CHAPTER 2

LITERATURE SURVEY

John Holland’s pioneering book, “Adaptation in Natural and Artificial Systems” (1975, 1992) showed it is inspired by the natural evolution process. The goal of the study was to show the adaptive nature of the evolution process where an individual’s survival capacity is determined by various factors that characterize it. This analogy of natural behavior with highly parallel techniques led to the development of Genetic Algorithm (GA). The existing adaptive technology steadily exploits the field of artificial intelligence to solve real world problems in science, engineering, economics, etc.

2.1 GENETIC ALGORITHM

GA is mainly designed to represent an intelligent exploitation of a random search to solve difficult optimization problems using the iterative procedure. It involves a set of elements which typically leads to better individuals through the genetic operators. This implies that GA is very simple to use.

Due to the fact that, GA are robust and adaptive in nature, its application is not confined to the problem and can be applied to combinatorial problems of any discipline with slight modification of its elements. Since, GA is not a problem specific its basic structure remains the same even if any modification of the elements like representation, fitness value, choice of parameter, operators take place. This feature of GA solves difficult problems quickly, accurately, and reliably where little is known with the knowledge of the fitness function of the individuals.

Application of GA to an optimization problem needs to determine the connections between numerous components and determining the set of parameters to ensure the
system is robust and cost effective (Stender, 1994). For optimization problems, one is looking for some individual solutions better than the others. Borrowing heavily from the language of biological evolution, these individuals are said to be the fittest in a population of competing solutions. Then over successive iterations, the GA evolves the population of competing solutions until it becomes consistently very fit individuals. GA being a randomized search algorithm, in a large solution space it acts as an intelligent tool to explore the prospective regions for finding better solution in an acceptable amount of time.

### 2.1.1 Basic Operation of Genetic Algorithm

Basically, GA operation starts with an encoding of decision variables, the chromosomes that will be replicated to form the population. A fitness value is assigned to each individual indicates how well, it is better or close to a better solution. Genetic operators affect the individuals of a population and represent a possible solution to a given problem. The best individuals with a high fit are reproduced by the operational parameters of crossover and mutation for the next generation. In standard genetic algorithms (SGA), (D. E. Goldberg, 1989) this procedure is repeated a number of times until the termination condition specified by the user is satisfied towards convergence to find the optimal solution (Hajela, 1992) (Stender, 1994).

The steps of applying a GA are as follows (Melanie, 1998):

**Step 1:** Create an initial population of \( N \) individuals randomly for evolution.

**Step 2:** Define a suitable fitness function for the individuals.

**Step 3:** Perform genetic operations (crossover and mutation) to generate offspring.

**Step 4:** Evaluate the fitness of each individual.

**Step 5:** Select \( N \) superior individuals according to their fitness values to form the next generation.

**Step 6:** If the termination criterion is not satisfied, go to Step 3; otherwise, stop the algorithm.
2.1.2 Building Blocks of Genetic Algorithm

The design of GA is structured with the key components in order to choose an encoding, a fitness function, operators, parameters and initialization method and termination condition.

- **Chromosome/ Individuals:** each chromosome constitutes a set of individual structures called genes, each of which contains the solution associated with a specific parameter or variable of the search space.

- **Population:** Number of individuals with the same length of chromosome.

- **Fitness Function:** The value assigned to an individual based on how far or close an individual is from the solution, better the fitness value, better the solution it contains.

### Encoding of Chromosome

GA process begins with a representation that describes the problem states. It is initialized with a population of individuals randomly with the suitable encoding. The encoding specifies a mapping of possible solutions (individuals) to the problem into structures. The structure that encodes a solution is called a chromosome or individual. A decision variable is called a gene and its value is called an allele. The most commonly used way of encoding is a binary string as shown.

| Chromosome 1 | 1101100100110110 |
| Chromosome 2 | 1101111000011110 |

### Evaluation Function

The next step of GA is to specify a function, a procedure to assign a score to any possible solution or structure in the population. Using a biological metaphor, the score is a numerical value, fitness of the individual solution that indicates how well the individual adapts to the environment. The task of GA is to discover solutions
that have high fitness values among the set of all possible solutions in every successive generation.

**Design Configurations of Genetic Operators**

Once the encoding and the fitness function are specified, the individuals of each generation are combined to evolve new solutions. In general, GA uses several genetic operators like selection, crossover, mutation, etc. Before explaining more about crossover and mutation, some details of selection procedure is discussed:

**Selection**

The selection process simulates the “survival-of-the-fittest” and replicates individuals with high fitness values and removes the individuals with low values to give preference to better individuals. There are various techniques to implement this operator and some of the common techniques are Proportionate selection (Goldberg 2005) Tournament selection (Brindle 1981) and variants of tournament selection (Sokolov and Whitley, 2005), Ranking selection (Baker, 1987).

**Genetic Operators**

In order to explore new solutions, the GA relies on two variation operators: crossover and mutation. There are many ways to perform crossover and mutation but examples of commonly used are:

**Reproduction**

After the construction of the intermediate population by the selection operator is complete, recombination occurs to create the next population from the intermediate population based on its fitness. An important aspect of this operation is that every individual has some probability of selection, even if the fitness is poor.
Crossover

The crossover, recombination operation involves two individuals, i.e. a point between genes in the search space is chosen and the genes between those points are swapped between the two individuals. The crossover operation produces two offsprings.

Each offspring contains some genetic material from each of its parents as shown. The “|” symbol indicates the randomly chosen crossover point.

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>11011</th>
<th>00100110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>11011</td>
<td>11000011110</td>
</tr>
<tr>
<td>Offspring 1</td>
<td>11011</td>
<td>11000011110</td>
</tr>
<tr>
<td>Offspring 2</td>
<td>11011</td>
<td>00100110110</td>
</tr>
</tbody>
</table>

Standard Genetic Algorithm (SGA) uses the one point crossover, where mating chromosomes are cut once. Other crossover techniques have also been devised, often involving more than one cut point. In two point crossover, chromosomes are regarded as loops by connecting the ends together. Two cut points decide a segment, and two chromosomes exchange the segment. It performs the same task as one point cross over, but more general.

Single-Point crossover (Holland, 1992)

A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offsprings. Consider the following two parents which have been selected for crossover.

Chromosome 1: 11001|010

Chromosome 2: 00100|111
After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring 1: **11001|111**

Offspring 2: **00100|010**

**Two Point Crossover**

A crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offsprings. Consider the following two parents which have been selected for crossover.

Chromosome 1: **110|010|10**

Chromosome 2: **001|001|11**

After interchanging the parent chromosomes between the crossover points, the following offsprings are produced:

Offspring 1: **110|001|10**

Offspring 2: **001|010|11**

**Uniform Crossover (Syswerda, 1989)**

A crossover operator that decides (with some probability – known as 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 gene.

Consider the following two parents which have been selected for crossover:

Chromosome 1: **11001010**

Chromosome 2: **00100111**
If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from Chromosome 1 and the other half will come from Chromosome 2. A possible set of offspring after uniform crossover are:

Offspring1: \(1_10_21_20_10_21_11_2\)

Offspring2: \(0_21_10_21_11_20_1\)

Note: The subscripts indicate which parent the gene came from.

**Mutation**

Mutation is to prevent all solutions in population falling into a local optimum. The mutation process allows creation of new individuals. For single bit genes, a point along the string is selected at random and the bit is switched from one to zero or from zero to one. For multi-bit genes, a new valid pattern is chosen from its allowed values. The mutation depends on the encoding as well as the crossover is shown below.

Original offspring 1 11011100011110

Original offspring 2 110110100110110

Mutated offspring 1 11011100011110

Mutated offspring 2 11011100011110

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 a better solution than the previous solutions. 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 choice).
**GA Parameters**

Apart from other parameters of GA like population size, number generations, elitism strategy, the crossover and mutation probability play a vital role in determining the exploration and exploitation characteristics of GA (DeJong and Spears, 1990).

*Number of Generations* defines the maximum size of the run.

*Population size*

In practice, much concentration is given to the population size in designing GA is a critical parameter that plays a vital role which affects the efficiency of the algorithm (Lobo and Lima, 2005). Few chromosomes in the population would have few possibilities to explore only a small search space and may result in poor performance. On the other hand, too many chromosomes in the population slow down the search process by exploring more space and prevent premature convergence to local solutions. At the same time, a large population needs more evaluations per generations and may slow down the convergence rate. Previous research work shows that after reaching some limit, there is no use in increase of the population size which does not solve the problem faster and the determination and optimization of population size reviewed in (Khan, 2000) Throughout the GA run when the population size remains unchanged can be classified as static and when it is adjusted during execution can be classified as dynamic (Smith and Smuda, 1995), (Arabas, et al., 1994), (Voosen and Muhlenbein, 1994), (Hinterding et al., 1996), (Back et al., 2000).

*Selection Probability*

It gives the probability that design configurations are not changed during the evolution.
**Crossover probability**

Crossover rate is the probability indicating how often crossover will be performed within two chromosomes. A higher crossover rate introduces new strings more quickly into the population. If the crossover rate is too high, high-performance chromosome are eliminated faster than producing improvements. A low crossover rate may cause stagnation due to the lower exploration rate. For uniform crossover, a higher probability of contributing ones parents’ allele lowers the rate of disruption.

**Mutation probability**

Mutation rate is the probability with which the design configuration is randomly changed i.e. each bit position of a chromosome change randomly leading to a new population. Mutation is made to prevent falling GA into local extreme, but it should not occur very often which will result in random search. A low mutation rate helps to prevent any bit positions from getting stuck to single values, where as high mutation rate results in essentially random search. The mutation parameters of the genetic algorithm are reviewed in (Back et al., 2000).

**Initialization method and Termination Criteria**

The last steps of applying a GA are termination criteria. The genetic algorithm is usually initialized with a population of random individuals, but sometimes a fraction of the population is initialized with previously known (good) solutions. Following the initialization step, each individual is evaluated according to the user’s specified fitness function. Thereafter, the GA simulates evolution on the artificial population of solutions using operators that mimic the survival-of-the-fittest and principles of natural genetics such as recombination and mutation. The process is repeated a number of times (or generations) until some specified stopping criteria is met. A number of criteria can be chosen for this purpose, including among others, a maximum number of generations or time elapsed, on reaching predefined fitness value, or at population convergence.
Recommended Parameter Settings

There is no general theory to set the parameter values of GA even though tuning of various parameters are very important for the efficiency of the algorithm. Hence, some recommendations for standard parameter setting to implement the genetic algorithm suggested in the literature are:

- **Population size:** It may be surprising, that very big population size usually does not improve performance of GA (means speed of finding solution). Good population size is about 20-30 based on the problem.

- Crossover rate should generally be high, about 80%-95%.

- Mutation rate should be normally very low and best rates reported are about 0.5%-1%.

- However, some research suggested Population size =50, Crossover rate = 0.6 (De Jong and Spears, 1990), Population size = 20-30, Crossover rate = 0.75 – 0.95, Mutation rate = 0.005 – 0.01(Schaffer et al., 1989) and (Grefenstette, 1986) reported Population size = 30, Crossover rate = 0.9, Mutation rate = 0.01.

2.2 SELF-ORGANIZATION

Technological systems such as machines and building of structures have been organized based on the human intensions as external commands. However, many natural systems become structured by their own internal processes, these are self-organizing systems and the emergence of order within them is a complex phenomenon that intrigues scientists of all disciplines.

The term “self-organization” used in different ways with different meanings and definitions, suggested to build the self-organizing systems as in Cybernetics (Heylighen and Joslyn, 2001), Biology (Camazine et al., 2003) (Feltz et al., 2006), Computer Science (Heylighen and Gershenson, 2003) (Mamei et al., 2006) (Kohonen et al., 2000), Complexity (Schweitzer, 1997) and Information Theory (Shalizi, 2001).
Nowadays computing systems, are open systems, evolves in a dynamic and complex environment. They are designed as sets of interacting components, highly distributed both conceptually and physically. The growing complexity of these systems and their large-scale distribution make use of traditional approaches based on hierarchical, functional and decomposition without centralized control. Increasingly, a real need for new paradigms, mechanisms and techniques allowing these systems with the capacity to autonomously manage their functioning and evolution, is expressed.

Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system (i.e.) interactions internal to the system, without intervention by external directing influences. A self-organizing system functions without central control, and through contextual local interactions between its components. Components achieve a simple task individually, but a complex collective behavior emerges from their mutual interactions. Such a system modifies its structure and functionality to adapt changes as per requirements and to the environment based on previous experience (Camazine et al., 2001).

Pattern is an emergent property of the system, rather than a property imposed on the system by an external ordering influence. Emergent properties are features which arise from unexpected interactions among the system components.

An emergent property cannot be understood by examining the properties of the system’s components in isolation. It is important to point out that system components do not necessarily have to interact directly. Individuals may interact indirectly if the behavior of an individual modifies the environment and thus affects the behavior of other individuals. By tuning the rules of interaction, selection shapes the pattern that is formed and thus the products of group activity can be adaptive.

2.2.1 Self-organizing Systems

Self-organizing system is a physical, chemical or biological system that takes a form that is not imposed by an external directing influence, i.e. without a central control.
Self-organizing systems are designed as sets of similar lower-level components interacting both conceptually and physically in order to obtain the pattern at the global level of a system. The challenge is to understand how the lower-level components interact to produce a common pattern. These components may interact directly or indirectly, depending on the modification of the behavior of a component affecting the behavior of the other components (Camazine et al., 2001), (Heylighen, 2002), (Seeley, 2002).

Many natural systems become structured by their own internal processes. Any system that takes a form that is not imposed from outside can be said to be self-organized. Pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system (i.e.) interactions internal to the system, without intervention by external directing influences. Components achieve a simple task individually, but a complex behavior collectively emerges from their mutual interactions (Camazine et al., 2001).

The two basic modes of interaction among the components of self-organizing systems are Positive feedback and Negative feedback.

*Positive feedback:* As an example, the hormone oxytocin stimulates muscular contractions of the uterus, which in turn stimulate the release of more oxytocin (Soper et al., 1997).

![Positive feedback diagram](image)
**Negative feedback**: As an example, a rise in blood sugar leads to the production of insulin. As insulin level rise, glucose is removed from the blood (Soper et al., 1997).

![Diagram of Negative Feedback]

2.2.2 **Properties of Self-organization**

The correlation or coherence between separate components produced by self-organization defines an ordered configuration. Organization can be defined as the characteristic of being ordered or structured to fulfill a particular function.

**Systemness**

Self-organization takes place in a system, in a coherent whole that has parts, interactions, structural relationships, behavior, state, and a border that delimits it from its environment.

**Information Production**

Self-organizing systems are information-producing systems.
**Dynamic**

Self-organizing systems are characterized by multiplicity of interactions. This emphasizes that the SO systems are dynamic and require continual interactions among the lower-level components.

**Decentralized Control**

Decentralized control refers to a particular “architecture of information flow” in the absence of external control. Individuals in self-organized social groups do not rely on instructions from well-informed individuals. In self-organizing systems, control of the organization is typically distributed over the whole of the system. All parts contribute evenly to the resulting arrangement.

**Complexity**

Self-organization systems are complex systems. The complexity of the system depends on the number of elements, connections between them and the system’s structure.

**Global Order from Local Interactions**

In the self-organized state, all segments of the system are strongly correlated. Global organization emerges from the elements of the system that interacts locally.

**Control Parameters**

Self-organizing system consists of a set of parameters which influence the state and behavior of the system.

**Critical Values**

In a self-organizing system if certain critical values of the control parameters are reached, structural change takes place and the system enters a phase of instability and criticality.
**Fluctuation and Intensification**

Small disturbances from inside the system intensify themselves and initiate the formation of order.

**Robustness and Resilience**

Self-organizing systems are relatively insensitive to perturbations or errors, and have a strong capacity to restore by themselves. One reason for the fault-tolerance is the redundant, distributed organization: the non-damaged regions can usually make up for the damaged ones. Outside a critical phase, the structure of the system is relatively stable concerning local disturbances and change of boundary conditions. Another reason for the intrinsic robustness is that self-organization thrives on randomness, fluctuations or “noise”.

**Stigmergy**

Stigmergy refers to the mechanism by which members of the group coordinate their activity. The stimulation of the members by the very performances they have achieved is a significant one inducing accurate and adaptable responses. Self-organizing system shows a stigmergic information flow.

**Openness**

Self-organization can only take place if the system imports entropy that is transformed; as a result, energy is exported or dissipated.

**Dense Hierarchies**

In dense hierarchies in addition to stigmergic information flow, there is also an idea that direct communication between individuals or between groups and individuals is important. ‘The hierarchy is also highly connected or ‘dense’ in the sense that each individual member is likely to communicate with any other.
Non-linearity and Feedback

In self-organizing systems, the relation between cause and effect is much less straightforward: small causes can have large effects, and large causes can have small effects. In a critical phase, causes and effects cannot be mapped linearly; similar causes can have different effects and different causes can have similar effects, whereas large changes can also result in only small effects.

Feedback Loops, Circular Causality

Feedback loops occur within a self-organization system; circular causality involves a number of processes:

\[ p_1, p_2 \ldots p_n \ (n \geq 1), \text{and } p_1 \text{ results in } p_2, p_2 \text{ in } p_3 \ldots p_n \rightarrow 1 \text{ in } p_n \text{ and } p_n \in p_1. \]

Bifurcation

Bifurcation means a phase transition from stability to instability or a sudden transition from one pattern to another following even a small change in parameter of the system. Small adjustments in such parameters can induce large changes in the state of the system. Once a fluctuation intensifies itself, the system enters a critical phase where its development is relatively open, certain possible paths of development emerge, and the system has to make a choice. This means dialectic of necessity and chance.

Selection

In a critical phase that can also be called the point of bifurcation, a selection is made between one of the several alternative paths of development.

Inner Conditionality

Self-organizing systems are influenced by their inner conditions and the boundary conditions and the boundary conditions from their environment.
**Relative Chance**

In self-organizing systems, there is dialectic of chance and necessity; certain aspects are determined, whereas others are relatively open and subject to chance.

**Symmetry Breaking**

The self-organizing system initially treats all configurations equally, but then it expresses a preference for one possibility. However, there are no objective criteria for preferring one stable configuration over another. The system makes an arbitrary decision to change the range of possibilities. This unpredictability creates the real novelty. The emerging structures have less symmetry than the foundational laws of the system.

**Organizational Closure and Emergence**

Organizational closure turns a collection of interacting elements into an individual, coherent whole. This whole has properties that arise out of its organization, and that cannot be reduced to the properties of its elements. Such properties are called emergent. Emergence refers to a process by which a system of interacting subunits acquires qualitatively new properties that cannot be fully predicted and cannot be found in the qualities of the components. This principle is also called order from chaos or order through fluctuation. The different types and aspects of emergence are discussed in (Fuchs, 2008).

These system-level properties arise unexpectedly from non-linear interactions among the system’s components. In the context of a dynamic system, the emergent pattern or property is called an attractor of the system. Under a particular set of initial conditions and parameter values, an attractor is the state toward which the system converges over time (Camazine et al., 2001), (Heylighen, 2002), (Kelso, 1995).
2.2.3 Self-organizing Genetic Algorithm

The most common property of the self-organizing system, emergence is considered here. If a system that has higher-level properties at the level of elements then their product of interaction is the global solution of the problem.

If a system is said to be self-organizing, then the elements of the system interact in order to achieve dynamically a global function or behavior (Gershenson, 2006). Otherwise a system is said to be robust when it undergoes any adaptation, modify itself to fit better within the system. Robustness will allow the system to withstand any perturbations in the internal structure without losing its function. Thus, GA being a robust system, with the possibility of local interaction of the components to emerge out globally is chosen for applying the self-organization concept to solve the given problem.

The aim of the self-organizing genetic algorithm is to create an automated computer program that solves the problem with little or no information from the user. Hence, the number of external parameters is reduced by self-configuring the parameters by the algorithm itself. The following section explains the parameters of the GA and the method of self-organizing the parameters.

2.2.4 Mapping of Self-organization Properties to GA

The possible ways of mapping SO properties to GA are:

*Systemness and Complexity*

SOGA applied to a particular problem is a complex system generating an optimal solution by various operations.

*Control Parameters*

Control parameters of SOGA are chromosome length, fitness value, crossover/mutation rate and its range, population size and betterment of the results.
**Information Production**

During execution of SOGA, various component information like chromosomes, its fitness value, and optimal crossover/ mutation rate are produced by various interactions of the components.

**Dynamic and Global Order from Local Interactions**

In SOGA, value of a particular parameter is decided by a dynamic interaction among various other parameters, for example self-organizing selection of suitable crossover point depends on the chromosome length, initial rate and number of input sequences. The process continues for a number of generations until the termination condition is satisfied.

**Decentralized Control**

When SOGA is applied to a problem, almost all or most of the parameter values are assigned in a self-organizing manner without an external control.

**Critical Values, Fluctuation and Intensification**

Crossover/ mutation rate can be made to increase cyclically when there is no betterment of the fitness value. On reaching the upper limit of the rate the process terminates. These are critical values deciding the number of generation and hence termination of GA.

**Robustness and Resilience**

SOGA randomly generates a population of solutions which are modified by various operators. In various generations, the initial population is modified and the best or top scoring chromosomes will be selected. Saving the best solution after each process ensures that the solution at the end is best of all generations.
**Stigmergy and Dense Heterarchy**

With self-organizing mutation, rate will be increased cyclically when fitness value converges. In this process, increase in fitness rate depends on the convergence of fitness value in the previous generation.

**Openness**

SOGA is an open system that accepts the changes in previous steps and modifies accordingly and carry it to next steps.

**Non-linearity and Feedback and Circular Casualty**

The fitness value is critical in assigning the crossover/ mutation rates between the generations. Even a small increase in fitness value can help in retaining the same rate. In GA, chromosomes from initial population undergo crossover, resulting population undergoes mutation and this cyclic process continues till termination.

**Bifurcation and Selection**

With a self-organizing population size, fitness value convergence act as the point of bifurcation. At convergence, population size is increased which implies selection.

**Inner Conditionality**

The parameters of SOGA are assigned based on various conditions and interactions of the components.

**Relative Chance**

In some cases SOGA parameters like initial population size, selection/ crossover/ mutation rates are determined whereas, chromosome length, optimal rates and number of generations are assigned in a self-organizing manner during execution.
Symmetry Breaking, Organizational Closure and Emergence

SO parameter values may change in any generation based on the input and significant improvements in result. This unpredictability creates the real novelty (Camazine et al., 2001), (Heylighen et al., 2002), (Kelso, 1995), (Fuchs, 2008).

2.2.5 Methods of Self-organizing GA Parameters

The various methods of self-organizing GA parameters are explained below:

Self-organizing Encoding

In GA, the chromosomes in the population are usually encoded as fixed-length strings. In SOGA, the length of the chromosome can be made to change adaptively based on the problem (Wu et al.)

Self-organizing on Population Size

Population size can be made to change adaptively based on the problem.

- Population size can be self-organized by generating both small and large populations. Fitness of each of the chromosomes is calculated. If the average fitness of the larger population is higher than the smaller then the program continues with the larger population, if not with the smaller population.

- Each time at convergence population size is doubled till it reaches an optimal upper limit (Harik, 1999).

Self-organizing Number of Generations

In GA, the algorithm terminates with a condition of a specified number of generations or on reaching a satisfactory fitness level or at convergence (no further increase in the fitness score). If the termination of the algorithm is due to a maximum number of generations, an optimum solution may not have been reached. Hence, it is necessary to self-organize the number of generations based on the problem.
**Self-organizing Selection Operator**

In GA, the choice of the selection operator is usually one or combination of more than one operator. In SOGA, certain conditions are defined to choose the appropriate selection operator for a particular problem, for example based on the average fitness of the generated chromosome.

**Self-organizing Crossover/ Mutation Operator**

- The choice of the operator can be self-organized by defining conditions based on which the appropriate operator or rate is chosen.
- Crossover/ Mutation operation is performed with a number of methods and based on the average fitness of the resulting chromosomes, an appropriate method is chosen (Hong et al., 2002).
- The algorithm can be executed initially with a minimum optimal crossover/ mutation rate. At each point of convergence, the rate can be increased cyclically till it reaches the optimal upper limit (Zhang et al., 2009) (Bao-Juan et al., 2008).
- The crossover/ mutation rates adapted initially from a high rate to a minimum optimal rate (Breukelaar et al., 2008).
- Chromosomes corresponding to the larger and smaller crossover/ mutation rate are generated. In addition, chromosomes corresponding to the current value obtained by increase or decrease in the rate are generated. The chromosome with higher fitness is chosen (Thierens et al., 2002).

GA can also be self-organized by implementing a conditional increase in the rate of both crossover and mutation until the corresponding optimal upper limit is reached.
Figure 2.3 Mapping of Self-organization to Genetic Algorithm

2.3 MULTIPLE SEQUENCE ALIGNMENT

Multiple Sequence Alignment (MSA) is identified as one of the challenging tasks in Bioinformatics belongs to a class of hard optimization problems called combinatorial problems (Thompson, 2003). The simultaneous alignment of three or more nucleotide/ amino acid sequences is one of the common tasks in the field of Bioinformatics. Multiple sequence alignment is a pre-processing tool in the subsequent analyses of protein families. The importance of these methods continues to increase with the exponential growth of sequence databases.
2.3.1 Introduction

The main problem in MSA is its exponential complexity with the considered input data set. These alignments may be used to identify profiles or hidden models that may be used to acquire knowledge for distantly related members of the family sequences, newly discovered sequences, and existing sequence databases.

Molecular sequences of DNA, RNA or protein are composed of several kinds of elements. DNA and RNA sequences are composed of four kinds of basic symbols (nucleotides), and protein is composed of 20 (amino acids).

Example for Alignment Elucidation

Consider an input of four sequences, each contains 17 nucleotides length and of total length 68,

\[
\begin{align*}
&\text{T} & \text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{c} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{t} & \text{A} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{c} & \text{c} & \text{t} & \text{c} & \text{c} & \text{T} & \text{G} & \text{A} & \text{C} & \text{C} & \text{T} & \text{G} & \text{A} & \text{C} & \text{T} \\
&\text{T} & \text{t} & \text{c} & \text{c} & \text{t} & \text{c} & \text{c} & \text{T} & \text{G} & \text{A} & \text{C} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{C} & \text{T}
\end{align*}
\]

Some of the possible alignments include

\[
\begin{align*}
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C}
\end{align*}
\]

Exact Match: 10

\[
\begin{align*}
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C}
\end{align*}
\]

Exact Match: 11

\[
\begin{align*}
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} \\
&\text{T} & \text{t} & \text{t} & \text{C} & \text{c} & \text{t} & \text{c} & \text{C} & \text{t} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C} & \text{T} & \text{G} & \text{A} & \text{T} & \text{C}
\end{align*}
\]

Figure 2.4 An Illustration of MSA
**Time complexity of MSA is calculated as given below:**

Time Complexity: \((\text{Number of sequences})^{(\text{Average length of all sequences})}\)

<table>
<thead>
<tr>
<th>Number of sequences</th>
<th>= 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average length of all sequences</td>
<td>= 22</td>
</tr>
<tr>
<td>Then possible alignment is</td>
<td>= 13162170384267136</td>
</tr>
<tr>
<td>Number of sequences</td>
<td>= 6</td>
</tr>
<tr>
<td>Average length of all sequences</td>
<td>= 20</td>
</tr>
<tr>
<td>Then possible alignment is</td>
<td>= 3656158440062976</td>
</tr>
</tbody>
</table>

2.3.2 **Technical Constraints of MSA**

Multiple Sequence Alignment is a complicated problem. It stands at the crossroad of three distinct technical difficulties:

- The choice of the sequences
- The choice of an objective function (i.e., a comparison model)
- The optimization of that function

**The choice of sequences**

The global MSA methods make sense if they are assumed to be dealing with a set of homologous sequences, i.e., sequences sharing a common ancestor. When the condition is not met, consider the use of local MSA methods. In any case, always be aware that given inappropriate sequences, most multiple alignment tools will nonetheless produce an alignment. It will be the responsibility of the biologist to realize that this alignment is meaningless.

**The choice of an objective function**

Objective Function (OF) defines the mathematical objective of the search. Given a perfect function, the mathematically optimal alignment will also be biologically optimal. Yet, the function defines a mathematical optimum and rarely has an argument that this optimum will also be biologically optimal.
Defining a proper objective function is a highly non-trivial task and an active research field of its own right. In theory, an OF should incorporate everything that is known about the sequences, including their structure, function and evolutionary history. This information is rarely at hand and is hard to use, so it is usually replaced with sequence similarity. Thus, a very simple general function is often used with affine gap penalties. Insertions or deletions are scored using affine gap penalties that penalize a gap once for opening and then proportionally to its length.

This penalty scheme is a major source of concern because it requires two parameters:

- The gap opening
- The gap extension penalty.

The adequate values can only be set empirically and may vary from one set of sequences to the next.

2.3.3 Computational Complexity of MSA

The third problem associated with MSAs is computational complexity. Considering their most obvious properties, it is convenient to classify existing algorithms in three main categories: exact, progressive and iterative.

*Exact algorithms* are high-quality heuristics that deliver an alignment usually very close to optimality sometimes but not always within well-defined boundaries. They can only handle a small number of sequences (< 20) and are limited to the sums-of-pairs objective function.

*Progressive alignments* are by far the most widely used method. They depend on a progressive assembly of the multiple alignments where sequences or alignments are added one by one so that never more than two sequences (or multiple alignments) are simultaneously aligned using dynamic programming. This approach has the great advantage of speed and simplicity combined with reasonable sensitivity, even if it is by nature a heuristic that does not guarantee any level of optimization.
Iterative alignment methods depend on algorithms ability to produce an alignment and to refine it through a series of cycles (iterations) until no more improvements can be made. Iterative methods can be deterministic or stochastic, depending on the strategy used to improve the alignment. The simplest iterative strategies are deterministic. It involves extracting sequences one by one from a multiple alignment and realigning them to the remaining sequences, some of these methods can even be a mixture of progressive and iterative strategies. The procedure is terminated when no more improvement can be made (convergence). Stochastic iterative methods include HMM training, simulated annealing and genetic algorithms. The main advantage is to allow for a good conceptual separation between optimization processes and objective function.

The most widely used approaches to MSA implements a heuristic search that builds the result by combining a pair-wise alignment, like ClustalW (Thompson et al., 1994) and T-Coffee (Notredame,2000). Its main disadvantage is that when the errors are made at any step, the error will be propagated to the final result. This is a critical problem if more sequences are added to the alignment process. One approach in addressing this problem is to use an extension of dynamic programming for simultaneously aligning multiple sequences. These algorithms often have high-quality solutions, as compared to a progressive approach. The disadvantage of those algorithms is the complexity increase, additional running time and memory requirements. It allows their use in problems with a limited number of sequences. Other iterative approaches include simulated annealing, genetic algorithms and evolutionary programming. Simulated annealing can be slow, but woks well to improve alignment.

Evolutionary programming works well with highly similar blocks. Recently, genetic algorithms with local searching have been considered as a good solution for this type of problem to avoid local optimization. The local search identifies a better applicant before the next iteration. (Ray et al., 2005) The genetic algorithm is applied with bit matrices, but may cause degeneracy in search performance, with poorly designed operators. To overcome this problem the concept of self-organization is considered to automate the parameter settings of GA. This
concept can be applied for an application in any domain which facilitates a non-specialist to work without any difficulties (Notredame, 2002).

2.3.4 Applications of MSA

Multiple sequence alignment allows comparison of the structural relationships between sequences by simultaneously aligning multiple sequences and constructing connections between the elements in different sequences (Edgar et al., 2006). The widely used applications of MSA include:

- Conserved sequence/pattern recognition (Bailey et al., 2006).
- Profile analysis (Gribskov et al., 1987).
- Homology modeling of protein structures.
- Secondary structure prediction (Simossis et al., 2005).
- Point mutations and insertion or deletion events detection (Abd El-Aziz et al., 2010).
- Comparison of sequences in different aspects like identity, similarity, and homology (Das et al., 2004).
- Protein family identification.
- Functionally important sites, such as binding sites, active sites, or sites corresponding to other key function identification, by locating conserved domains (Erik Lindahl et al., 2000).
2.4 ENHANCEMENT MECHANISMS ON SELF-ORGANIZING GENETIC ALGORITHM

Despite simplicity of genetic algorithm, the working principle remains the same for any problem depicting its robustness which needs careful selection of parameter settings. Choosing unsuitable parameters and methods might result into longer program runs or even bad optimization results.

It is evident from the literature, theory of GA seems to be easy but in practice it is tedious without expertise in GA techniques. To practice GA, it is very important to understand the complexity of interaction of the operators and its parameters. Also, appropriate parameter settings of GA for a particular problem may not be suitable for others. Number of research work towards variant of GAs (Choi et al., 2000), (Vasconcelos, 2001) and modified version of basic operators implemented in different manner gave the way to variety of GA operators (Potts et al., 1994), (Sareni et al., 1998). The method of applying the operators affect the success of a genetic algorithm and the associated effects can vary according to the type of problem. However, there is no general theory for the choice of operator or setting of parameters.

This difficulty was taken up by many researchers and suggested different methodologies for adaptive or self-adaptive mechanism. Previous research works on adaptive mechanism is for manual tuning of parameters on static environment. Investigation on this literature requires considerable time and effort and also beyond the scope of this research.

There have been variants of GA, which attempted to overcome the above said difficulties and recommended various approaches that are self-adaptive in nature. These approaches adapt the parameters of GA with the help of feedback. However, the solution is not achieved through the internal behavior of the parameters when GA progresses i.e., within the iterations.
The understanding of interaction between the parameters is very important in getting the required information in order to automate the right choice of parameter setting during GA run.

Yet another mechanism based on self-organizing concept was found to be more effective to self-configure the parameters efficiently thereby improving the performance.

Though the concept of self-organization is very promising to solve problems of complex nature, it has not been fully attempted in the interdisciplinary research.

2.4.1 Self-organizing Mechanisms to Make GA Simple

It is apparent that there is no general theory to apply self-organization to solve problems. Many recent researches in various disciplines have employed self-organizing mechanism to solve complex problems in a simple manner; general (Heylighen, 2003b), (Di Marzo Serugendo et al., 2004); (Zambonelli and Rana et al., 2005), software engineering (Wooldridge et al., 2000), (Zambonelli et al., 2003), biology (Camazine et al., 2003) (Feltz et al., 2006), mathematics (Lendaris et al., 1964) and others (Skar and Coveney, 2003).

2.4.2 Self-organizing Mechanisms to Enhance the Performance of GA

Omid Khayat et al., (2009) proposes self-organizing fuzzy neural network based on GA and PSO (SOFNNGAPSO) in which the parameters of consequent parts were obtained using the error function and the parameters of premise parts in an iterative process. In the second stage, an optimization process based on GA and PSO was used to evaluate the structure of SOFNNGAPSO with the objective of adjusting the parameters of premise parts and consequent parts. Initializing the parameters of consequent parts randomly, updating them in the parameter- dependent iterative process and using GA and PSO in the optimization phase which are generally slow to reach the global optimum have made the proposed algorithm dull.
Adrian Agoginoen (2004) discusses a genetic algorithm that usually decomposes the task by encoding a chromosome over multiple genes. It faces the difficult credit assignment problem of evaluating how a single allele in a chromosome contributes to the full solution. Accurately measuring the fitness of an individual allele in such a large search space requires many trials and may result in stagnation. Their work proposes self-organizing multi-agent system where each agent focuses on simpler problem of producing a single allele. Instead of utilizing recombination to search for a good chromosome, the multi-agent approach has a large advantage that each agent can use its own evaluation function to evaluate a single allele independently.

Wakao Sasaki et al., (1999) discusses the optical neural network system that includes genetic algorithms along with back propagation schemes, which make it possible that the system avoids the local minimum problems and make the learning processes faster and better. This system makes use of the genetic algorithms and the back propagation at the same time to derive selectively the merits from these two methods.

Il-Kwon Jeong and Ju-Jang Lee, (1997) introduces the use of a genetic algorithm to discover the rules that govern emergent cooperative behavior. A self-organizing genetic algorithm was applied to automate the discovery of rules for multi-agents playing soccer. Investigating how relatively simple agents can adaptively learn to solve a complex problem by self-organizing genetic algorithm to find a rule table for agents.

Il-Kwon Jeong et al., (1998) presents a self-organizing genetic algorithm (SOGA) as a multimodal function optimizer which sets GA parameters such as population size, crossover probability, and mutation probability adaptively during the execution of a genetic algorithm. In SOGA, GA parameters change according to the fitness of individuals. A self-organizing genetic algorithm (SOGA) was designed to prevent the premature convergence and to sustain the convergence capacity of GA. Using adaptive population size, the execution time of the algorithm can be significantly lowered.
Anna et al., (1999) discussed a novel method for organizing a randomly distributed set of sensors. The system automatically generates the optimization problems depending on the events happening in the environment. The optimization problem is solved by using genetic algorithms. This makes it necessary to develop an adaptive genetic algorithm to speed up the convergence of the algorithm when the number of targets increases. Alternatively, distributed genetic algorithms can be developed to reduce the computation time so that the self-organization algorithm is applicable in real time.

Kusum Deep, (2008) presents a C-SOMGA for constrained optimization which is based on the features of genetic algorithm and self-organizing migrating algorithm (SOMA), aimed to use a penalty free constraint handling selection. C-SOMGA is not only easy to implement but can also provide feasible and better solutions in less number of function evaluations. It works with a very low population size, hence it is inexpensive in terms of function evaluations.

Zhang Jinhua et al., (2009) proposes a self-organizing genetic algorithm (SOGA) constructed with a dominant selection operator enhancing the action of dominant individuals and a cyclical mutation operator that periodically varies the mutation probability during execution. A nonlinear regression analysis demonstrates that the algorithm is able to avoid premature convergence with a higher convergence speed with self-organization properties.

Renato Tinos et al., (2007) RIGA, the new individuals created in the current chain reaction are preserved in a sub-population. Two strategies for replacement of individuals in the current population is carried (i) some individuals with randomly generated individuals and (ii) lowest fitness individuals with random immigrants. To self-organize RIGA, (SORIGA) two modifications are made in the replacement function and the selection scheme.

Hagen et al., (2005) suggested a single-objective algorithm to improve the duration of the process execution. Further it addresses the presence of multiple evaluation criteria by a Multi-Objective Genetic Algorithm (MOGA) to find the acceptable
Pareto solutions as ‘Trade-offs’. Secondly MSGA uses many sexes as optimization criteria and each individual is evaluated according to the optimization criterion related to its sex; a multi-parent crossover is applied and a set of non-dominated solutions is updated during execution.

Manuj Darbari et al., (2011) proposes a dynamic control strategy by combining the Neuro-genetic self-organizing approach on Petrints. The use of genetic learning method performs rule discovery of larger system with rules fed into a conventional system. The main idea to use genetic algorithms with neural network is to search for the appropriate weight change in neural network which optimizes the learning rate of the entire network. A good Genetic Algorithm can significantly reduce neuro-petrinet in aligning with the traffic conditions; otherwise it is a very complex issue.

Wu et al. presents an algorithm combining the genetic algorithm and self-organizing neural network to solve MSA. This approach demonstrates improved performance in long DNA and RNA data sets exhibiting small similarity. Self-organizing neural network as local optimization like classification is embedded into genetic algorithm to keep away from local optima.

Renato Tin’os and Shengxiang Yang, (2005) proposed a genetic algorithm with random immigrants for dynamic optimization problems, where the worst individuals and its neighbours are replaced in every generation. In this GA, the individuals interact with each other and, when their fitness is close, as in the case where the diversity level is low, one single replacement can affect a large number of individuals. This simple approach can make the system to a kind of self-organization behavior (SOC), which is useful to maintain the diversity of the population in dynamic environments and allows the GA to escape from local optima when the problem changes.

Rahul Khanna et al., (2006) proposed a novel approach to design a self-organizing network based on genetic algorithm to adaptively create various components such as cluster members, cluster-heads, and next-cluster. These components are then used to
evaluate the average fitness of the system based on the sequence of communication links towards the sink.

It can be shown that the periodic run of a genetic algorithm will help conserve the overall energy of the system with maximum operability. As it is seen the individual components tend towards maximizing their fitness with the passing generations in a uniform manner. This shows that the goal of maximizing the system fitness along with individual component fitness can be achieved with a considerably reduced complexity. The algorithm also prevents the over optimization of an individual fitness component at the cost of other components.

Naoyuki Kubota, (1996) suggested a virus-evolutionary genetic algorithm (VEGA) based on virus theory of evolution composed of a host population of candidate solutions and a virus population of substrings of host individuals. A reverse transcription operator overwrites a virus string on a host's string and a transduction operator generating a new virus from a host string. The virus infection operators enable increase of effective schemata with reverse transcription and transduction. The reverse transcription plays roles of crossover and selection simultaneously, since the reverse transcription generates new individuals overwriting on host individuals according to the virus infection rate. Further, transduction generates new virus individuals and evolves the virus population. Therefore, co-evolution of the virus population and the host population enables quick solution of the optimization problem.

Vinicius Tragante et al., (2010) proposed a new strategy to control the number of random immigrants in GAs applied to the protein structure prediction problem. Instead of using a fixed number of immigrants per generation, controls the number of new individuals to be inserted in the generation according to a self-organizing process. The random immigrants inserted in every generation maintain, or increase, the diversity of population is advantageous to GAs applied to complex problems like the protein structure prediction. The rate of replaced individuals in the standard random immigrants approach is defined apriori, and has a major influence on the performance of the algorithm.
2.5 OBSERVATIONS

Irrespective of the self-organization approach to realize an adaptive GA, it is observed that majority of the work strive to bring about an improvement to GA by addressing premature convergence or diversity in the population. Several self-organizing GAs that aim to promote diversity were studied. The observations made out of the study are as follows:

- An increase in execution time and additional computational overhead is associated when an attempt is made to increase the possibility of getting better solutions by generating the chromosomes corresponding to dual population size i.e. to both minimum and maximum size and
- for the lower and upper optimal limit in addition to the chromosomes generated for the current crossover/mutation rate.

- With the initialization of parameters of consequent parts randomly, updating them in the parameter-dependent iterative process using GA and PSO in the optimization phase are generally slow to reach the global optimum.

- C-SOMGA works with a very low population size to make it inexpensive in terms of function evaluation. But low population size has the disadvantage of decrease in diversity which may lead to convergence at local optima.

- Many reported research concentrates either on modification of a single parameter or multiple parameters simultaneously. The modifications of the parameters depend mainly on specific representation of the procedure adopted.
  - In general, various parameters like selection, crossover, mutation, and population size are assigned with a fixed value. This may not be suitable for varying input, for example, when a high crossover rate (80%-90%) and a low mutation rate (0.5%-1%) is assigned.
In case of dynamic parameter values at every convergence,

- the values are doubled till it reaches the optimal upper limit.
- the values are increased for a given interval till it reaches the optimal upper limit.

It is generally observed that many algorithms do not consider a sufficient number of benchmark dataset for experimentation and validation.

From the above it is found that only a few research explored the direct combination of self-organization and genetic algorithm. In most of the cases, GA is used only for optimization and other techniques like neural network is used for self-organization.

2.6 SUMMARY

This chapter has tried to bring about the basics of genetic algorithm, their design considerations and a brief report on various genetic operators. The outline description of the self-organization concept and their properties considered for mapping of GA are discussed. Also a detailed survey of the test problem, Multiple Sequence alignment is studied. The previous reported research works on self-organizing genetic algorithm is discussed. An extensive study of different mechanism has been made and some notable instances of previous works are reviewed. There are sufficient instances of positive results to show that it is worthwhile to focus research in this area.