Chapter 1

Introduction

This Chapter introduces the basic concept of an optimization problem. It also explains the major terminologies and tools used in the entire thesis. As a part, the literature is being reviewed in it. Further, it is concluded with the Chapter-wise summary of this thesis.

1.1 OPTIMIZATION

A key word associated with Operations Research is 'Optimization', which deals with the determination of the best course of action amongst the different alternatives available in a decision-making problem. It can be regarded as the process of finding the optimal value (the greatest or the smallest value, as the case may be) of a function (usually called objective function) under a given set of circumstances (often called constraints). Optimization can thus be viewed as an important tool used in the decision making process where one or more of the specified objectives is (are) optimized under a prescribed set of constraints. Depending on the nature of the objective function, constraint functions, existence of number of variables, number of objective functions; the optimization problems are divided into several categories. The objective function which is sought to be optimized may be convex or non convex, linear or nonlinear, fractional or geometric, single objective or multi-objective. Sometimes the explicit mathematical formulation of the function may not be known. The constraints are linear or non linear depending on the linear or non linear nature of the decision/design variables present in the constraint functions. Optimization problems in which constraints are present are called 'Constrained optimization problems', where as an optimization problem having no constraints are called 'Unconstrained optimization problems'. If there is one objective function to be optimized is called 'Single objective optimization problem'. There can also be a situation in which more than one objective function is to be optimized subject to the same set of constraints. Such problems are known as

'Multi-objective optimization problems'. There exist several books on describing the mathematical theory and problem solving approaches for

linear/nonlinear, unconstrained/constrained, single/multi-objective optimization problems (Fiacco and McCormick (1990), Nesterov and Nemirovskii (1994), Horst and Tuy (1996); Bertsekas (1999), Bazara et al. (2006), Ruszczynski (2006)).

Optimization problems arise in almost in every real field of Engineering, Science, Technology, Business Administration, Management, Pharmaceutical sciences etc. Most of the practical design problems like analysis on electrical circuits, design of chemical production plant, the structural design of buildings or bridges, aircraft scheduling can be modeled as an unconstrained or a constrained optimization problems. Sometimes, the existence of highly non-linear constraints and a large number of decision variables make the problem more complex. Hence there is a need of developing an efficient and effective optimization technique.

In general, a nonlinear optimization problem may have one or more local optimal solution, where as in case of linear programming problem every local optimal solution is also its global optimal solution. Particularly, in case of non linear programming problem, locating the global optimal solution is a difficult task. It is because of the existence of a large number of local optima. While the optimization technique tries to find the global solution, there is a chance of trapping in some local optima during the execution of the computational process. Therefore researchers could not derive the guarantee of locating the global solution by some particular computational optimization technique. In the consequent sections the general mathematical formulation of both unconstrained and constrained optimization problems are presented.

1.1.1 Mathematical Formulation

Unconstrained Optimizations

The general mathematical view of an unconstrained optimization problem is

$$\left.\begin{array}{l}\text{Minimize } f(\vec{x}) \\ \\ \text{Subject to } \vec{x} \in F,\ F \subset R^n,\end{array}\right\} \qquad (1.1)$$

where f is a real valued function and F is the set of feasible points. Here above problem is a single objective one where the aim is to find the solution vector $\vec{x} = [x_1, x_2, x_3, x_4, ..., x_n]^T$ that satisfies (1.1).

Constrained Optimization:

The mathematical formulation of a general constrained optimization problem (COP) is given as follows.

$$\left.\begin{array}{l}\textit{Minimize or Maximize } f(\vec{x}) \\ \\ \text{Subject to} \\ g_k(\vec{x}) \le 0, k = 1,2,3,4,...,m \\ h_p(\vec{x}) = 0, p = 1,2,3,4,...,l\end{array}\right\} \qquad (1.2)$$

Here S is the $n-$dimensional search space in R^n bounded by the parametric constraint which is defined as $S = \{\vec{x} \in R^n / l_j \le x_j \le u_j, j = 1,2,3,...,n\}$, where l_j and u_j are the lower bound and upper bound of the j^{th} decision variables, respectively. Again, the Feasible search space F is defined as

$$F = \{\vec{x} \in S / g_k(\vec{x}) \le 0, h_p(\vec{x}) = 0, \quad k = 1,2,..., m, \quad p = 1,2,..., l\}.$$

The aim of solving a constrained optimization problem is to locate the solution vector $\vec{x} = [x_1, x_2, x_3, x_4, ..., x_n]^T \in F \subseteq S \subseteq R^n$ that satisfies the prescribed constraints with optimize the objective function. Solving COPs in the presence of equality constraints makes the problem complex and it is difficult to solve. However each equality constraint can equivalently represented as the combination of two inequality constraints.

It is more difficult to solve a constrained problem as compared to

unconstrained problem in optimization due to the following factors.

i. Reduced feasible region in COPs
ii. Multimodality of the objective function
iii. Presence of equality constraints

Particularly, finding the global optima in constrained optimization problem is a typical task because the presence of constraints causes the location of the minimum to change which has been illustrated in Fig. 1.1. Here F represents the feasible search space.

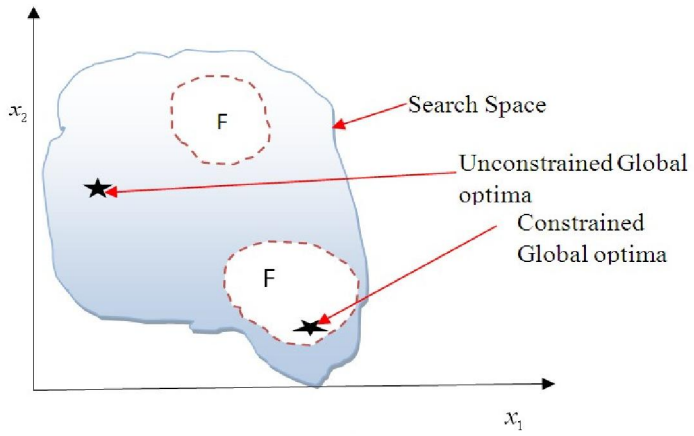

Fig. 1.1: Location of optima in constrained Vs. unconstrained optimization

1.1.2 Some Definitions

Local Optima

A point $x^* \in F$ is said to be a relative minimum point or **local minima** of f over F, if there is an $\in > 0$ such that $f(x) \geq f(x^*)$ for all $x \in F$ within a distance $\in$ of x^* i.e. $(x \in F, |x - x^*| < \in)$. If $f(x) > f(x^*)$ for all $x \in F, x \neq x^*$, within a distance $\in$ of x^*, then x^* is said to be a strict relative minimum point of f over F.

A point $x^* \in F$ is said to be a relative maximum point or **local maxima** of f over F, if there is an $\in > 0$ such that $f(x) \leq f(x^*)$ for all $x \in F$ within a distance $\in$ of x^* i.e. $(x \in F, |x - x^*| < \in)$. If $f(x) < f(x^*)$ for all $x \in F, x \neq x^*$, within a distance $\in$ of x^*, then x^* is said to be a strict relative maximum point of f over F.

Global Optima

A point $x^* \in F$ is said to be **global minimum point** of f over F if $f(x) \geq f(x^*)$ for all $x \in F$. If $f(x) > f(x^*)\ \forall x \in F, x \neq x^*$, then x^* is said to be a strict global minimum point of f over F.

Similarly, a point $x^* \in F$ is said to be **global maximum point** of f over F if $f(x) \leq f(x^*)$ for all $x \in F$. If $f(x) < f(x^*)$ for all $x \in F, x \neq x^*$, then x^* is called as a strict global maximum point of f over F.

1.2 GLOBAL OPTIMIZATION TECHNIQUES

During past few decades, the field of optimization has grown into manifolds. Many researchers have proposed various global optimization techniques to solve various problems in Science, Engineering, Technology and Management. The global optimization techniques can be broadly divided into two categories, viz. exact (or deterministic) methods and heuristic methods (Horst and Pardalos, 1995). A review on these methods is presented below.

1.2.1 Deterministic Methods

Deterministic method exploits the analytical properties of the objective function. They rely on thorough search of the optimization domain. In this section, a brief literature survey has been discussed for both unconstrained and constrained optimization problems. There exist several methods for unconstrained

optimization. They are Trust Region method (Gu and Mo, 2008; Fu and Sun, 2005), Quasi Newton method (Wei et al., 2006a), Conjugate gradient method (Wei et al., 2006b, Fletcher, 1997; Hestens and Stiefel, 1952; Fletcher and Reeves, 1964; Yuan, 2009; Andrei, 2013; Andrei, 2010). Similarly for constrained optimization several methods exist viz. Sequential Quadratic Programming (Xue et al., 2009), Projected Gradient Method (Luenbenger, 1974; Chun-Xia and De-Tong, 2011), Interior Point Method (Zhu, 2003; Fan, 2009). Apart from this there exist several popular deterministic approaches like Branch and Bound algorithms, Lipschitz optimization and Interval analysis, Homotopy continuation methods, etc. Branch and Bound (Borchers and Mitchell, 1994; Leyffer, 2001) technique is applied mostly in case of NP-hard discrete optimization problem. It is performed in two basic stages. In branching stage, the entire problem is spitted into sub-problems and in bounding stages the lower/upper bounds of the objective function value for the sub-problem is calculated and the unpromising sub-problems are eliminated. Interval based global optimization algorithms use Branch and Bound techniques with iterated bisection of the problem domain. These techniques are applied to unconstrained, constrained as well as non smooth optimization. Initially the Interval techniques were established by Moore (1966, 1976), Skelboe (1974), Caprani and Madsen (1979). The Homotopy continuation method (HCM) is applied to diverse problems in the field of engineering and sciences viz. multidisc electronic circuits (Vazquez et al., 2005), non linear control system (Reif et al., 1998), inverse kinematic problems (Wu, 2006).

In real world, we come across highly non-linear, multi-modal functions to be optimized. In some cases the nature of the function is not known. As a result the deterministic/traditional methods fail because they are highly dependent on the mathematical theory of optimization like convexity and concavity of the objective function. However, to solve a complex structured problem having higher dimension, the traditional methods mostly become handicapped. In such cases, heuristic methods are being treated as an alternate

paradigm. These methods make use of probabilistic or stochastic processes to explore the search space. Although heuristic methods do not give absolute guarantee for achieving exact optimal solution, these methods are sometimes preferred over the deterministic method because of their efficiency of providing near optimal solution (very close to the exact solution) and they are applicable to a wider class of problems.

1.2.2 Heuristic methods

These days, quite a good number of heuristic techniques are available in the literature for obtaining the global optimal solution. They are broadly categorized as Evolutionary algorithm, Swarm Intelligence under one umbrella of Meta-heuristic algorithms. Evolutionary algorithm comprises GA, Gene Expression Programming, Genetic Programming, Evolutionary Programming, Evolution Strategy, Differential Evolution, etc. Swarm Intelligence includes Ant Colony Optimization, Particle Swarm Optimization, Honeybee Algorithm, Bacterial Foraging Optimization, Artificial Immune Systems, Artificial Bee Colony, Bacterial-GA foraging, etc. Since the theme of the thesis is entirely based on the hybridization of the above meta-heuristics, so a brief review on them is essential and are discussed in the consequent sub-sections. However, the major differences between deterministic methods and heuristic methods are shown in the following Table 1.1.

Table 1.1: Difference between Tradition and Heuristic methods

Sl. No.	**Traditional Methods**	**Heuristic Methods**
i.	Parameters are directly used to solve the problems	Parameters need to be coded to solve the problems
ii.	Search form a single point	Search from a population of points
iii.	Use derivatives or other subsidiary information	Use pay off (objective function) information
iv.	Use deterministic transition rules	Use probabilistic transition rules

1.2.2.1 Review on Evolutionary Algorithms

Evolutionary Algorithms are heuristic search algorithms follow the mechanisms of evolution and natural genetics. These are the alternatives to traditional optimization technique for location of the global optimal solution. Their search mechanism is based on the probabilistic methods which drag them and lead to the global optimal solution after certain iterations. In general, Evolutionary Algorithms are population based search techniques having almost the similar features in their working principle. These are stochastic global optimization algorithms. Many evolutionary methods have been successfully applied in real life experiments (Box (1957), Friedman (1959), Bledsoe (1961), Bremermann (1962) and Reed et al. (1967), Baricelli (1957), Fraser (1957 a, 1957b), Martin and Cockerham (1960)). However, few of the popular algorithms are Genetic Algorithm, Evolutionary Strategies, Evolutionary Programming, Genetic Programming, etc.

In order to use optimized real value parameters of devices in airfoil, Evolution Strategies (ES) have been introduced by Rechenberg (1965, 1973). It was further developed by Schewefel (1975, 1977). Fogel et al. (1966) developed Evolutionary Programming (EP). Later, Genetic Algorithm (Holland, 1975) is being introduced which mimics the principle of natural selection and natural genetics laid by Charles Darwin's principle of 'survival of the fittest'. Further, Genetic Programming (GP) has been developed in early 1990s (Koza, 1992; Koza et al., 1999) which aims at enabling the automatic generation of computer programs for high-speed and prolific software growth. It unites the biological similes from Darwinians theory of evolution with machine learning approaches of computer science. Therefore, many efficient algorithms are generated which acclimatize themselves for open-ended task. But all the EAs include the following common features.

- Representation of the Strings
- Evaluation of Fitness function

- Selection operator (Survival of the fittest)
- Variation operators i.e. Crossover and Mutation
- Survivor Selection Mechanism

The encoding structures in EAs are different in different versions of EAs. The encoding system differs according to the choice of EAs. GA uses strings over a finite alphabet, Evolutionary Strategies use vector valued function and Genetic Programming (GP) uses the concept of trees.

In this thesis, the mechanism of GA is borrowed and hence its mechanism is discussed in details in Section 1.3.

1.2.2.2 Review of Swarm Intelligence Technique

Swarm Intelligence is the emergent collective intelligence of group of simple agents like group foraging of social insects, nest building of social insects, collective sorting and clustering, division of labour, co-operative transportation (Bonabeau et al., 1999). There are several interesting books on swarm intelligence techniques (Eberhart et al., 2001; Lim et al., 2009; Tan et al., 2010; Panigrahi et al., 2011). Some popular swarm intelligence techniques are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO) and many others. ACO is motivated by ants foraging behavior and is introduced by Marco Dorigo and his colleagues (Dorigo et al., 1996). Ants live in colonies. Their focus is concentrated on the survival of the colony instead of the individual survivance. During the searching procedure, if an ant finds better amount of food source, it takes some of it and returns to the nest. During the return trip, it leaves 'pheromone trail' on the land depending on the amount and value of food source. Other ants are guided by that pheromone trail. The indirect communication among the ants via the pheromone trails is acknowledged as 'Stigmergy' and it facilitates them to discover the shortest path connecting their nest and food sources. A similar population based meta-heuristic called Particle Swarm Optimization (PSO) is proposed by

(Kennneady and Eberhart, 1995). PSO is based on the social behaviors observed in animals or insects like bird flocking, fish schooling and animal herding. In PSO, individual particle represents the potential solution of the search space which seeks an optimal or good enough solution. The particles inform the current positions to the neighboring particles. The position of each particle is adjusted according the particles velocity (i. e. rate of change) to the difference of position between the current best locations and the overall best location of the neighboring particle. As the simulation goes, the swarm focuses more and more on an area of the search space containing high quality solutions.

Bacterial Foraging Optimization (BFO) is proposed as a newer swarm intelligence algorithm by Passsino (2002), which is based on the behavior of E. coli bacteria. In BFO, individual bacterium is intended for affluent areas via chemotaxis. They live in surroundings in which their food source diffuses. So, they can sense and react to its existence. The bacterium rotates its flagella either in clockwise or counterclockwise direction. Its movement will be run if it is moving in predefined direction. Similarly, in case of tumble it moves in random direction. When the bacterium encounters higher concentration of nutrient, runs are of longer duration due to its internal chemistry. In addition, the bacteria secrete chemicals attract to each other. This happens in nutrient rich environments and extra bacteria are recruited to utilize the food source. The self attractant and chemotactic behaviors are forming pattern formation under certain conditions (Badrene and Berg, 1991). Apart from this, many fine points have been explained for E.Coli bacterium and related bacteria via watchful experimentation (Berg and Brown, 1972; Segall et al., 1986; DeRosier, 1998). BFO has been explained in detail in Section 1.4, as it has been taken as in ingredient in the proposed algorithm of the thesis.

1.3 GENETIC ALGORITHM

Genetic Algorithm (GA) is one of the most popular EAs because of its ease implementation and is conducive for noisy environment. These are efficient and

effective population based search techniques inspired by the Darwinian principles of Biological Evolution. GA is developed by John Holland (University of Michigan, USA) in 1975. In USA, GA has become the most popular Evolutionary Computing (EC) technique after the book by Goldberg (1989) entitled 'genetic Algorithm in search Optimization and Machine Learning'. This book explains the concept of genetic search in such a lucid manner that it is accepted by wide community of engineers and scientists. There are several interesting books describing GA (Dejong, 1975; Melanie, 1999; Haupt and Haupt, 2004). There also exist a series of literature surveys and reports regarding the fundamentals of GA (Beasley et al., 1993; Srinivas and Patnaik, 1994; Michalewicz, 1994). It has a wide range of applications in the emerging research fields like VLSI design, Strategy Planning, Machine Learning, Control Engineering, Scheduling, Game Playing, Robotics, Combinatorial Optimization, Design and Signal Processing, etc. The working principle of GA is explained below.

1.3.1 Working Principle of GA

GA works on a population of individuals. These individuals are called strings/ chromosomes which consist of genes. The genes of a string could be a bit, a real number or character depending upon the nature of the decision variables of the problem. The algorithm is based on four operators; viz. Selection, Crossover, Mutation and Elitism. They are applied in series at each cycle of GA. Initially a set of populations $P(t)$ is created for a particular generation t . In each generation, the individual in the population represents a potential solution to the problem at hand. Each individual is evaluated to give some measure of its fitness. According to the fitness of strings, new population is created from the existing population. It is done by Selection operator which is based upon the Darwinian principle of 'survival of the fittest'. The fitter the individuals are, the more chance of being selected into the new population. Once the selection is over, individuals in the existing population undergo stochastic transformation by

way of other genetic operators like Crossover and Mutation. In crossover, the two parent strings are picked up randomly according to the fixed crossover Probability (with a higher rate, in general) and they are crossed to produce two children strings. In this way the best characteristics of the parent chromosome are transferred to the child strings and the better strings are produced.

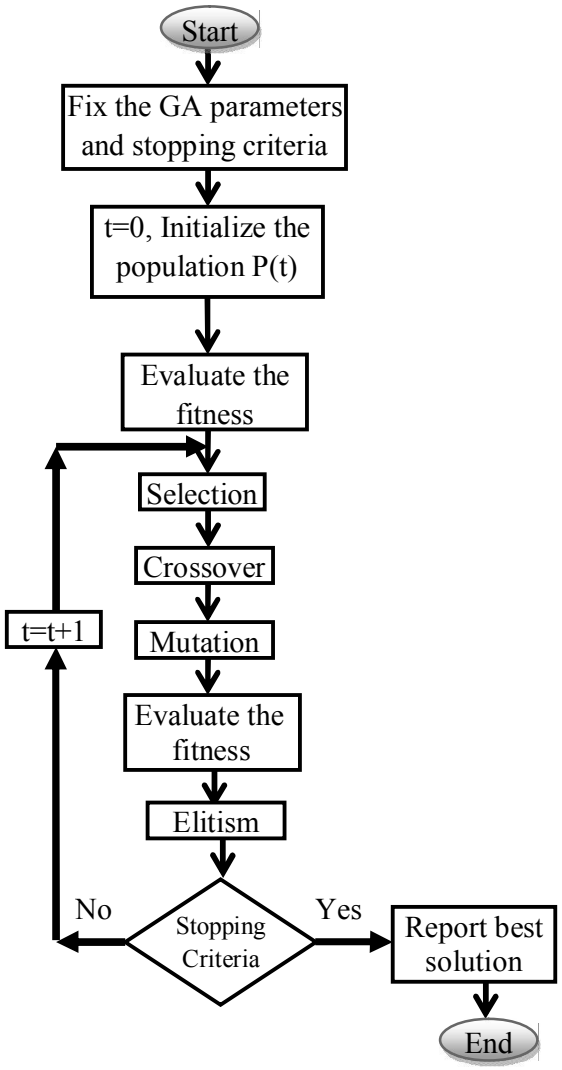

Fig. 1.2: Flow diagram of GA

In mutation, the new individuals are created with some random change in the gene of the chromosome with low mutation probability. After selection,

Crossover and mutation process complete, elitism is applied (which is optional). In elitism, the initial population of that particular iteration and the population created after the mutation process of that iteration are mingled, sorted and the best half of the population is chosen. By this way, the worst strings are die-off and the better strings will only survive to participate for the further process. This completed the generation. GA is allowed to run for several such generations. As the generation goes, the fitness of the individual is expected to improve over the time. It stops with the pre-prescribed stopping criteria. Once the algorithms stops, the best individual will be reported which represents the optimal or near optimal solution to the problem in hand. The flow diagram of GA has been given in Fig. 1.2. Each operator used in this flowchart is explained in detail in the next Section.

1.3.2 GA operators

GA uses Encoding, Selection, Crossover, Mutation and Elitism as the key operators in each of its generations. They are discussed below.

1.3.2.1 Encoding

In GA, initially the decision variables are encoded to another space. Encoding is a mapping from phenotype space to genotype space. There exist several types of encoding viz. Binary encoding, Value encoding, Permutation encoding, Tree encoding and many such other encoding. In general binary encoding is a popular approach in Binary GA. Binary encoding has been used in this thesis. So, it is discussed below.

Binary Encoding:

The simplest and popular approach among the Encoding is Binary Encoding. The decision variables are encoded with 0's and 1's as shown in Fig. 1.3. Binary encoding gives many possible chromosomes even with a small number of alleles. It has been used in the initial research on GA.

Chromosome 1	0	1	1	0	1	0	1	0	1
Chromosome 2	0	0	0	1	1	0	1	1	0

Fig. 1.3: Binary Encoding

1.3.2.2 Selection/ Reproduction

Selection operator follows certain criteria of fitness function evaluation of the strings. It extracts the above average strings and forms multiple copies of best strings from an existing population. It quantifies the optimality of the solution and picks up the string in a probabilistic manner. The better fit individuals are selected to create the mating pool for crossover. The population may converge to suboptimal solution due to strong selection and too slow selection decrease the process of evolution. Selection strategy is analogous to Darwinians principle of the survival of the fittest. Several selection mechanisms exist. Few popular methods are listed below.

- Roulette Wheel Selection
- Boltzmann Selection
- Rank Selection
- Steady state Selection
- Tournament Selection

In this thesis, Tournament Selection operator is only used and hence is discussed below.

Tournament Selection:

Tournament Selection has the advantage that it can be implemented in both non-parallel and parallel architectures. Based on the number of candidates participate at a time, the tournaments can also be classified in a number of ways. However, this thesis uses only the Binary tournament selection. In this case tournaments

are played between two candidate solutions and the better solution is picked up and placed in the mating pool. This is better explained in Fig 1.4. Initially, 1st and 2nd solution are picked up and the string which is more fit is placed in 1st slot of the mating pool. Next, to fill the 2nd slot, the 2nd and 3rd solutions are compared. The process continues till all the slots are filled up. In this way, each solution participates in two tournaments. The best solution will win both times and makes two copies of it in the new population. The worst solution will lose both the tournaments and will be eliminated from the population. Thus, the solution may have zero, one or two copies of a each string in the mating pool**.**

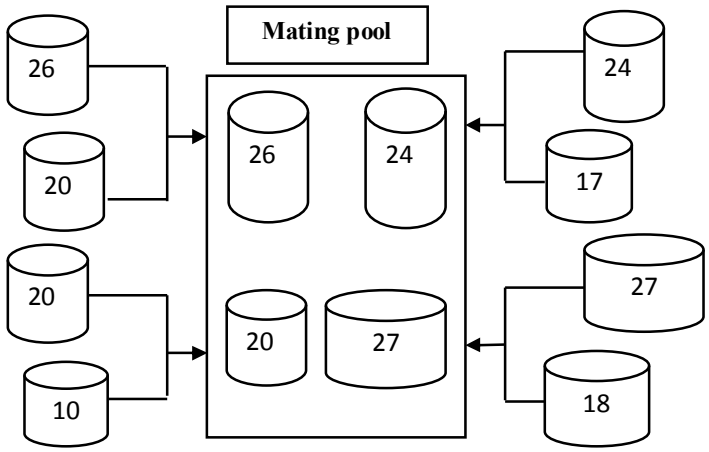

Fig. 1.4: Pictorial representation of binary tournament selection

1. 3.2.3 Crossover

Generally cross over operator is the amalgamation of the two parent strings for producing two children strings. In this case, the exchange of discovered knowledge occurs in the form of genes between the two parent chromosomes. The main idea behind this operator is that it may create two better strings in comparison to the parent strings if it takes best characteristics of the parent strings.

Based on binary encoding, several crossover operators exist as listed below.

- One point crossover
- Two point crossover
- Uniform crossover
- Arithmetic crossover
- Heuristic crossover

Here in this thesis, one point crossover has been used and it is explained below.

One point Crossover:

In this case two parent strings and a crossover site is selected at random. The chromosomes after the crossover site are swapped between the parent strings. As a result, two child strings are produced as shown in Fig. 1.5.

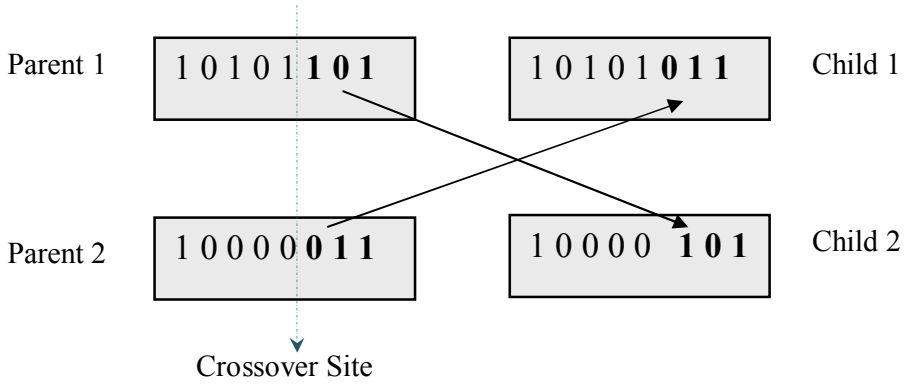

Fig. 1.5: Pictorial representation of one point crossover

1. 3.2.4 Mutation

In genetic search, mutation plays a major role. It maintains the diversity of the population by creating a small perturbation in the population. It prevents the stagnation of population and helps to get rid of trapping in local optima due to the proper exploration of search space. Various types of mutation operators exist. They are Bitwise, Non uniform, Boundary, Uniform, Gaussian mutation operators. Since this thesis uses the binary GA, so the popular bitwise mutation has been picked up for further study and is detailed below.

Bitwise Mutation:

This mutation can be applied only on binary strings. A bit is chosen randomly within the chromosome with a very small mutation probability. The randomly chosen bit is then mutated for creating a new chromosome in the vicinity of the existing chromosome. The value of the chosen gene is inverted randomly from 0 to1 and vice versa. It is shown in Fig. 1.6.

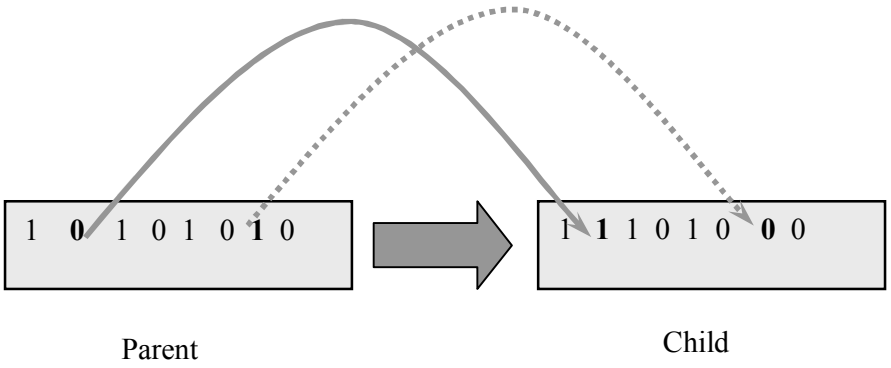

Fig. 1.6: Pictorial representation of Bitwise Mutation

It is clear from the above figure that the 2nd and 7th bits of the chromosome have participated in the mutation process and are mutated to create new chromosomes. The changed bits are bold faced in the figure.

1.3.2.5 Elitism

Elitism operator helps to keep better strings in the population and to eliminate the worst strings. There exist several types of Elitism viz. Complete Elitism, Partial Elitism and Alternate Elitism. But, in this thesis 'Complete Elitism' has been used, which works with the following steps.

(i) Store the population of strings with their fitness function value before and after the GA cycle,

(ii) Mingle the two set of population of strings to make the population size double,

(iii) Arrange the strings in descending (ascending) order of their fitness function value for a minimization (maximization) problem and

(iv) Keep the first half of the population of strings to participate for the next GA cycle.

The diagrammatic representation of the complete elitism is picture in Fig. 1.7.

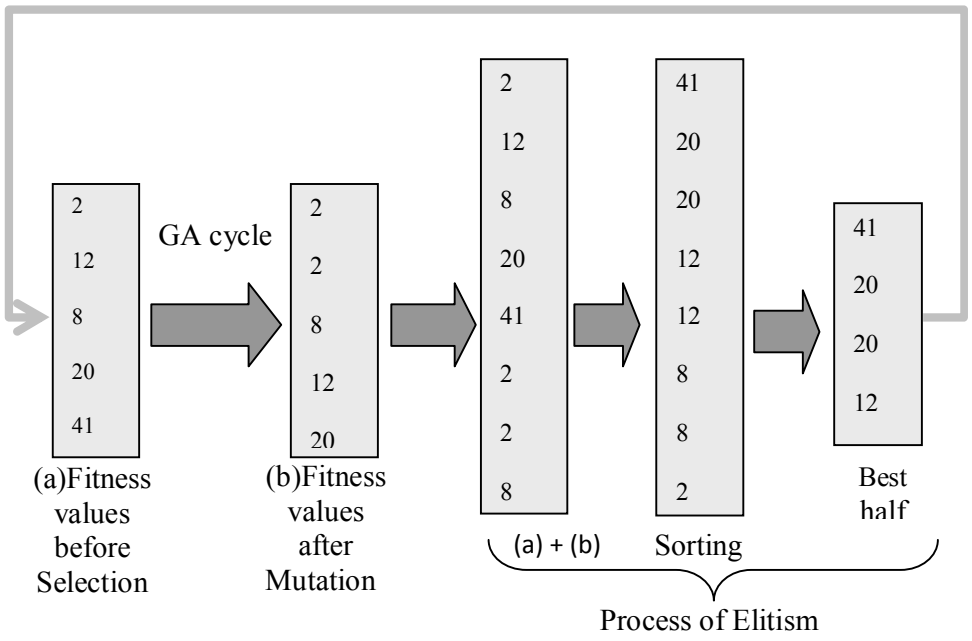

Fig. 1.7: Pictorial representation of Complete Elitism

1.3.3 Applications of GA

Till date GA has been successfully used in many real life applications. In designing Supply Chain Network Management (SCN), the application of GA is quite fruitful, where the multiple and conflicting objectives of this problem are cost, service level and resource utilization. Altiparmak et al. (2006) used the priority based encoding in GA and picked a real life problem, where a plastic production company of Turkey has been considered. Based on the explorative

nature of GA, it has wide gamut of application in the field of Data mining and Machine learning where Data bases are huge search spaces. Srinivasa et al. (2007) applied Self Adaptive Migration Model Genetic Algorithm (SAMGA) for data mining problems. They compared their algorithm with Single Population in Genetic Algorithm (SGA) and Island Model of GA (IGA) methods. At the same time, Sikora and Piramuthu (2007) has great contribution in designing robust GA for solving the data mining problem, where feature selection strategy has been merged with mining strategy. Recent advances in Telecommunication industry has open new arena in the field of Wireless Communication Network. By dint of which, Wireless Asynchronous Transfer Mode (WATM) has gained popularity. Bandwidth optimization in WATM has played an important role in dynamic routing in ATM Network. In this context, Routray (2010) proposed Enhanced Genetic Algorithm (EGA) to solve the problem. He used multi parameter encoding mechanism and compared the result of GA by taking three types of selection operator i.e. Roulette Wheel Selection, Truncation Selection and Tournament Selection. He compared EGA with simple GA. Recently, the attentions of reliability engineers have been focused on reliability optimization of Redundancy Allocation Problem (RAP). It is a mixed integer nonlinear programming problem. Sahoo et al. (2012) used the GA where time to failure of each component is following the Weibull distribution. The problem has been converted into unconstrained optimization problem by using Big-M penalty method. Land optimization problem is a complex combinatorial optimization problem having high computational complexity. As per the laws of Food and Agriculture Organization (FAO), optimization of the land use planning involves several constraints like social and economic conditions, availability of water alternatives, and assessment of potential land. That's why the optimization problem is having lots of alternative solutions. Porta et al. (2013) in their paper proposed parallel GA and they considered land optimization problem of Municipal Land use in Galicia, an autonomous region in North West Spain. Thus GA has shown its high performance in handling a good number of complex problems in recent era.

1.4 BACTERIAL FORAGING OPTIMIZATION (BFO)

Depending on type of animal, environmental characteristics and search strategy the living organism follows different type of foraging behavior. They adopt different approaches and try to locate, handle and ingest food. Recently, various swarm intelligence algorithms came into existence taking into account the foraging behavior of living organism. Amongst several swarm intelligence techniques (as discussed in Section 1.2.2.2), BFO is the recent popular algorithm and it is being used for many practical applications. BFO came into existence in 2002 proposed by K.M. Passino in a seminal paper.

It mimics the foraging behavior of E. coli bacteria found in human intestine. The E. coli bacterium has diameter of 1 μm and length of about 2 μm and under favorable conditions it reproduces in 20 min.

1.4.1 Working Principles of Bacterial Foraging Optimization Algorithm (BFOA)

Let's consider the position of the bacterium as a n dimensional vector in the search domain which is defined as $\theta = [\theta_1, \theta_2, \theta_3, \ldots, \theta_n]^T$. Let θ^i be the location of the i^{th} bacterium.

Let $\theta = \{\theta^i, i = 1, 2, 3, \ldots, S\}$. $\theta^i = [\theta_1^i, \theta_2^i, \theta_3^i, \ldots, \theta_m^i, \ldots, \theta_n^i]^T$, where θ_m^i is m^{th} component of the bacterium. Let cost (objective function) associated with the location of the i^{th} bacterium at the j^{th} chemo tactic step, k^{th} reproduction step and l^{th} elimination-dispersal step is given by $j(i, j, k, l)$.

In the optimization domain, the foraging behavior of bacteria motivates them to climb up the nutrient concentrate. Here the nutrient function/objective function is taken as $J(\theta)$. It indicates different locations of the bacterium. If the nutrient function is strictly negative, the location of the bacterium is nutrient-

rich, strictly positive value indicates the noxious environments. If objective function $J(\theta)=0$, it is in neutral medium. Basically BFOA has 4 operators including Chemo taxis, Swarming, Reproduction, and Elimination-Dispersal. They are discussed below.

(I) Chemo taxis:

The most important computational step in BFO is chemo taxis which were proposed by Bremermann and his co-workers. The behavior of the bacteria in presence of Chemo attractants in concentration gradient is called chemo taxis. In this step, the bacteria decide the proper direction of the movement in search domain. This foraging behavior of E.coli bacteria are achieved by a set of tensile flagella to perform two basic operations such as tumble (moving in different direction) and swim (moving in predefined direction). During chemo taxis step, the bacteria moves towards a nutrient environment in clockwise direction and moves towards a noxious environment in anti clock wise direction. The position of the i^{th} bacterium in $j+1^{th}$ chemo tactic step, k^{th} reproduction state, l^{th} elimination dispersal step is evaluated by the Eq. (1.4) given below. $\theta^{i}(j,k,l)$ is the current location of i^{th} bacterium. Create a direction vector $\Delta(i)=\left[\Delta_1(i),\Delta_2(i),\Delta_3(i),\Delta_4(i),...,\Delta_n(i)\right]$ on the current location, where each element belongs to [-1, 1]. During the bacterial movement as defined by run or tumble, the run length unit $C(i)$ is defined as the step size of the Chemo tactic loop. If move of the bacterium is run, $\Delta(i)$ remain fixed, otherwise $\Delta(i)$ is randomly generated for the tumble movement. After completion of each bacterial movement, the nutrient function $J^{i}(\theta^{i})$ for the i^{th} bacterium is evaluated.

$$\theta^{i}(j+1,k,l)=\theta^{i}(j,k,l)+C(i)\frac{\Delta(i)}{\sqrt{\Delta^{T}(i)\Delta(i)}} \qquad (1.4)$$

(II) Swarming:

It is based on the assumption of reliance upon a bacterium by all other bacteria. Exploiting this idea, the bacteria in a group are in general attracted to a location, where one bacterium reaches to a location of highest nutrient content. The cells release an attractant aspertate if they got stimulations by a high level of succinate and aggregate into groups. Also, they warn each other to maintain a safe distance by releasing repellent from them. There by the bacteria form intricate and stable spatio-temporal patterns (swarming) and are congregated into groups. They move in concentric patterns with high bacterial density. Mathematically the combined effect of repellance and attraction is taken into account and it is represented by the following Eq. (1.6), which is called cell to cell signaling function, where it is added with the actual objective function value $J(i,j,k,l)$ to present a time varying objective function. $J_{cc}\left(\theta,\theta^{i}(j,k,l)\right)$ is the distance function where it is basically the distance of the randomly chosen bacterium under consideration from all other bacteria. The attractant and repelling coefficients of the signaling function are considered judiciously and defined as follows in Eq. (1.5). Let

$d_{attract}$ = The amount of attractant released
$w_{attract}$ = The diffusion rate
$h_{repellant}$ = The magnitude of repelling effect
$w_{repellant}$ = The width of the repelling effect

$$\left.\begin{aligned} &d_{attract} = 0.1,\ \ w_{attract} = 0.2, \\ &h_{repellant} = 0.1,\ \ w_{repellant} = 10 \end{aligned}\right\} \qquad (1.5)$$

$$\left.\begin{aligned} J_{cc}\left(\theta,\theta^{i}(j,k,l)\right) &= \sum_{i=1}^{S} J_{cc}^{i}\left(\theta,\theta^{i}(j,k,l)\right) \\ &= \sum_{i=1}^{S}\left[-d_{aattract}\ \exp\left(-w_{attract}\sum_{m=1}^{n}\left(\theta_m-\theta_m^{i}\right)^2\right)\right] \\ &+ \sum_{i=1}^{S}\left[h_{repellant}\ \exp\left(-w_{repellant}\sum_{m=1}^{n}\left(\theta_m-\theta_m^{i}\right)^2\right)\right] \end{aligned}\right\} \quad (1.6)$$

(III) Reproduction:

Conducive temperature and sufficiency of food inspires the bacteria to increase them in length and they are divided into two parts and form the replica of themselves. This phenomenon is particularly called reproduction. The health of i^{th} the bacterium is J_{health}^{i} given by Eq. (1.7)

$$J_{health}^{i} = \sum_{j=1}^{N_c+1} j\left(i,j,k,l\right) \quad (1.7)$$

(IV) Elimination-Dispersal:

The Elimination–Dispersal phenomenon can be described as follows. The sudden increase of temperatures, the noxious influence of environment, sudden rise of water, sudden attack of animals or any other influences compel the bacteria to disperse into a entirely new region in the search space. Some bacteria are killed also. Ultimately the progress of the chemo taxis step is lowered down and may be destroyed. This forces the premature convergence of the global optimal solution. To simulate this phenomenon, some bacteria are liquidated randomly and some bacteria are produced randomly in the search domain.

Bacterial Foraging Optimization Algorithm (BFOA):

Step 0: Initialization of parameters

(i) S: the number of bacteria to be initialized within the search space.

(ii) n: is the number of parameters to be optimized

(iii) N_s : the Swim step length

(iv) N_c : the number of Chemo tactic steps of the Chemo tactic loop

(v) N_{re} : the total number of reproduction steps

(vi) N_{ed} : the total number of Elimination Dispersal steps

(vii) P_{ed} : the probability of Elimination-Dispersal

(viii) Initialization of $d_{attract}, w_{attract}, h_{repellant}, w_{repellant}$

Step 1: Set $j = k = l = 0$ (1.8)

Step 2: Initialization of the Elimination–Dispersal loop; $l = l + 1$

Step 3: Initialization of the Reproduction loop; $k = k + 1$

Step 4: Chemo taxis loop:

(a) Initialization of the chemotactic loop; $j = j + 1$

(b) $for\ (i = 1, 2, 3, \ldots, S)$, evaluate the objective function

$$J(i, j, k, l) = J(i, j, k, l) + J_{cc}\left(\theta, \theta^i(j, k, l)\right) \quad (1.9)$$

(c) Let $J_{last} = J(i, j, k, l)$ (1.10)

(d) Randomly generate n dimensional vector $\Delta(i)$,

Where, $\Delta(i) = \left[\Delta_1(i), \Delta_2(i), \Delta_3(i), \ldots, \Delta_n(i)\right]$, $\Delta_n(i) \in [-1,\ 1]$

(e) Compute the chemotactic step as Eq. (1.4)

(f) Again compute the objective function value as in Eq. (1.9) in $j+1^{th}$ Chemo tactic step

(g) ***Swim Loop:***

(i) Let m=0;

(ii) While ($m < N_s$)

$m = m+1$

$$If \;\; J(i, j+1, k, l) < J_{last}$$

$$J_{last} = J(i, j+1, k, l) \qquad (1.11)$$

Again compute the objective function value as Eq. (1.9)

Else, let $m = N_s$. Swim loop is ended.

(iii) If $(i < S)$ Go to S***tep 4*** (b) to process the next bacteria, Else go to the ***Step 5***

Step 5: $If \;\; j < N_c$

Go to ***Step 4(a),*** Else go to the ***Step 6.***

Step 6: Reproduction Loop:

(a) Evaluate the Health of the i^{th} bacterium defined as $J_{health}^{\,i}$ as per Eq. (1.7).

(b) Sort the bacteria in ascending order of their health. Select the first half of the bacterial population and copy those strings in the lower half of the population. The weaker bacteria pass away and the bacteria in good health replicate themselves.

(c) *If* $k < N_{re}$, go to ***Step 3***. Else go to ***Step 7.***

Step 7: Elimination and Dispersal:

(a) Eliminate and disperse the bacteria in the search domain for $i = (1, 2, 3, \ldots, S)$ with probability P_{ed} .

(b) *If* $l < N_{ed}$, go to ***Step 2,*** Else stop.

1.4.2 Applications of BFOA

Bacterial Foraging Optimization Algorithm (BFOA) has been successfully applied in the field of Electrical and Control Engineering problems. Tripathy and Mishra (2007) in their paper solved the multi objective problem considering the Unified Power Flow Controller (UPFC) location, series injected voltage, transformer tap positions as the design variables. They solved the combined

Continuous Power Flow (CPF) and Optimal Power Flow (OPF) problem where minimization of power loss and maximization of Voltage Stability Limit (VSL) are considered as the objective functions. The algorithm has been applied on ten machines, 39-bus New England Power System. Mishra and Bhende (2007) proposed BFOA algorithm for optimization of the parameters of Proportional Plus Integral (PI) controller for the Active Power Filter (APF) control and observed that the convergence BFO based PI (BF-PI) controller is efficient as compared to GA based PI controller (GA-PI) for finding the global optimum solution. Stock Market Prediction has wide commercial applications. Majhi et. al. (2009) proposed Adaptive Bacterial Foraging Algorithm (ABFO) and Bacterial Foraging Algorithm for efficient prediction of stock market. They compared their algorithms with standard GA and PSO by minimizing the Mean Square Error (MSE). Now a days, multilevel inverters gained wide interest among the research community. The inverters are used for DC to AC conversion and they are used in many applications such as Flexible AC transmission system (FACTS), High Voltage DC lines (HVDC) and Electrical drives. Salehi et al. (2010) proposed Bacterial Foraging Algorithm (BFA) to eliminate low-order harmonics and optimized stepped voltage of Multilevel Inverter. Robot manipulator optimization has attracted many researchers to design various types of controllers for tracking control of the Robot Manipulator. BFO has also been successfully applied to handle PID controller problems. Aghajarian et al. (2012) has applied BFO in designing Fuzzy controller for Robot Manipulators and compared with PSO. The performance of the controller is evaluated in the joint space and in Cartesian space. Recently, Image Registration is a emerging topic of research in the field of Computer Vision (CV) and Computer Graphics (CG). 3-D representation of real world scenarios in a wide applications area like Robotics, Automobile engineering, 3-D ranges imaging technology. Costin and Bejinariu (2012) adapted BFO for single slice to 3-D Position Emission Tomography (PET) and Computed Tomography (CT) multimodal Image registration. Bermejo et.al (2013) in their paper adapted the foraging behavior of E.Coli bacteria and solved range image registration problems and analyzed the

performance against various image registration methods. In the context of Image Segmentation, DJEROU et al. (2013) have developed novel Bacterial Chemo taxis Multi Objective Algorithm (BCMOA) and introduced in their paper multi-level image threshold, satisfying several segmentation criteria. They applied the algorithm on different types of images and compared their algorithm with NSGA-II and tested the robustness of BCMOA.

1.5 METHOD OF HYBRIDIZATION

During the last three decades, evolutionary computing techniques have grown into manifold. According to No Free Lunch Theorem (NFLT), it is not possible in part of a single state-of-the-art optimization technique to succeed over all the remaining computational techniques in handling all sorts of optimization problems. Depending upon the complexity of the problem concerned, there is a need of hybridized techniques which helps in balancing exploration and exploitation capability over the search space. Based upon this idea, many hybridized techniques came into existence. Since the scope of this thesis is based upon the hybridization of GA with BFO, some reviews has been done upon the successful hybridization of GA and BFO with other nature inspired algorithms. Further, the successful hybridization of GA with BFO and their applications are presented.

1.5.1 Hybridization of GA with other nature inspired algorithms

Due to the increasing popularity of GA, researchers attempted to hybridize GA with other nature inspired algorithms, just to further enhance the efficiency of GA. The hybrid GA also is well applied in different branches of Science and Engineering. Hybrid decision tree/GA has been used to classification problem in Data Mining. Carvelho and Freitas (2004) have exploited the property of decision tree in classifying examples belonging to large disjuncts and have used GA for classification of small disjuncts in data mining. Gene selection in Microarray data problem is an important topic in supervised classification. In

this regard, Huerta et al. (2010) have hybridized Fishers Linear Discriminate Analysis classifier with GA, where the discriminate coefficients of LDA have been embedded with crossover and mutation operator of GA. Software testing has a wide gamut of application because of its importance in increasing the confidence in software reliability. Software testing consumes 50% of the total cost as regard to Software Development. In this respect, generation of test data in oriented paths is another hot topic of research. Gong et al. (2011) has beautifully handled the problem by dividing target paths into several groups. In each sub-optimization domain, GA employs domain based fitness function, thereby increasing the efficiency of generating test data. Machine Learning and Knowledge Discovery is an important issue in Supervised Classification. Sarkar et al. (2012) proposed Decision Tree Genetic Algorithm (DTGA) algorithm to improve the accuracy of prediction over any classification problem regardless of domain, size, dimensionality and class distribution. Decision Tree based rule inducer has been used in the first phase of the classification to produce rules from the training data. GA has been used in final stage to provide more accuracy. The system has been compared with Neural Network, Naïve Bayes, Rule based classification using Rough set theory. They applied the algorithm DTGA on several benchmark problems taken from the Machine Learning repository in the University of California at Irvine. GA has been hybridized with Fuzzy Simulation (Taleizadeh et al., 2013) to solve Multi-Periodic Inventory Control Problem.

1.5.2 Hybridization of BFOA with other nature inspired algorithms

Biswas et al. (2007a) tried to hybridize BFO with PSO and named it as Bacterial Swarm Optimization (BSO). It has been applied to spread spectrum radar poly–phase code design. Biswas et al. (2007b) proposed the hybridization of Chemo tactic step with Differential Evolution, named as Chemo Differential Evolution (CDE). The mutation operator of DE has been decoupled with chemotactic step of BFOA. It has been successfully applied over several benchmark problems. Image Processing is the new area of research where Edge detection is an

essential task. Verma et al. (2011) merged BFO with probabilistic derivative technique of ACO and applied the novel bacterial foraging technology to solve the problem and compared the result with traditional Canny, Edison, Rothwell, Sobel and Susan Edge detector technology. Chatzis and Koukas (2011) proposed hybridization of PSO with BFO and named it Synergetic Bacterial Swarming Optimization (SBSO) and further applied on several benchmark problems. They inserted the swarming dynamics of PSO with BFO. Job Shop Scheduling Problem (JSSP) is an NP hard problem. For a given set of jobs, finding out the optimal scheduling is a static challenge in JSSP. Shivkumar and Amudha (2012) have hybridized BFO with Harmony Search Algorithm (HSA) and designed Harmony Bacterial Swarm Algorithm (HBSA) and applied on JSSP problem. In the field of biomedical research, Medical Image Registration is the process of finding the optimal spatial transformation. George and Kanan (2012) in their paper hybridized Markov Random Field (MRF) with BFOA for brain Magnetic Resonance Image (MRI) and compared the result with Markov Random Field hybridized Genetic Algorithm (MRF-GA). In modern power systems, stability of the power system is a major concern which will restore forces equal or greater than the distributing forces and maintain the equilibrium state. There are different type of conditions of a power system i.e. steady state stability, transient state stability, harmonics and disturbance, collapse of voltage and loss of reactive power. If the power system is unstable, it will cause major blackouts. So many methods came into existence for tuning of the PID controller to resolve this phenomenon. Abdul-Ghaffar et al. (2013) proposed a method for the stability of a single machine infinite bus power system using PID-PSS (Power System Stabilizer) where Hybrid Particle Swarm Bacterial Foraging Optimization has been used for fine tuning of the parameters.

1.5.3 Hybridization of GA with BFO

Kim et al. (2007) hybridized BFOA with GA and it is named as GA-BF and successfully applied the algorithm to PID controller problem. Kim and Abraham (2007) have applied Hybridized GA and BFO for designing Disturbance

Rejection tuning for PID controller. Verma et al. (2012) proposed Hybridized Bacterial Foraging and GA for optimal time table generation which is highly nonlinear NP-hard optimization problem. Shivkumar and Amudha (2013) tried to hybridize the Bacterial Swarming technology with GA and named it as Genetic Bacterial Swarming Algorithm (GBSA) and applied on the Permutation Flow Shop Scheduling Problem (PFSSP).

1.6 HANDLING CONSTRAINTS

The general constrained optimization problem is already discussed in Section 1.1. The basic idea for handling the constraints in it is to convert it into unconstrained problems first and then to solve it. The complexity of the problem normally increases if few of the existing functions are non-linear in nature. In this case, the optimization problem is called as non-linear optimization problem. To handle such complex problems, the easy handling constraint rules will help to save the computational time and makes the convergence faster. In the literature, many of constraint handling techniques exist. They are mainly classified as follows.

- Penalty Function Approach
- Feasibility Maintenance
- Separation of Objective function and Constraint Equations
- Multi Objective Evolutionary Algorithms (MOEAS)
- Keeping a feasibility solution by Special Representations and Genetic Operators
- Parallel population Approaches
- Hybrid Approaches
- Other Novel Approaches

The first three popular and useful approaches have been discussed briefly as follows. However, the rest are not in the scope of this thesis.

1.6.1 Penalty Function Approaches

The most well-liked approach among the constraint handling techniques is Penalty function approach. It was initiated by Courant (1943) and afterward extended by Carroll (1961), and Fiacco and Mc Cormick (1966). The constrained optimization problem is transformed into a series of unconstrained optimization problem. The solution obtained is the solution of the original constrained problem. The objective functions are added with penalty function. The penalty function consists of the penalty parameter multiplied by the measure of constraint violation. If the penalty parameters selected are very high, the potential solution gets trapped into local optimum and if the values are too low the process lowered down and feasible solution cannot be detected. The various types of approaches are (i) Death penalty, (ii) Static penalty, (iii) Dynamic penalty, (iv) Self Adaptive Fitness Formulation, (v) Adaptive Segregational Constraint Handling (vi) A Simple Multimembered Evolution Strategies, (vii) $\alpha-$ Constrained optimization, (viii) Stochastic Ranking and (ix) Bracket operator penalty. Among them, few of some popular approaches are discussed below. This thesis uses 'bracket operator penalty' which will be discussed at place where it is used (refer section 5.2.2).

Death Penalty:

It rejects the infeasible candidates. If any of the constraints is violated by the solution set, then the solution is rejected and another solution is generated. In this case large feasible region is required. With very small feasible region, the solution may face again the stagnation problem.

Static Penalty:

The penalty factor is independent of generation number. The factor remains constant throughout the evolutionary process. Homaifer et al. (1994) proposed the static penalty, in which

$$fitness(\vec{x}) = f(\vec{x}) + \sum_{k=1}^{r} \left(R_{i,k} \times \max[0, \phi_k(\vec{x})]^2 \right), \text{ where} \tag{1.12}$$

$\phi_k(\vec{x})$ denotes in general k^{th} equality as well as the inequality constraint.

Here r : the total constraints, m : is the total inequalities and l : the total equalities. $R_{i,k}$: the penalty coefficients for $i = 1, 2, 3, ..., s$. $f(\vec{x})$: the objective function and s : the number of violations.

Dynamic Penalty:

The penalty factor is dependent on the existing generation number. As the generation increases the penalty factor increases. Joines and Houck (1994) proposed the Dynamic penalty as follows.

$$\left.\begin{aligned} & fitness(\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \times SVC(\beta, \vec{x}), \\ & \text{where } C, \alpha, \beta \text{ are constants and} \\ & SVC(\beta, \vec{x}) = \sum_{k=1}^{m} D_k^{\beta}(\vec{x}) + \sum_{p=1}^{l} D_p(\vec{x}) \\ & D_k^{\beta}(\vec{x}) = \begin{cases} 0, g_k(\vec{x}) \le 0, \ 1 \le k \le m \\ |g_k(\vec{x})|, \ otherwise \end{cases} \\ & D_p(\vec{x}) = \begin{cases} 0, & -\varepsilon \le h_p(\vec{x}) \le \varepsilon, \ 1 \le p \le l \\ |h_p(\vec{x})|, & otherwise \end{cases} \end{aligned}\right\} \tag{1.13}$$

Dynamic penalties are better as compared to Static penalties as commented by the researchers. But deriving a good penalty factor is a typical task.

Adaptive Penalty:

This method sets the penalty coefficient by taking the feedback from the search process. It imparts good result when applied to GA. Additional parameters are not required for this method. The constraint violation of each constraint is

observed for each candidate solutions in the search process and accordingly the penalty coefficients will be set to minimize the violations. Barbosa and Lemonge (2004) defined the fitness for adaptive penalty as follows.

$$fitness\ (\vec{x}) = \begin{cases} f(\vec{x}),\ \ if\ \ \vec{x}\ \ \ is\ \ valid \\ \overline{f}(\vec{x}) + \sum_{j=1}^{r} k_j v_j(\vec{x}),\ \ \ otherwise \end{cases} \tag{1.14}$$

$$\overline{f}(\vec{x}) = \begin{cases} f(\vec{x}), if\ f(\vec{x}) \ < \langle f(\vec{x}) \rangle \\ \langle f(\vec{x}) \rangle, otherwise \end{cases} \tag{1.15}$$

Here the penalty coefficient k_j corresponding to the j^{th} constraint is defined at

every generation is given by

$$k_j = |\langle f(\vec{x}) \rangle| \frac{\langle v_j(\vec{x}) \rangle}{\sum_{l=1}^{r} [\langle v_l(\vec{x}) \rangle]^2}, \tag{1.16}$$

Where $\langle f(\vec{x}) \rangle$ is the average fitness in the current population and $\langle v_l(\vec{x}) \rangle$ is the violation of the l^{th} constraint averaged over the current population.

Adaptive Segregational Constraint Handling in Evolutionary Algorithm (ASCHEA):

The **ASCHEA** method uses the Adaptive penalty method proposed by Hamida and Schoenauer (2000). The Adaptive penalty method is as follows (Coello.pdf).

$$fitness\ (\vec{x}) = \begin{cases} f(\vec{x}),\ \ if\ \ \vec{x}\ \ \ is\ feasible\ solution \\ f(\vec{x}) - Penal(\vec{x}),\ otherwise \end{cases} \tag{1.17}$$

where

$$Penal(\vec{x}) = \alpha \sum_{j=1}^{m} g_j^+(\vec{x}) + \alpha \sum_{j=1}^{l} \left(h_j(\vec{x})\right) \tag{1.18}$$

The positive part of $g_j(\vec{x})$ is defined as $g_j^+(\vec{x})$ and the penalty factor used for all the constraints is α. The penalty factor is adapted based on the desired ratio of feasible solution (with respect to the entire population) τ_{target} and the current ratio at generation t i. e. τ_t. The adaptive penalty factor is defined as follows.

$$\alpha(t+1) = \begin{cases} \alpha(t)/\, fact, \; if \;\; (\tau_t > \tau_{target}) \\ \alpha(t) * fact, \quad otherwise \end{cases} \tag{1.19}$$

Here, $fact > 1$ and τ_{target} are user defined parameters and

$$\alpha(0) = \left| \frac{\sum_{i=1}^{n} f_i(\vec{x})}{\sum_{i=1}^{n} V_i(\vec{x})} \right| * 1000 \tag{1.20}$$

where $V_i(\vec{x})$ is the sum of the constraint violation of individual i^{th} solution.

α -Constrained Optimization:

The **satisfaction level** in this method is the combination of the satisfaction level of each constraint (equality as well as inequality constraints) defined in equation (1.2). The satisfaction level defined on the inequality constraint and equality constraint in Eq. (1.2) are the piecewise continuous functions which are defined as follows (Takahama and Sakai.pdf).

$$\mu g_k(\vec{x}) = \begin{cases} 1, if\ g_k(\vec{x}) \le 0 \\ 1 - \dfrac{g_k(\vec{x})}{b_k}, \; if\, 0 \le g_k(\vec{x}) \le b_k \\ 0, \; otherwise \end{cases} \tag{1.21}$$

$$\mu h_p(\vec{x}) = \begin{cases} 1 - \dfrac{|h_p(\vec{x})|}{b_p}, if\ |h_p(\vec{x})| \le b_p \\ 0, \qquad otherwise \end{cases} \tag{1.22}$$

Here, $b_k, k = 1,2,3,...,m$ and $b_p, p = 1,2,3,...,l$ are positive real numbers.

$$\begin{cases} \mu(\vec{x}) = 1, if\ g_k(\vec{x}) \le 0, h_p(\vec{x}) = 0, for\ all\ k\ and\ p \\ 0 \le \mu(\vec{x}) \le 1, otherwise \end{cases} \tag{1.23}$$

Now the objective function becomes

$$Minimize\ \ \mu(\vec{x}) = \min_{k,p}\left\{\mu g_k(\vec{x}),\ \mu h_p(\vec{x})\right\} \tag{1.24}$$

1.6.2 Feasibility Maintenance

This approach maintains the feasibility of the solutions in the population by an attempt of converting the infeasible solutions into feasible one. There are basically two approaches.

- Repairing Strategy
- Homomorphous Mapping

Repairing Strategy:

Mainly there are three types of repairing strategies as discussed below.

(i) In **Lamarckian approach** (LMCK), the infeasible solution is transformed into feasible solution through genetic modification. It replaces the infeasible solution for further evolution.

(ii) In **Baldwinian approach** (BLDW), combination of learning and evolution has been considered for evolution. It is less destructive in nature and solutions are repaired for further evolution. Although the

speed of convergence is slow, it still tries to reach near the global optimum.

(iii) In **Annealing approach** (ANNL), the infeasible solutions are accepted with certain probability.

Homomorphous Mapping:

'Homomorphous mapping' maps the feasible domain onto a hypercube and performs the evolutionary operators within the hypercube. Homomorphous mapping has high computational cost because each individual undergoes through some optimization methods by forward and backward mapping.

1.6.3 Separation of Objective function and Constraint Equations:

- Co evolution
- Superiority of Feasible points

Co evolution:

This technique is first introduced by Paredis (1994) where the population is initially divided in to two parts. Constraint equations are kept in the first part of the population and second part of the population consists of potential solutions to the problem. This strategy adopted in this process is analogous to Predator-Prey model. The selection pressure of one population and the fitness of other population are interrelated. A solution having high fitness in the second population satisfies the maximum number of constraints in the first population. A constraint having high fitness in the first population violates a lot of solutions in the second part. So, there arises an encounter between the constraints in the first part and solutions in the second part. Every candidate solution remembers its total number of encounters. The fitness is evaluated based on the sum of the encounters the individual faced in the last n encounters.

Superiority of Feasible points:

The central idea of this method draws the inspiration from the fact that feasible solutions are superior to infeasible solutions. The method works well as long as the assumption holds true. But when the ratio of the feasible search space and the whole search space is too small, the algorithm is trapped into the local optimum. Powell and Skolnick (1993) implemented heuristic rule for evaluation of the infeasible solutions proposed by Richardson et al. (1989). All feasible solutions are mapped onto the interval $(-\infty,1)$ and all the infeasible solutions are mapped onto the interval $(1,\infty)$. The fitness function proposed by Powell and Skolnick, 1993 is defined as follow.

$$fitness_i(\vec{x}) = \begin{cases} f_i(\vec{x}), if \ \ \vec{x} \ \ feasible \\ 1 + s\sum_{i=1}^{r} \phi_i(\vec{x}), \ \ otherwise \end{cases} \tag{1.25}$$

$f_i(\vec{x})$ is mapped onto the interval $(-\infty,1)$ and the constraint function $\phi_i(\vec{x})$ is mapped onto $(1,\infty)$ for $i = 1,2,..,r$. Here r denotes the total number of constraints (l equality as well as m inequality constraints) and s is a constant.

1.7 THESIS OVERVIEW

1.7.1 Motivation

Motivated by the recent state-of-the-art hybridized algorithms of GA with BFO, an effort has been done in this regard keeping in view the further enhancement of GA for tackling complex optimization problems. The efficiency, robustness and reliability of the algorithm are first tested on unconstrained and constrained optimization problems. Further, the algorithm has been applied on real life problems in unconstrained optimization as well as in constrained optimization. In case of unconstrained optimization, three examples of Model Order Reduction Problems (MOR) have been picked up to be solved by the said algorithm. In case of constrained optimization, the algorithm has been applied to

solve Non-Convex Economic Load Dispatch problem.

1.7.2 Summary of the thesis

The Chapter wise summary of the thesis is discussed below.

The **Chapter 1** is introductory in nature. Mathematical formulation of unconstrained as well as constrained optimization problems has been discussed in it. A review of Evolutionary Algorithms and Swarm Intelligence techniques were done. Literature review of GA and BFO have been presented in Chapter 1 with their successful applications to real life problems. The hybridization of GA and BFO with other nature inspired algorithms has been discussed. The introductions to constrained optimization followed by various type of constraint handling techniques have also been explained in this chapter. The major tool and terminologies used in this thesis are discussed in length.

Chapter 2 introduces the proposed hybridization of GA and BFO, called chemo-inspired GA (CGA). Both the inspiration and proposition is presented in this chapter. Therefore, it may be treated as the heart of the thesis. The supremacy of GCA is being realized over GA-BF (introduced by Kim et. al., 2007).

The efficiency of the proposed CGA has been well tested over a set of four unconstrained benchmark functions. It is further applied to solve three real life problems. There are (i) System of linear equations, (ii) Frequency Modulation Sound parameter Identification problem, and (iii) Capacities of Gas production Facilities in Mechanical Engineering. The first two real life problems has been compared with LXPM and HLXPM (Deep and Das, 2009), where real coded GA uses power mutation and Laplace crossover. The third real life problem is compared with GA. The overall conclusion of this chapter is CGA is faster and better than

i. Simple GA
ii. Real coded GA (i.e. LX-PM, that uses Laplace crossover and power mutation)

iii. Hybridization of GA and BFO

Therefore, hybridization of only chemotaxis step of BFO in GA is sufficient enough as compared to the hybridization of the entire BFO with GA.

In **Chapter 3**, the test bed is increased to a set of 22 benchmark functions of different taste. It has been further compared with Quadratic Approximation Hybridized GA (QGA), where GA has been hybridized with Quadratic Approximation operator (Deep and Das, 2008).

Chapter 4 presents the application of CGA to solve Model Order Reduction (MOR) problems of time invariant linear continuous system of Single Input and Single Output system. It is worth to note here that MOR is an unconstrained optimization problem. In this Chapter, three different cases of Model order reduction problems have been picked up and solved. The results have been compared with Linearly Increasing Cognitive Learning factor in Differential Evolution (LICLDE) (Bansal and Sharma, 2012), Fitness Based position update process in Differential Evolution (FBDE) (Bansal et al., 2012) and other state-of-the-art algorithms. The strength of CGA in solving unconstrained real life problem is well established.

In **Chapter 5**, CGA is further developed to solve the constrained optimization problems. The algorithm thus named as Chemo-GA for Constrained Optimization (CGAC), where the last 'C' stands for Constrained Optimization. 15 typical benchmark problems have been taken from the literature and solved by CGAC. The algorithm has been compared with LXPMC and HLXPMC (Deep and Das, 2013), where the real coded GA uses Laplace Cross over and Power Mutation. Further, computational time and number of function evaluations required by CGAC are compared with its competitors over a set of 11 benchmarking function. This chapter considers the successful application of CGAC over a set of 6 typical engineering design optimization problems. It has been compared with Differential Evolution with Level Comparison (DELC), Differential Evolution with Dynamic Stochastic Selection (DEDS), GA and

other such evolutionary algorithms. The efficiency of CGAC is confirmed in its conclusion.

Chapter 6 presents a successful application of CGAC on a Non-Convex Economic Load Dispatch (ELD) problem. Four power system problems having 3, 13, 40 and 15 generators are being considered. The first three test cases are solved taking into account the valve point loading effect including the power balance constraints and generator capacity constraints. The results have been compared with CPSO-SQP (PSO hybridized with Sequential Quadratic Programming), EP-SQP (Evolutionary Programming hybridized with Sequential Quadratic Programming), PSO and many other popular algorithms in terms of total generation cost, CPU time. Further, the best, average and worst function values along with standard deviations by CGAC are compared with the result of 32 different state-of-the-art algorithms. The high efficiency of CGAC is clearly concluded. CGAC also outperforms GA-BFO in terms of number of function calls, objective function values and CPU time. At the end of this chapter, the last test case i.e. the 15 generator problem is solved considering all the generational constraints viz. (i) Quadratic cost function, (ii) Generator capacity constraint, (iii) Power balance constraint, (iv) Ramp rate limits and Prohibited Operating Zones (POZ) constraints. The algorithm is being compared with Iteration PSO (IPSO), GA, PSO, Different versions of Chaotic PSO (CPSO), Self-organizing Hierarchical PSO (SOH_PSO), etc.

In **Chapter 7**, the conclusions of the thesis are outlined. All in all, the chemo hybridization of GA performs faster in solving both unconstrained and constrained optimization problems. It is better, above a large number of similar bio-inspired algorithms in terms of less CPU time, better function value, faster convergence and impact of less standard deviation. Lastly, the scopes of this present work are suggested for future research.