CHAPTER 5

HYBRID GENETIC ALGORITHMIC APPROACH TO ESTIMATION

5.1 INTRODUCTION

Researchers became disenchanted with classical and neo-classical models of intelligence so they looked in directions where they can find exact solutions to their problem. The two prominent fields which arose, connectionism in networking includes Evolutionary Computing and Artificial Neural Networks which supports parallel processing. This does computation capitulating true or approximate solutions to optimization and search problems. These algorithms are categorized as inclusive search heuristics. These are algorithms implemented as computer based simulators which represent a population called chromosomes also called genotype, Yielding candidate solutions to optimization problems which provide better and accurate solutions. This chapter deals with optimization of effort.

5.2 EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms is a branch of Artificial Intelligence that involves combinatorial optimization tasks. This Evolutionary Computation offers advantages like flexibility, robustness to adapting circumstances. Such characteristics made a wider scope of problem solving using EA. Evolutionary Computing is inspired by biological mechanism which is further evolved as a system of personalized programs. This programming technique
generates solution to large optimization and search problems. EA are relatively easier to find a good potential solution; this potential solution is further broken into discrete segments not of fixed size where a best solution was found out of it. EA caters major areas like automatic programming, signal processing, Bioinformatics, social systems and optimization. In many cases the system has mathematical functions, and the values are certainly imputed or simulated perfectly with the Intelligence. The components of EA are definition of individual representation, fitness function for evaluation, population, parent selection mechanism, variation operators, recombination, mutation and survivor selection for replacement.

GA’s have been applied fruitfully to solve scheduling problems in software projects like resource allocation and its estimation; task scheduling and allocation onto homogeneous multiprocessor systems. The hybrid algorithm combines the search capability of GA’s with a graphical approach to provide the best-possible solution to a effort estimation problem in an proficient manner when compared to other probabilistic techniques. The proposed algorithms are represented with its design. The exactness and the efficiency of this hybrid approach are evaluated by comparing them with COCOMO II Model. A hybrid imputational approach is presented in this chapter.

5.2.1 Advantages of EA

They performed best on certain problems with multimodal functions.

- Integration of priori knowledge on the problem domain is easier.
- They are stochastic.
• The evaluation of solution was done parallel except selection operation which is performed serially. Popular optimization techniques like simulated annealing do not support parallelism.

• EA can be ideal in places where adaptability is of major concern whereas traditional methods of optimization are robust to dynamic changes.

• The supreme benefit of EA lies to address problems which cannot permit human experts to solve it.

5.3 PREAMBLE TO GENETIC ALGORITHMS

General optimization problems which have large search spaces could be solved using Genetic algorithms. These were developed in order to address problems which have human experts or even in its absence. Genetic algorithms are used in this research to enhance the performance of effort estimation. This approach is useful to select the most relevant features of effort estimates and cost drivers which have certainly significant influence on the COCOMO effort estimation model even if the input level is incomplete. If uncertainty exists in the input parameters, errors in the estimates are more or accuracy may not be good.

5.4 BASIC PROCESS OF GENETIC ALGORITHM

The Genetic algorithm does combination of selection, crossover and mutation operators with the absolute aim of finding the most suitable solution to an estimation problem until actual criterion is met. Genetic algorithms start working with the initial population which evolves toward a population that is expected to produce good reasonable estimates. The solution to a problem in this domain is referred as a chromosome. This
chromosome is usually an array of bits. These chromosomes are composed of a group of genes. The input estimation parameters are referred as genes in this structure. The reproduction evaluation for each iteration in Genetic algorithms is called generation.

The probabilistic selection of chromosome from a current population is done based on the fitness function. Then the selected chromosomes are subjected to crossover, mutation and other operations. In comparison with the other Evolutionary Algorithms, Genetic Algorithms focuses on fixed length, in certain cases variable string of characters could also be used.

5.5 IS GENETIC PROGRAMMING AN EXTENSION OF GENETIC APPROACH

Genetic programming is a method of genetically breeding a population of program to find a solution for solving a problem. This is done iteratively to transform a population of computer program into new generation of program. Genetic programming is an extension of the Genetic algorithm. Programs are not represented as LOC but as syntax trees usually includes nodes and links. The instructions to be executed are referred as nodes and the arguments for each instruction are indicated by links.

5.6 GENETIC PROGRAMMING IN SOFTWARE ESTIMATION OF EFFORT – A IMPUTATIONAL APPROACH

Genetic algorithms and Evolutionary Computing are found suitable to solve incredibly complicated problems which are pretty simple but complicated for solving it. The productivity and suitability could be realized in areas where
- Mathematical analysis could not provide analytic solutions.
- Poor understandability of relationship about the variables is involved.
- Small enhancement in performance is highly appreciable.

Imputation is just the replacement of missing values in the engineering data with some estimation techniques and basic assumptions. The major problem that obstructs useful application of imputation methods is the presence of bias, when an estimator’s long-run average (expectation) differs from the quantity being estimated. This deviation which arises in estimation becomes a risk. When this difference is systematical, the results of analyses may be biased and may not be accurate; false conclusions are easily drawn with such end results.” (Huisman 2000) Definitely, an imputation method will be reasonable and dependable, if bias is eliminated while preserving the relationship between the data items and within the data. Missing data are frustration to a certain extent. Solving such issues is very intricate to put into operation and it is problem specific. Hence in short imputed data is used in order to find a suitable replacement for better accurate analyses. Imputation is a method of modifying for lost information. Missing reactions to information articles is a normal situation in any survey. This missingness regularly happens since the respondent denies or is unable to furnish information for a particular field or many and also because of typographical mistakes. Missing information may likewise effect from mis-keying or by editing process.

5.6.1 Is Imputation A Better Model or Not?

Definitely imputation model is a compatible one and is affluent enough which confirms based on the associations and relationships among the other variables in the data set. Imputation imposes models like probability
model, Decision tree models, neural models on the complete data set and also other models.

5.7 EFFORT ESTIMATION OPTIMIZATION USING GENETIC APPROACH

Genetic algorithm is used in this research to enhance the performance of effort estimation. This approach is useful to select the most relevant features of effort estimates and cost drivers which have certainly significant influence on the COCOMO effort estimation model even if the input level is incomplete. In order to increase the accuracy level of the proposed estimation described in the previous sections, we have implemented the formulated effort estimation through generic genetic algorithm.

Genetic algorithm uses projects as population (collection) of chromosomes (Real coded) which goes through genetic operations for estimating effort and repairing effort multipliers (genes) values to generate a new population. This process of evolution is guided by the fitness function which measures the goodness of the chromosomes is repeated by the predefined number of times until the given stop criteria is satisfied. The chromosome with best valued genes form the final population, reported as output of the algorithm. The basic operators involved solving the proposed effort estimation using GA are selection, crossover and mutation have been represented in Figure 5.1 which have been illustrated as follows.

The figure depicts the cyclic flow process of Reproduction, Crossover and Mutation for N populations. Effort estimation process starts with an initial population of N project (chromosomes) with 17 Effort multipliers (genes) as input which follows genetic operations of Reproduction, Crossover and Mutation. These genetic operators undergo series of process on each chromosome which is cyclic. Reproduction
performs the operation of selecting best pair of parents for effective effort estimation. In reproduction initial population chromosomes are grouped based on project mode and each chromosomes in each group is ranked based on fitness function. The opted pair of parental chromosomes undergoes crossover operation for repairing non-nominal effort multipliers weightage through adjustment on optimistic group (OG). Weightage of each non-nominal multiplier are adjusted by adding weightage or deducting weightage based on, weightage tracking made on all the chromosomes for specific genes. Two point crossover operation is performed for generating offsprings. Further mutation operation is carried out on each nominal gene of the offsprings generated which undergoes a series of process for offsprings effort estimation. Effort estimation is made succeeding nominal weightage adjustment of both pessimistic (PG) and optimistic group (OG) in each offspring. Mutation of 0.01% is carried over nominal genes in the offspring by means of weightage adjustment by adding or deducting from the original weightage. Termination check is made if N population has been completed inorder to terminate the process else the cyclic process from reproduction, crossover and mutation operation will be carried on till completions of effort estimation for N projects.

A population of Chromosomes is generated and this is also called as Genome in GA; this is represented in the form of simpler strings in certain cases other data structures are also preferred. Numerous individual solutions are in the beginning generated to form an initial population. The size of the population depends on the size and complexity of the problem. Population generation is performed randomly which covers the entire search space.
Figure 5.1. Work flow for genetic approach to effort estimation
5.8 OBJECTIVE AND FITNESS FUNCTION

GA’s operates on a number of potential solutions, called a population, consisting of some encoding of the parameter set simultaneously. The objective function provides a measure of how individual chromosomes have performed in the problem domain. If it is a minimization problem it will have a lowest numerical value. The fitness function used in this estimation problem is used to transform the objective function value into a measure of relative fitness thus

\[ F(x) = g(f(x)) \] (5.1)

Where

‘f’ is the objective function
‘g’ transforms the value of the objective function to a non-negative number and
and ‘F’ is the resulting relative fitness

This mapping is always necessary when the objective function is to be minimized as the lower objective function values correspond to fittest individuals. In many cases, the fitness function value corresponds to the number of offspring that an individual can expect to produce in the next generation.

The transformation function used for proportional fitness assignment is as follows

Individual Fitness:

\[ F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)} \] (5.2)
where $N_{\text{ind}}$ is the Population Size and $f(x_i)$ is Objective function value of ith individual. This does not change the position or location of the solution but it either minimizes or maximizes the problem.

**PSEUDO CODE FOR EFFORT ESTIMATION**

Pseudo code 1 // Effort Estimation using Genetic Algorithm

\[
\begin{align*}
N & \leftarrow \text{Population size} \\
P_{\text{max}} & \leftarrow \text{Sub population size} \\
P_n & \leftarrow \text{Number of iteration for generating sub population from N populations} \\
P_{\text{max}} & \leftarrow 0 \\
P_{\text{start}} & \leftarrow 1 \\
P_{\text{max}} & \leftarrow N / 2 \\
P_n & \leftarrow N / P_{\text{max}} \\
\text{For } j & \leftarrow 1 \text{ to } (P_n) \\
\text{For } i & \leftarrow P_{\text{start}} \text{ to } P_{\text{max}} \\
P & \leftarrow \text{Initialize sub population ( )} \\
P & \leftarrow \text{Perform Selection ( )} \\
(\text{Offi, Offi+1}) & \leftarrow \text{Crossover operation (Pi, Pi+1)} \\
\text{Offspring Evaluation (Offi, Offi+1)} \\
(\text{Offi}', \text{Offi+1}') & \leftarrow \text{Mutation operation on nominal valued genes (Offi, Offi+1)} \\
\text{Offspring Evaluation (Offi', Offi+1')} \\
\text{Minimizing the } f(x) \text{ for the offsprings by finding the value of ‘x’}
\end{align*}
\]

\[
f(x) = (\prod_{j=1}^{i} 0.5*LOC_{ij}^{1+PG\_EM_{ij}}) \times (\prod_{j=1}^{i} 0.5*LOC_{ij}^{1/PG\_EM_{ij}}) / LOC \quad \text{where, } 0 < x < 1 \text{ and } LOC \geq 1
\]
If \( j \leftarrow \text{Pmax And } i \neq \text{Pn} \) then

\[
\text{Pstart} \leftarrow \text{Pmax} + 1
\]

\[
\text{Pmax} \leftarrow \text{Pstart} + \text{Pmax}
\]

Else

\[
\text{Pstart} \leftarrow \text{Pmax} + 1
\]

\[
\text{Pmax1} \leftarrow N - \text{Pmax}
\]

\[
\text{Pmax} \leftarrow \text{Pstart} + \text{Pmax1}
\]

End if

End For

End For

The general structure of the effort estimation is shown in Pseudo code 1. This codes a solution from underestimation or overestimation of effort multipliers for accurate effort. To code a solution, projects with 17 effort multipliers are chromosomes, where genes are effort multipliers used as initial population. The Schema theorem proposes that “fit individuals having chromosomes with particular patterns of gene that perform well, termed schemata” (Goldberg D.E. 1989). The fit individuals find more chances in reproduction phase these are proliferated into the population and this increase the opportunity of better solutions. GA implicitly conducts a parallel search over a large number of schemata. This implicit parallelism is one of the reasons for the good performance of GA (Beasley D., Bull 1993). GA’s find facility to have good building block (Goldberg D.E. 1989) which has short length schemata which tends to enhance the fitness of an individual when integrated into it. Since fit individuals have more chances of reproduction, these schemata are propagated in the population, and the chance of finding better solution increases. Also, since every chromosome has
a large number of gene patterns, GA absolutely conducts a parallel search over a huge amount of schemata.

5.8.1 Encoding in GA

Encoding is the first step in Genetic Algorithm to solve any optimization problem. The result of this step is chromosomes or also called as decision variables. These decision variables are represented as individual genes in the chromosomes. The below figure shows how Chromosomes are represented. These individual chromosomes describe a “genotype” which corresponds to a single set of variables in the “phenotype” also called decision space.

<table>
<thead>
<tr>
<th>Phenotype</th>
<th>Chromosome</th>
<th>Genotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>X = [8,30,6]</td>
<td>0100 1111 0011</td>
<td>01000 11110 00110</td>
</tr>
<tr>
<td>Integer Encoding</td>
<td>8 30 6</td>
<td>8306</td>
</tr>
</tbody>
</table>

**Figure 5.2 Chromosome encoding**

Initially, Goldberg suggested the binary alphabet to be the most appropriate encoding system, as it provides the largest amount of schemata. This is the “traditional way” of encoding. Even so, it is possible to use other encoding systems, for example, integer numbers, sequences, or real numbers. Recently, it has been proven that there is no “best choice” for the encoding system and that a single given problem can be solved efficiently using different encoding philosophies (El-Sharkawi M 2008). The crucial aspect is that the algorithm, i.e. the encoding and the genetic operators, promote the processing of good “building blocks” (Deb K 2001). The 5 bit binary representation will be as follows.
Figure 5.3 Representation of chromosome

5.9 \textbf{INITIALIZATION}

The size of population is a major key factor to attain a competent GA. The efficiency of the algorithm is strongly dependent on this. Some of the researchers suggested that population sizes can range between 30 and 100 individuals (Song, Y.H., Irving 2001); but, Deb confirmed that the best size of the population depends on the complexity of the crisis and that there is no universal suggestion that applies to all category of problems. The more “difficult” the problem is the larger the population should be.

\textbf{PSEUDO CODE FOR EFFORT ESTIMATION}

Pseudo code 2 // initial population \textit{POP} of size \([Pmax \times CMax]\)

\begin{verbatim}
Start

Step 1: Set \(Pmax, CMax\)

Step 2: Initialize \(Pcur = 0\)

Step 3: Obtain the range of values for each \(EM\)

Step 4: Repeat through Step 6 Until \(Pcur <= Pmax\)

Step 5: Generate \(Indiv\) of size \([1 \times CMax]\)

\end{verbatim}
\[ \forall i [1 \leq i \leq CMax], \quad \text{Indiv}[i] = \text{UniformRandom}(\text{range}(EM_i)) \]

Step 6: Add the generated individual into the Population

\[ P_{cur} = P_{cur} + 1; \quad POP[P_{cur}] = \text{Indiv}; \]

Step 7: return POP

Stop

Each project considered to be a chromosome in the population with effort multipliers values as genes. The matter of generation of sub-populations depends on the total population size.

\[ P_{\text{max}} = \frac{N}{2} \]

(5.3)

\[ P_n = \frac{N}{P_{\text{max}}} \quad (5.4) \]

where,

- \( N \rightarrow \) Population size
- \( P_{\text{max}} \rightarrow \) Sub population size
- \( P_n \rightarrow \) Total iteration for generating sub populations

To evaluate real value encoding problems like effort estimation representing chromosomes has been used as a straight forward way to work on real-time data. Fitness function in GA plays one of the major roles in effective chromosomal selection. Direct proportionality is used in fitness evaluation considering four genes (LEXP, PCAP, CPLX and ACAP) as a base for best pairing of chromosomes as parents. Reason behind for considering specifically four effort multipliers for fitness function evaluation
Problem Complexity (CPLX) requires efficient analyzing capability for effective problem solving i.e. Application capability (ACAP) which could be represented as below in Equation (5.5)

\[ \text{CPLX} \propto \text{ACAP} \tag{5.5} \]

Apart from problem solving capability programming capability (PCAP) and language experience (LEXP) are required for effective problem solving which could be represented as below in Equation (5.6)

\[ \text{LEXP} \propto \text{PCAP} \tag{5.6} \]

From Equation (5.5) and (5.6) Fitness Function has been expressed in Equation (5.7)

\[ F(x) = \frac{\text{LEXP}}{\text{PCAP}} \left[ \frac{\text{CPLX}}{\text{ACAP}} \right] \tag{5.7} \]

where,

- \( F(x) \) → Fitness Function
- LEXP → Language experience Effort multiplier
- PCAP → Programming capability Effort multiplier
- CPLX → Problem complexity Effort multiplier
- ACAP → Application capability Effort multiplier

The population size in this estimation was large enough to provide the GA with good number of schemata to initiate the search. The diversity of the schema will result in convergence of the GA towards broader global
optima. But on the same time this larger population involves greater inertia in evolving towards its optimality.

5.10 REPRODUCTION

The selection process could also be represented a Reproduction. The mode for selecting each project is based on project modes and fitness function evaluated for that project. Note that the project mode need not necessarily have the same number of modes taken for selection. In selection technique integration of grouping and ranking technique has been followed for best parents pairing apart from fitness evaluation. The selection techniques have been represented in Pseudo code 2, which is an integration of grouping and ranking techniques. The idea behind this is to utilize the advantage of grouping and ranking techniques for identifying best matching pairs of chromosomes (projects) for performing crossover. First the chromosomes are grouped based on mode of projects for N populations. Secondly the chromosomes under each group are sorted in descending order based on fitness function. Using this technique best pair of chromosome is selected for generating offsprings.

Individual genomes are selected from the initial population for later breeding called a selection process. Fitness functions for each individual chromosome providing fitness value. Individual solutions are chosen through a fitness-based process, where fitter solutions are classically more expected to be chosen.

The selection of individuals can be viewed as two separate processes:
1) One process is the determination of the number of trials an individual can expect to receive, and

2) Conversion of the expected number of trials into a discrete number of offspring.

The first part is concerned with the transformation of raw fitness values into a real-valued expectation of an individual’s probability to reproduce. The second process is the probabilistic selection of individuals for reproduction based on the fitness of individuals relative to one another and is sometimes known as sampling.

5.10.1 Selection

The fitness function is evaluated for each individual, providing fitness values, which are then normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that the sum of all resulting fitness values equals 1.

The population is sorted by descending fitness values.

Accumulated normalized fitness values are computed (the accumulated fitness value of an individual is the sum of its own fitness value plus the fitness values of all the previous individuals). The accumulated fitness of the last individual should be 1 (otherwise something went wrong in the normalization step).

Random number is selected in the range of 0 to 1.

The selected individual is the first one whose accumulated normalized value is greater than R.
If this procedure is repeated until there are enough selected individuals, this selection method is called fitness proportionate selection or roulette-wheel selection. If instead of a single pointer spun multiple times, there are multiple, equally spaced pointers on a wheel that is spun once, it is called stochastic universal sampling. Repeatedly selecting the best individual of a randomly chosen subset is tournament selection. Taking the best half, third or another proportion of the individuals is truncation selection.

Individuals for producing offspring are chosen using a selection strategy after evaluating the fitness value of each individual in the selection pool. Each individual in the selection pool receives a reproduction probability depending on its own fitness value and the fitness value of all other individuals in the selection pool. This fitness is used for the actual selection step afterwards. Some of the popular selection schemes are Roulette Wheel Selection and Tournament Selection.

### 5.10.2 Roulette Wheel Selection

Many selection techniques employ a “roulette wheel” mechanism to probabilistically select individuals based on some measure of their performance. A real-valued interval, Sum, is determined as either the sum of the individuals’ expected selection probabilities or the sum of the raw fitness values over all the individuals in the current population. Individuals are then mapped one-to-one into contiguous intervals in the range \([0, \text{Sum}]\). The size of each individual interval corresponds to the fitness value of the associated individual. For example, in Figure 5.4 shows the circumference of the roulette wheel is the sum of all six individual’s fitness values. Individual 5 is the most fit individual and occupies the largest interval, whereas individuals 6 and 4 are the least fit and have correspondingly smaller intervals within the roulette wheel. To select an individual, a random number is generated in the interval \([0, \text{Sum}]\) and the individual whose segment spans the random number is
selected. This process is repeated until the desired number of individuals has been selected.

Figure 5.4 Roulette wheel selection

The basic idea behind this roulette wheel selection method is stochastic sampling with replacement (SSR). Here, the segment size and selection probability remain the same throughout the selection phase and individuals are selected according to the procedure outlined above. SSR gives zero bias but a potentially unlimited spread. Any individual with a segment size $> 0$ could entirely fill the next population. Stochastic sampling with partial replacement (SSPR) extends upon SSR by resizing an individual’s segment if it is selected. Each time an individual is selected, the size of its segment is reduced by 1.0. If the segment size becomes negative, then it is set to 0.0. This provides an upper bound on the spread of. However, the lower bound is zero and the bias is higher than that of SSR. Remainder sampling methods involve two distinct phases. In the integral phase, individuals are selected deterministically according to the integer part of their expected trials. The remaining individuals are then selected probabilistically from the
fractional part of the individual’s expected values. Remainder stochastic sampling with replacement (RSSR) uses roulette wheel selection to sample the individual not assigned deterministically. During the roulette wheel selection phase, individual’s fractional parts remain unchanged and, thus, compete for selection between “spins”. RSSR provides zero bias and the spread is lower bounded. The upper bound is limited only by the number of fractionally assigned samples and the size of the integral part of an individual. For example, any individual with a fractional part > 0 could win all the samples during the fractional phase. Remainder stochastic sampling without replacement (RSSWR) sets the fractional part of an individual’s expected values to zero if it is sampled during the fractional phase. This gives RSSWR minimum spread, although this selection method is biased in favour of smaller fractions.

5.10.3 Tournament Selection

In tournament selection a number of individuals are chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals to choose. These selected parents produce uniform at random offspring. Fitness proportionate selection, also known as roulette wheel selection, is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination.

Pseudo code 3:  // Selection Technique

PGroup ← Total group classification

For i ← Pstart to Pmax

P ← PGroup  // Grouping of chromosomes based on project mode
End For
For j \leftarrow 1 \text{ to } P\text{Group}
For i \leftarrow P\text{start} \text{ to } P\text{max}

\text{Sorting} (\text{PGroup}[\text{Pi, Pi+1}]) \quad //\text{Fitness based Ranking in descending in each group}
\text{Select} ([\text{Pi, Pi+1}]) \quad //\text{Roulette Wheel selection from the sorted chromosomes}
End For
End For

\section*{5.11 CROSSOVER}

In genetic algorithms, crossover is a genetic operator used to vary the chromosomes from one generation to the next. This step is analogous to reproduction and biological crossover, upon which genetic algorithms are based. A new chromosome is produced by combining (Mating) two chromosomes which is called as offspring that are produced in Genetic operations. The background plan behind crossover is that the new chromosome may be enhanced than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. Here in this problem the crossover operation is applied with a probability called Crossover rate. This ranges between 0.6 to 1 (Beasley D and Bull D 1993). The studies show that the rate is between 0.65 and 0.85. The parents which are not combined are directly taken to the next population. The types of Crossover are single point crossover, two point crossover, arithmetic crossover and uniform crossover.

\subsection*{5.11.1 Single Point Crossover}
In this crossover the binary string from beginning of chromosome to the crossover point is copied from one parent, and the remaining is copied from the second parent.

In the following example two parents have been selected in the crossover. The symbol “|” is used for randomly choosing the crossover point.

Parent 1: 11011|011
Parent 2: 00100|111

The following offspring was produced after the interchange of parent chromosomes using single point crossover.

Offspring1: 11011|111
Offspring2: 00100|011

5.11.2 Two Point Crossover

In Two Point crossover, crossover points are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent.

Consider the following 2 parents which have been selected for crossover. The “|” symbols indicate the randomly chosen crossover points.

Parent 1: 110|110|11
Parent 2: 001|001|11

Using the above parent chromosomes after interchange the following offspring are produced.

Offspring1: 110|001|11
Offspring2: 001|110|11
**Arithmetic crossover** – In this crossover new offspring was generated using some arithmetic operations.

A crossover operator that linearly combines two parent chromosome vectors to produce two new offspring according to the following Equations (5.8, 5.9)

\[
\text{Offspring}_1 = a \times \text{Parent}_1 + (1- a) \times \text{Parent}_2 \tag{5.8}
\]

\[
\text{Offspring}_2 = (1 - a) \times \text{Parent}_1 + a \times \text{Parent}_2 \tag{5.9}
\]

Where ‘a’ is a random weighting factor (chosen before each crossover operation).

Consider the following 2 parents (each consisting of 4 float genes) which have been selected for crossover:

Parent 1: (0.3) (1.4) (0.2) (7.4)

Parent 2: (0.5) (4.5) (0.1) (5.6)

If \(a = 0.7\), the following two offspring would be produced:

Offspring1: (0.36) (2.33) (0.17) (6.86)

Offspring2: (0.402) (2.981) (0.149) (6.842)

5.11.3 **Heuristics**

In this heuristics, a crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring are created according to the following Equations (5.10)
Offspring1 = BestParent + r * (BestParent – WorstParent)

Offspring2 = BestParent

where r is a random number between 0 and 1.

It is possible that Offspring1 will not be feasible. This can happen if r is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds. For this reason, heuristic crossover has a user settable parameter (n) for the number of times to try and find an ‘r’ those results in a feasible chromosome. If a feasible chromosome is not produced after n tries, the Worst Parent is returned as Offspring1.

A crossover operator that decides (with some probability – know as the mixing ratio) which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover).

For some problems, this additional flexibility outweighs the disadvantage of destroying building blocks. Consider the following 2 parents which have been selected for crossover:

Parent 1: 11001010

Parent 2: 00100111

If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. Below is a possible set of offspring after uniform crossover:

Offspring 1: 1102101021021112
The next steps in creating a new population are the Mating and Crossover: As described above previously there exist variety of different types of Mating. One easy to understand type is the random mating with a defined probability and the b_nX crossover type. This type is described most often, as the parallel to the Crossing Over in genetics and is evident:

1. $P_M$ percent of the individual of the new population will be selected randomly and mated in pairs.
2. A crossover point will be chosen for each pair as shown in Figure. 5.5.
3. The information after the crossover-point will be exchanged between the two individual of each pair.

Genetic crossover operation performed on the selected pair of chromosome preceding mutation has been represented in Pseudo code 3.
Pseudo code 4: // Crossover Operation

POG ← Total effort multipliers in OG group

PPG ← Total effort multipliers in PG group

CMax ← Total effort multipliers of OG and PG group

G ← Effort multiplier of a chromosome

Gmax ← Maximum value of specific effort multiplier

Gmin ← Minimum value of specific effort multiplier

For i ← Pstart to Pmax

P ← Select (Pi, Pi+1) //Selecting parents for performing crossover

For j ← 1 to CMax

If Gi,j ≠ Gi+1,j then

Swap (Gi,j, Gi+1,j) //Gene of parents found to be dissimilar swap respective gene

End if

End For

For j ← PPG+1 to CMax // Performs non-nominal EM weightage adjustment in PG group

Gmax ← Gmin // Min EM weightage of whole P is adjusted to Max of whole P

Gmin ← Gmax // Max EM weightage of whole P is adjusted to Min of whole P

End for

For j ← 1 to CMax then

Offi,j ← Pi,j //EM adjusted chromosome of Pi will be copied to Offi

Offi+1,j ← Pi+1,j //EM adjusted chromosome of Pi+1 will be copied to Offi+1

End for

End for
Effort multiplier genes with dissimilar weightage are identified from the chromosomes for which swapping of gene weightage is performed. Gene Transformation is carried to identify gene convergence level where, at most necessity for an effort multiplier gene that has to be repaired that has been represented in Fig.5.1. Repairing is performed on effort multiplier of OG group by means of gene weightage adjustment. Carrying out weightage adjustment by adding or deducing weightage through tracking whole population is done.

Figure5.8. illustrates repairing carried on OG group. Highly weighted gene will be deduced to its preceding low weightage while, for low weightage gene will be added to its succeeding high weightage. Here only non-nominal genes are only considered for repairing. Two point crossover operations are performed on chromosomes for offspring generation considering gene transformation as a base.
Figure 5.6. OG Group gene transformation
**Figure 5.7. PG Group gene transformation**

<table>
<thead>
<tr>
<th></th>
<th>RELY</th>
<th>DATA</th>
<th>CPLX</th>
<th>TIME</th>
<th>STOR</th>
<th>VIRT</th>
<th>TURN</th>
<th>ACAP</th>
<th>AEXP</th>
<th>PCAP</th>
<th>VEXP</th>
<th>LEXP</th>
<th>MODP</th>
<th>TOOL</th>
<th>SCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.15</td>
<td>0.94</td>
<td>1.3</td>
<td>1.3</td>
<td>1.21</td>
<td>1.15</td>
<td>1</td>
<td>0.86</td>
<td>1</td>
<td>0.86</td>
<td>0.9</td>
<td>1.07</td>
<td>0.91</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>1.4</td>
<td>0.94</td>
<td>1.3</td>
<td>1.11</td>
<td>1.56</td>
<td>1.07</td>
<td>0.86</td>
<td>0.82</td>
<td>0.86</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P1</td>
<td>1.4</td>
<td>0.94</td>
<td>1.3</td>
<td>1.11</td>
<td>1.56</td>
<td>1.15</td>
<td>1.15</td>
<td>0.86</td>
<td>1</td>
<td>0.86</td>
<td>0.9</td>
<td>1.07</td>
<td>0.91</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>1.15</td>
<td>0.94</td>
<td>1.3</td>
<td>1.3</td>
<td>1.21</td>
<td>1.07</td>
<td>0.86</td>
<td>0.82</td>
<td>0.86</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Figure 5.8. OG Gene Repairing**
<table>
<thead>
<tr>
<th></th>
<th>RELY</th>
<th>DATA</th>
<th>CPLX</th>
<th>TIME</th>
<th>STOR</th>
<th>VIRT</th>
<th>TURN</th>
<th>ACAP</th>
<th>AEXP</th>
<th>PCAP</th>
<th>VEXP</th>
<th>LEXP</th>
<th>MOOP</th>
<th>TOOL</th>
<th>SCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>P2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>M1</td>
<td>1.15</td>
<td>0.94</td>
<td>1.3</td>
<td>1.3</td>
<td>1.21</td>
<td>1</td>
<td>1.07</td>
<td>1.19</td>
<td>1.29</td>
<td>1.17</td>
<td>1.1</td>
<td>1.07</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O1</td>
<td>1.4</td>
<td>0.94</td>
<td>1.3</td>
<td>1.11</td>
<td>1.56</td>
<td>1.15</td>
<td>1.19</td>
<td>1.17</td>
<td>1.21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.9 Two - Point Crossover
5.12 MUTATION

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than it was previously possible. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set to high, the search will turn into a primitive random search.

Genetic Server and Genetic Library include the following types of mutation:

Flip Bit - A mutation operator that simply inverts the value of the chosen gene (0 goes to 1 and 1 goes to 0). This mutation operator can only be used for binary genes.

Boundary - A mutation operator that replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for integer and float genes.

Non-Uniform - A mutation operator that increases the probability that the amount of the mutation will be close to 0 as the generation number increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the later stages of evolution. This mutation operator can only be used for integer and float genes.
Uniform - A mutation operator that replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian - A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes.

Mutation operation performed on the offsprings and its evaluation has been represented in pseudo code 4 and pseudo code 5 has been explained as follows.

```
Pseudo code 5: // Mutation

For i ← Pstart to Pmax
For j ← 1 to CMax
mvar ← Gi+1,j -1 //Identifies Mutation variation for each gene of parents
Pmvar ← 0.01 percentage of mvar //Identifies percentage of variation
If Gi+1,j > Gi,j then //Performs Gene weightage comparison
Gi+1,j ← (Gi+1,j - mvar) + Pmvar //0.01 percent of gene value will be added
Gi,j ← (Gi+1,j + mvar) - Pmvar //0.01 percent of gene will be reduced
Else
Gi,j ← (Gi,j - mvar) + Pmvar
Gi+1,j ← (Gi+1,j + mvar) - Pmvar
End if
If j > PPG then
```
OG\textsubscript{i} $\leftarrow$ OGi * Gi,j  
//Calculates overall weightage of OG group of Pi

OG\textsubscript{i+1} $\leftarrow$ OGi+1 * Gi+1,j  
//Calculates overall weightage of PG group of Pi+1

Else

PG\textsubscript{i} $\leftarrow$ PGi * Gi,j  
//Calculates overall weightage of PG group of Pi

PG\textsubscript{i+1} $\leftarrow$ PGi+1 * Gi+1,j  
//Calculates overall weightage of PG group of Pi+1

End if

End For

For j $\leftarrow$ 1 to CMax

Off\textsubscript{i} , Off\textsubscript{i+1} $\leftarrow$ Offspring Evaluation()

End For

End For

Pseudo code 6:  
// Offspring Evaluation

\text{LOC} $\leftarrow$ Lines of coding in each project

\alpha $\leftarrow$ effort multipliers in PG group

\beta $\leftarrow$ effort multipliers in OG group

For i $\leftarrow$ Pstart to Pmax  
//Performs calculation for estimating Effort Person Months

For j $\leftarrow$ 1 to CMax  
//Performs Estimation for OG and PG Groups

\text{Maximize } F(x) , \left( \prod_{i=1}^{n} \left( 0.5 * LOC_i \right)^{EM_i} \prod_{i=1}^{n} \left( 0.5 * LOC_i^{\alpha} \right)^{EM_i} \right) / LOC_i , 0 < x < 1

End For
Offi ← EPMi
Offi+1 ← EPMi+1
End For

The fitness function uses an evaluation function to calculate a value of effort for the individual so that they can be compared against the actual. The mutation operator offers a search element to the Genetic Algorithm. It is very vital because it keeps the diversity of the population by searching regions of the choice space that were not explored in the past or by bringing back genes that were detached through selection. After crossover this operator is applied to the new offspring. Mutation operation is performed on nominal values of the offsprings by means of gene weightage adjustment of 0.01% based on gene weightage comparison. Here we have considered statically a minimal of 0.01% of mutation. First mutation variation level is identified based on which percentage of variation is identified. Secondly based on the percentage of variation weightage is added or deduced for the specific offsprings effort multiplier genes.

\[
m_{\text{var}} = \text{Em}_\text{Weightage} - 1
\]

\[
P_{\text{mvar}} = \frac{(0.01 \times m_{\text{var}})}{100}
\]

\[
\text{Em} = (\text{Em} - m_{\text{var}}) + P_{\text{mvar}}
\]

\[
\text{Em} = (\text{Em} + m_{\text{var}}) - P_{\text{mvar}}
\]

where,

\[
\text{Em} \rightarrow \text{Gene weightage mutated}
\]

\[
\text{Em}_\text{Weightage} \rightarrow \text{Gene weightage}
\]
mvar→ mutation variation
Pvar→ percentage level of mutation

5.12.1 Updation of Population Using Reinsertion and Elitism

The final step of GA is to re-insert offspring into the population. Population is kept constant in all generations; if reinsertion is performed then replacement has to be carried. A traditional replacement mechanism is to replace all old population with the reinserted offspring. A major point in updating population is elitism. The best performing chromosome individuals from old population are given an opportunity to survive directly in the new population. This is termed as “survival of the fittest”. Comparison between the parent and its offspring is made the best among the individuals from the old population to the new population. This ensures that always a good solution is given a chance to reenter, which has lost its opportunity by crossover or in mutation. This is a major concept in Multi-objective EA.

The reason behind for undergoing a minimal of 0.01% of mutation is to avoid elimination of nominal effort multiplier genes for effective and accurate effort estimation by avoiding under estimation. Since LOC is used as major factor for computing effort person month, the effort estimation formula by incorporating Cartesian product of OG group weightage of eq. (5.10) and PG group weightage of eq. (5.11), offsprings effort person month is evaluated as represented in eq. (5.12).

\[
OG = \left( \sum_{G_{wi}} \left[ 0.5 \times \text{pow}(LOC,(2/OG)) / \text{pow}(LOC,0.88) \right] \right) \tag{5.10}
\]

\[
PG = \left( \sum_{G_{wi}} \left[ 0.5 \times \text{pow}(LOC,(2 \times PGE)) / \text{pow}(LOC,0.88) \right] \right) \tag{5.11}
\]

\[
EPM = \frac{(PG \times OG)}{\text{pow}(3.0 \times \text{pow}(code_t,1.12),(\alpha + \beta - 1))} \tag{5.12}
\]
where,

\[
\begin{align*}
EPM &\rightarrow \text{Effort Person Month} \\
OG &\rightarrow \text{Optimistic Group} \\
PG &\rightarrow \text{Pessimistic Group} \\
LOC &\rightarrow \text{Lines of Code}
\end{align*}
\]

Elitism is avoided here by means of selecting new pair of parents, inorder to maintain the accuracy of effort estimation made through original datasets. Estimation of effort is carried out for the whole Population and the cyclic process terminates.

5.12.2. Convergence

If the GA is implemented properly, the population ultimately evolves and locates solution in the areas of it global optima (Beasley D., Bull D., Martin R. 1993). All GA’s are iterative and requires stopping criteria. The most common criteria are single objective. Here all genes are said to converge if 95% of the population has the value in common. The fitness value of the best in the population does not change in the continuous generations.

5.13 SUMMARY

Genetic algorithms have been applied effectively to solve a wide range of effort estimation problems in recent years. GAs provides better optimized results. GA can readily handle with integer variables, non-convex and non-differentiable functions. GA also supports working with a group of potential solutions. It avoids getting trapped in local minima. It solves problems by a natural way even with multi-objective problems. GA is good for “noisy” objective evaluations.
There are several software effort estimation models that can be used to find out effort estimation in software development. Hybrid model, when used for software effort estimation, offers certain unique advantages it provides a point for a software development effort, it estimates with Graphical and Genetic approach to improve the software effort forecast. Most of projects causes over or under estimation due to the uncertainty in the range of cost driver values. Definitely imputation model is a compatible one and is affluent enough which confirms based on the associations and relationships among the other variables in the data set. Imputation imposes models like probability model, Decision tree models, neural models on the complete data set and also other models. The ability of the proposed model is validated in the next chapter using performance factors like MRE, MMRE, PRED(n), RMSE, SF, MARE and VARE.