supply1[]={0,0,0,0,0.001,0.002,0.008,0.016,0.018,0.027,0.042,0.045,0.
046,0.044,0.044,0.034,0.035,0.027,0.018,0.019,0.012,0,0,0,0,0,0,0,0.0
06,0.012,0.023,0.029,0.034,0.034,0.035,0.038,0.038,0.035,0.029,0.012,
0.009,0.009,0.006,0.004,0.002,0,0,0,0,0,0,0,0.002,0.007,0.017,0.022,0
.024,0.023,0.022,0.023,0.027,0.025,0.027,0.019,0.008,0.008,0.013,0.01
4,0.003,0,0,0,0,0,0,0,0.002,0.006,0.011,0.009,0.01,0.013,0.014,0.026,
0.04,0.031,0.036,0.035,0.029,0.013,0.01,0.01,0.004,0,0,0,0,0,0,0,0.00
1,0.001,0.004,0.01,0.008,0.005,0.006,0.006,0.01,0.007,0.015,0.031,0.0
16,0.007,0.006,0.003,0.011,0,0,0,0,0,0,0,0.002,0.003,0.007,0.005,0.00
3,0.007,0.017,0.023,0.018,0.022,0.023,0.013,0.02,0.027,0.012,0.01,0.0
04,0,0,0,0,0,0,0,0,0.001,0.003,0.004,0.006,0.004,0.006,0.007,0.008,0.
008,0.007,0.008,0.003,0.005,0.009,0.006,0.003,0,0,0,0,0,0,0,0.002,0.0
04,0.006,0.009,0.009,0.007,0.01,0.016,0.031,0.036,0.036,0.032,0.029,0
.024,0.019,0.007,0.002,0,0,0};
double
Demand[]={14.065,13.96,13.615,14.44,13.295,14.135,12.93,13.795,13.835
,14.145,13.985,14.615,14.675,14.435,14.86,13.685,13.91,13.96,13.795,1
3.89,13.895,14.125,14.145,14.205,14.815,13.955,13.91,13.6,13.335,13.1
7,13.3,14.095,13.87,14.025,14.935,14.555,13.935,13.52,14.115,14.345,1
4.05,13.775,13.3,13.36,14.39,13.875,14.355,14.28,14.8,14.995,14.73,14
.515,14.82,17.74,23.815,26.24,27.66,29.98,28.23,28.745,28.8,25.99,25.
995,22.175,21.45,16.64,16.045,15.535,14.315,15.5,15.235,15.765,14.58,
14.88,14.985,14.96,16.275,18.235,24.74,27.14,27.415,28.495,28.49,27.6
15,29.02,27.185,26.53,23.415,19.655,17.375,16.3,15.545,15.955,15.43,1
6.885,14.42,15.055,14.49,14.735,14.99,15.45,18.09,24.035,25.99,27.02,
29.335,30.14,27.555,26.85,27.305,25.695,23.51,20.91,17.15,16.825,15.2
35,14.65,15.59,14.68,15.54,14.82,15.685,14.9,14.545,16.07,19.215,22.6
45,26.48,27.125,29.35,27.88,28.745,28.12,27.3,26.69,23.96,20.585,17.4
75,17.545,15.625,15.175,15.32,14.915,15.38,15.13,14.9,14.515,14.465,1
5.45,18.88,21.82,26.42,28.325,28.9,28.2,28.335,27.755,26.68,25.545,23
.675,21.465,17.28,16.405,15.705,15.595,15.29,15.365,14.335,13.9,13.23
5,13.87,13.29,14.135,13.3,13.2,13.4,13.95,14.24,13.795,14.315,14.645,
14.235,14.425,14.66,14.525,13.705,13.54,13.54,14.37,14.725,14.82,14.4
4};
```

```
       double x=0;
     double y=0;
     double z=0;
       double IC=0;
     double aa;
     double bb;
     double cc;

     for(int j=0; j<n; j++)                              //summation
of squares of supplies and difference at all n time steps
       {
         double a;
       double b;

a=supply1[j]*genome.phenotype(0)+supply2[j]*genome.phenotype(1);
       b=Demand[j]-
(supply1[j]*genome.phenotype(0)+supply2[j]*genome.phenotype(1));
       x += a*a;
         y += b*b;
       }
     for (int i=0;i<n ;i++)                              //Summation
of square of demand at all n time steps
       {
             z += Demand[i]*Demand[i];
       }

     aa=y/n;
     bb=z/n;
     cc=x/n;
     IC=sqrt(aa)/(sqrt(bb)+sqrt(cc));
     return(IC);
}
// stop when pop average is 95% of best
const float desiredRatio = 0.95;
GABoolean
GATerminateUponScoreConvergence(GAGeneticAlgorithm & ga){
if(ga.statistics().current(GAStatistics::Mean) /
ga.statistics().current(GAStatistics::Maximum) > desiredRatio)
return gaTrue;
else
return gaFalse;
}
```

References:

Anagnostopoulos, J.S. & Papantonis, D.E., 2007. Optimal sizing of a run-of-river small hydropower plant. *Energy Conversion and Management*, 48(10), pp.2663-70.

Anagnostopoulos, J.S. & Papantonis, D.E., 2008. Simulation and size optimization of a pumped–storage power plant for the recovery of wind-farms rejected energy. *Renewable Energy*, 33(7), p.1685–1694.

Antoniou, A. & Lu, W., 2007. *Practical optimization: algorithms and engineering applications*. New York, United States of America: Springer Science+Business Media,LLC.

Bagul, A.D., Salameh, Z.M. & Borowy, B., 1996. Sizing of a stand-alone hybrid wind-photovoltaic system using a three-event probability density approximation. *Solar Energy*, 56(4), pp.323-35.

Banos, R. et al., 2011. Optimization methods applied to renewable and sustainable energy: A review. *Renewable and Sustainable Energy Reviews*, 15, pp.1753-66.

Bilal, B.O. et al., 2010. Optimal design of a hybrid solarewind–battery system using the minimization of the annualized cost system and the minimization of the loss of power supply probability (LPSP). *Renewable Energy*, 35(10), pp.2388-90.

Born, F.J., 2001. *Aiding renewable energy integration through complimentary demand supply matching*. PhD Thesis. Glasgow: Univeristy of Strathclyde.

Celik, A.N., 2003. Techno-Economic Analysis of Autonomous PV-Wind Hybrid Energy Systems Using Different Sizing Methods. *Energy Conversion and Management*, 44, pp.1951-68.

Collette, Y. & Siarry, P., 2004. *Multiobjective optimization: principles and case studies*. 1st ed. New York: Springer-Verlag.

De Jong, A.K. & Spears, M.W., 1990. An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. In *First Workshop Parallel Problem Solving from Nature*., 1990. University of Dortmund.

Deb, K., 2005. *Optimization for engineering design: algorithms and examples*. New Delhi: Prentice Hall of India Private limited.

Fahimuddin, A.K.M., 2003. *PLATON Optimisation Using GALib*. Project work. Brunswick, Germany: Technical University Braunschweig.

Forrest, S., 1993. Genetic algorithms: principles of natural selection applied to computation. *Science*, 261, pp.872-78.

Gen, M. & Cheng, R., 1997. *Genetic algorithms and engineering design*. Wiley-IEEE.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA, United States of America: Addison-Wesley Longman Publishing Co., Inc.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co.

Gordon, V.S. & Whitely, D., n.d. *Series and parallel genetic algorithms as function optimizer*. Paper. Fort Collins: Colorado State University.

Gray, P. et al., 1997. *Evolutionary Algorithms:Genetic Algorithms, Evolutionary Programming and Genetic Programming*. [Online] Sandia National Laboratories Available at: http://www.cs.sandia.gov/opt/survey/ea.html [Accessed 6 August 2011].

Hakimi, S.M. & Moghaddas-Tafreshi, S.M., 2009. Optimal sizing of a stand-alone hybrid power system via particle swarm optimization for Kahnouj area in south-east of Iran. *Renewable Energy*, 34(7), pp.1855-62.

Haupt, R.L. & Haupt, S.E., 2004. *Practical Genetic Algorithm*. 2nd ed. New Jersey: A John Wiley & Sons, Inc., Publication.

Haupt, R.L. & Haupt, S.E., 2004. *Practical genetic algorithms*. 2nd ed. A John Wiley and Sons Inc.

Jebaraj, S. & Iniyan, S., 2006. A riview of energy models. *Renewable and Sustainable Energy Reviews*, 10(4), pp.281-311.

Jones, K.O., 2005. COMPARISON OF GENETIC ALGORITHM AND PARTICLE SWARM OPTIMIZATION. In *International Conference on Computer Systems and Technologies - CompSysTech.*, 2005.

Kornelakis, A. & Marinakis, Y., 2010. Contribution for optimal sizing of grid-connected PV-systems using PSO. *Renewable Energy*, 35(6), pp.1333-41.

Koutroulis, E., Kolokotsa, D., Potirakis, A. & Kalaitzakis, K., 2006. Methodology for optimal sizing of stand-alone photovoltaic/wind-generator systems using genetic algorithms. *Solar Energy*, 80(9), pp.1072-88.

Mahdavi, A., Hartkopf, V. & Mathew, P., 1999. *Towards the Building as a Power Plant: Computational Analysis of Building Energy Self-Sustenance*. Pittsburgh, Pennsylvania: Carnegie Mellon University.

Man, K.F., Kwong, S. & Tang, K.S., 1999. *Genetic algorithms: concepts and designs*. 1st ed. Springer.

Marczyk, A., 2004. *Genetic Algorithms and Evolutionary Computation*. [Online] Available at: http://www.talkorigins.org/faqs/genalg/genalg.html#strengths [Accessed 12 August 2011].

McConnell, J.J., 2007. *Analysis of algorithms: an active learning approach*. 2nd ed. London: Jones and Bartlett Publishers.

Mellit, A., Kalogirou, S.A. & Drif, M., 2010. Application of neural networks and genetic algorithms for sizing of photovoltaic systems. *Renewable energy*, 35(12), pp.2881-93.

Mellit, A., Kalogirou, S., Hontoria, L. & Shaari, S., 2009. Artificial intelligence techniques for sizing photovoltaic systems: A review. *Renewable and Sustainable Energy Reviews*, 13(2), pp.406-19.

Mitchell, M., 1998. *An introduction to genetic algorithms*. MIT Press.

Obitko, M., 1998. *Encoding*. [Online] Available at: http://www.obitko.com/tutorials/genetic-algorithms/encoding.php [Accessed 14 August 2011].

Parker, M. & Parker, G.B., n.d. *Using a Queue Genetic Algorithm to Evolve Xpilot Control Strategies on a Distributed System*. Paper. Computer Science, Indiana University.

Pohlheim, H., 2006. *Evolutionary Algorithms 1 Introduction*. [Online] (3.80) Available at: http://www.geatbx.com/docu/algindex.html [Accessed 9 August 2011].

Pohlheim, H., 2006. *Evolutionary Algorithms 2 Overview*. [Online] (3.80) Available at: http://www.geatbx.com/docu/algindex-01.html#P153_5403 [Accessed 9 August 2011].

Rieger, H. & Hartmann, A.K., 2002. *Optimization algorithms in physics*. 1st ed. Berlin: Wiley-VCH Verlag.

Scheaffer, R.L. & McClave, J.T., 1982. *Statistics for engineers*. Duxbury Press.

Solomatine, D.P., 1998. Genetic and other global optimization algorithms - comparison and use in calibration problems. In *Proc. 3rd Intern. Conference on Hydroinformatics*. Copenhagen, 1998. Balkema Publishers.

The MathWorks, I., 1994-2011. *Global Optimization Toolbox: Performing a Multiobjective Optimization Using the Genetic Algorithm*. [Online] Available at: http://www.mathworks.com/products/global-optimization/demos.html?file=/products/demos/shipping/globaloptim/gamultiobjfitness.html [Accessed 15 August 2011].

The Mathworks, I., 1994-2011. *Global Optimization Toolbox:Solve multiple maxima, multiple minima, and nonsmooth optimization problems*. [Online] Available at: http://www.mathworks.com/products/global-optimization/index.html [Accessed 4 July 2011].

Thiaux, Y., Seigneurbieux, J., Multon, B. & Ahmed, H.B., 2010. Load profile impact on the gross energy requirement of stand-alone photovoltaic systems. *Renewable Energy*, 35(3), pp.602-13.

Wall, M., 1996. *GAlib: A C++ Library of Genetic Algorithm Components*. Cambridge: Mechanical Engineering Department: Massachusetts Institute of Technology.

Wall, M., n.d. *GAlib:A C++ Library of Genetic Algorithm Components*. [Online] (2.4.7) Available at: http://lancet.mit.edu/ga/ [Accessed 20 May 2011].

Water, A.A., Abebe, A.J. & Solomatine, D.P., 1998. Application of Global Optimization to the Design of Pipe Networks. In *Proc. Int. Conf. Hydroinformatics-98.*, 1998.

Weise, T., 2009. *Thomas Weise*. 2nd ed. Free Software Foundation. Available at: http://www.it-weise.de/ [accessed July 2011].

Wikipedia, the free encyclopedia, 2011. *Deterministic Algorithms*. [Online] Available at: http://en.wikipedia.org/wiki/Deterministic_algorithm [Accessed 5 August 2011].

Wikipedia, the free encyclopedia, 2011. *Monte Carlo algorithm*. [Online] Available at: http://en.wikipedia.org/wiki/Monte_Carlo_algorithm [Accessed 5 August 2011].

Wikipedia, the free encyclopedia, 2011. *Randomized algorithm*. [Online] Available at: http://en.wikipedia.org/wiki/Randomized_algorithm [Accessed 5 August 2011].

Wikipedia, 2011. *Pareto efficiency*. [Online] Available at: http://en.wikipedia.org/wiki/Pareto_front [Accessed 18 Augustus 2011].

Yang, X.-S., 2010. *Engineering Optimization: An Introduction with Metaheuristic Applications*. Hoboken: John Wiley & Sons, Inc.

Yang, H., Zhou, W., Lu, L. & Fang, Z., 2008. Optimal sizing method for stand-alone hybrid solar–wind system with LPSP technology by using genetic algorithm. *Solar Energy*, 82(4), pp.354-67.