CHAPTER 4

OPTIMIZATION TECHNIQUES

4.1 INTRODUCTION

Nowadays, use of optimization techniques is becoming very popular in engineering optimization problems. In particular, greater attention is being given to Artificial Intelligence (AI) based optimization techniques such as Continuous Ants Colony optimization Algorithm, Artificial Neural Network (ANN), Genetic Algorithm (GA) etc for modeling the machining processes. However there are very less examples of practical implementation of AI techniques in actual machining operations. In this part of research work, an attempt has been made to identify the best non-traditional optimization technique among the other implemented techniques. For uniformity and comparison, standard available mathematical models are adopted from the refereed journals and the proposed techniques are tested for its robustness and versatile behaviour. The various non-traditional optimization techniques used in this work are as follows:

- Genetic Algorithm (GA)
- Simulated Annealing (SA) Algorithm
- Continuous Ants Colony (CACO) Algorithm
- Artificial Neural Networks (ANN)
4.2 GENETIC ALGORITHM

4.2.1 Introduction

Genetic Algorithm form a class of adaptive heuristics, based on principles derived from the dynamics of natural genetics. The searching process simulates the natural evolution of biological creatures and turns out to be an intelligent exploitation of a random search. A candidate solution (chromosome) is represented by an appropriate sequence of numbers. In many applications the chromosome is simply a binary string of 0’s and 1’s. The quality of its fitness function evaluates a chromosome, with respect to the objective function of the optimization problem. A selected population of solution (chromosome) initially evolves by employing mechanisms modeled similar to those used in Genetics. The figure 4.1 presents the flowchart for Genetic Algorithm

4.2.2 Fitness function

GA’s mimics the survival-of-fittest principle of nature to make a search process. Therefore, GA’s are naturally suitable for solving maximization problems. Minimization problems are usually transformed into maximization problems by suitable transformation. In general, a fitness function \( f(x) \) is derived from the objective function and used in successive genetic operations. The operation of GA’s begins with a population of random strings representing design or decision variables. Thereafter each string is evaluated to find the fitness value. The population is operated by three main operators – reproduction, crossover and mutation to create a new population of points. The new population is further evaluated and tested for termination. If the termination criterion is not met, the population is iteratively operated by the above three operators and evaluated. This procedure is continued until termination is met.
4.2.3 GA Operators

Reproduction

Reproduction is the first operator applied on a population. Reproduction selects good strings in a population and forms a mating pool and hence known as selection operator. There exist a number of reproduction operators in GA, but the essential idea in all of them is to pick the above-average strings from the current population and insert their multiple copies in the mating pool in a probabilistic manner.

In this work, Rank Order method is used as reproduction operator. The rank is given according to the fitness value of each chromosome. If fitness is more, higher the rank given, so that the probability (this is fixed for less rank and more for high rank) for selecting that particular string is more.

The probability values for selection, according to

\[(\text{Max-Min})(\text{Rank (i, t)-1})\]

\[
\text{Expected value (i, t) = Min + \frac{(\text{Max-Min})(\text{Rank(i, t)-1})}{N-1}} \tag{4.1}
\]

Crossover

In crossover, new strings are created by exchanging information among strings of the mating pool based on a probability \(P_c\). Actually strings are picked from the mating pool and some portions of the strings are exchanged between the strings. A string point crossover operator is used in this work which is performed randomly by choosing a crossing site along the string and by exchanging all bits in on the right side of the crossing site as shown.
Before crossover: The crossover site is selected as 7\textsuperscript{th} bit.

\[
\begin{array}{c|c}
0000111 & 000 \\
1101001 & 111 \\
\end{array}
\]

After crossover:

Crossover has been made between the strings after the 7\textsuperscript{th} bit.

\[
\begin{array}{c|c}
0000111 & 111 \\
1101001 & 000 \\
\end{array}
\]

**Mutation**

The mutation operator changes 1 to 0 and vise versa with a small mutation probability $P_m$. Here the operator performs a bit-wise mutation. The need for mutation is to create a point in the neighbor of the current point, thereby achieving a local search around the current solution. Mutation is also used to maintain diversity in the population.

Before mutation:

\[1101001000\]

After mutation:

\[1100001000\] (here the mutation is carried out on the 4\textsuperscript{th} bit)
4.2.4 Algorithm

Step 1: Choose a coding to represent problem parameters, a selection operator, crossover and a mutation operator. Choose a population
size n, crossover probability $P_c$ and a mutation probability $P_m$. Initialize a random population of string of size l. Choose a maximum allowable generation number $t_{\text{max}}$. Set $t = 0$;

Step 2: Evaluate each string in the population.

Step 3: If $t > t_{\text{max}}$ or other termination criteria is satisfied, terminate.

Step 4: Perform reproduction on the population

Step 5: Perform crossover on random pairs of string.

Step 6: Perform bit wise mutation.

Step 7: Evaluate string in the new population. Increment $t = t + 1$ go to Step 3.

End.

4.3 SIMULATED ANNEALING ALGORITHM (SA)

4.3.1 Introduction

The Simulated Annealing method [110] resembles the cooling process of molten metals through annealing. At high temperature, the atoms in the molten metal can move freely with respect to each another. But, as the temperature is reduced, the movement of the atoms gets reduced. The atoms start to get ordered and finally form crystals having the minimum possible energy. However, the formation of the crystal depends on the cooling rate. If the temperature is reduced at a very fast rate, the crystalline state may not be achieved at all; instead the system may end up in a polycrystalline state, which may have a higher energy state than the crystalline state. Therefore in order to achieve the absolute minimum state, the temperature needs to be reduced at a slow rate. The process of slow cooling is known as annealing in
metallurgical parlance. SA simulates this process of slow cooling of molten metal to achieve the minimum function value in a minimization problem.

The cooling phenomenon is simulated by controlling a temperature-like parameter introduced with the concept of the Boltzmann probability distribution. According to the Boltzmann probability distribution, a system in thermal equilibrium at a temperature $T$ has its energy distributed probabilistically according to

$$P(E) = \exp\left(-\frac{E}{kT}\right),$$

where $k$ is the Boltzmann constant.

This expression suggests that a system at a high temperature has almost uniform probability of being at any energy state, but at a low temperature it has a small probability of being at a high energy state. Therefore, by controlling the temperature $T$ and assuming that the search process follows the Boltzmann probability distribution, the convergence of an algorithm can be controlled using the Metropolis algorithm.

Let us say, at any instant the current point is $x^{(t)}$ and the function value at that point is $E(t) = f(x^{(t)})$. Using the Metropolis algorithm, the probability of the next point being at $x^{(t+1)}$ depends on the difference in the function values at these two points or on $\Delta E = E(t+1) - E(t)$ and is calculated using the Boltzmann probability distribution:

$$P(E(t+1)) = \min[1, \exp(-\Delta E/kT)].$$

If $\Delta E \leq 0$, this probability is one and the point $x^{(t+1)}$ is always accepted. In the function minimization context, this makes sense because if the function value at $x^{(t+1)}$ is better than that at $x^{(t)}$, the point $x^{(t+1)}$ must be accepted. When $\Delta E > 0$, which implies that the function value at $x^{(t+1)}$ is
worse than that at \(x^{(t)}\). According to the Metropolis algorithm, there is some finite probability of selecting the point \(x^{(t+1)}\) even though it is a worse than the point \(x^{(t)}\). However this probability is not the same in all situations. The probability depends on relative magnitude of \(\Delta E\) and \(T\) values. If the parameter \(T\) is large, this probability is more or less high for points with largely disparate function values. Thus, any point is almost acceptable for a large value of \(T\). For small values of \(T\), the points with only small deviation in function value are accepted. The figure 4.2 illustrates the flowchart of Simulated Annealing algorithm.

### 4.3.2 Algorithm

Step 1: Choose an initial point \(x^{(0)}\), a termination criterion \(\epsilon\). Set \(T\) as a sufficiently high value, number of iterations to be performed at a particular temperature \(n\), and set \(t = 0\).

Step 2: Calculate a neighbouring point \(x^{(t+1)} = N(x^{(t)})\). Usually, a random point in the neighbourhood is created.

Step 3: If \(\Delta E = E(x^{(t+1)}) - E(x^{(t)}) < 0\), set \(t = t + 1\); Else create a random number \((r)\) in the range \((0, 1)\). If \(r \leq \exp(-\Delta E/T)\), set \(t = t + 1\); Else go to Step 2.

Step 4: If \(|x^{(t+1)} - x^{(t)}| < \epsilon\) and \(T\) is small, Terminate. Else goto step 2.
The initial temperature (T) and the number of iterations (n) performed at a particular temperature are two important parameters which govern the successful working of the Simulated Annealing procedure. If a large initial T is chosen, it takes a number of iterations for convergence. On the other hand, if a small initial T is chosen, the search is not adequate to thoroughly investigate the search space before converging to the true optimum. A large value of n is recommended in order to achieve quasi-equilibrium state at each temperature, but the computation time is more. Unfortunately, there are no unique values of the initial temperature and n that work for every problem. However, an estimate of the initial temperature can
be obtained by calculating the average of the function values at a number of random points in the search space. A suitable value of $n$ can be chosen (usually between 20 to 100) depending on the available computing resource and the solution time.

4.4 ANTS COLONY ALGORITHM

4.4.1 Introduction

The Ant System [98] is a new kind of co-operative search algorithm inspired by the behaviour of colonies of real ants. The blind ants are able to find astonishing good solutions to shortest path problems between food sources and their home colony. The medium used to communicate information among individuals regarding paths, and decide where to go, was the pheromone trails. A moving ant lays some pheromone on the path they move, thus marking the path by the substance. While an isolated ant moves essentially at random, it can encounter a previously laid trail and decide with high probability to follow it, and also reinforcing the trail with its own pheromone. The collective behaviour that emerges in a form of autocatalytic behaviour where the more the ants following a trail, the more attractive that trail becomes for being followed.

4.4.2 The Path of Ants

There is a path along which ants are walking from nest to the food source and vice versa. If a sudden obstacle appears and the path is cut off, the choice is influenced by the intensity of the pheromone trails left by proceeding ants. On the shorter path more pheromone is laid down. The figure 4.3 details the behaviour of ants when faced with an obstacle in its search path.
Figure 4.3 Path traced by ants without and with obstacle

Ants Colony algorithm can be applied for the continuous function optimization problems. Here, the domain has to be divided into a specific number of R randomly distributed regions. These regions are indeed the trial solutions and act as local stations for the ants to move and explore. The fitness of these regions are first evaluated and sorted on the basis of fitness. Totally a population of ants explores these regions; the updating of the regions is done locally and globally with the local search and global search mechanism respectively. The distribution of local and global ants is illustrated in figure 4.4 and the flowchart of the Ants Colony algorithm is presented in figure 4.5.

4.4.3 Global Search

The global search creates G new regions by replacing the weaker portions of the existing domain. In the ACO random walk and trial diffusion are utilized for global search. By random walk procedure, the ants move in new directions in search of newer and richer stocks of food source. In the ACO simulation such a global search in the entire domain is done by process equivalent to crossover operation and mutation operations in G.A. Adding or subtracting with a probability proportional to the mutation probability carries
out the mutation step in ACO. The mutation step is reduced as per the relation.

\[ \Delta(T, R) = R \left[ 1 - r^{(1-T)b} \right] \]  

Where ‘\( r \)’ is a random number from [0, 1] ‘\( R \)’ is the maximum step size. ‘\( T \)’ is the ratio of the current iteration number to that of the total number of iterations; ‘\( b \)’ is a positive parameter controlling the degree of nonlinearity. The mutation radius is nonlinearly reduced with increasing iterations. The scaling down enables enhanced probability of locating maximum by concentrated search procedure called trial diffusion. The trial diffusion is quite similar to arithmetic cross over. In this step two parents are selected at random from the parent population space. The elements of the child’s vector can have either (1) The corresponding element from the first parent, (2) The corresponding element from the second parent and (3) A combination arrived from a weighted average of the above. i.e. If the random number is less than 0.5

\[ X_{\text{child}} = \alpha X_{i(\text{parent1})} + (1-\alpha) X_{i(\text{parent2})} \]  

Where \( \alpha \) is a uniform random number in the range (0 –1). If the random number is in between 0.5 and 0.75 then:

\[ X_{\text{child}} = X_{i(\text{parent1})} \]

And if the random number is in between 0.75 and 1.00 then

\[ X_{\text{child}} = X_{i(\text{parent2})} \]
4.4.4 Local Search

In the local search, the ants have the capability of selecting regions proportional to the current pheromone values of superior and inferior regions. Local updating is applied only on superior regions. In an ACO, local ants select a region ‘i’ given by

\[ P_i(t) = \frac{\tau_i(t)}{\sum \tau_k(t)} \]  

(4.4)

where ‘i’ is the region index and \( \tau_i \) is the pheromone trail on region \( i \) at time \( t \). After selecting the destination, the ant moves through a short distance. The direction of the movement will be the same as that of the previous direction if there is an improvement in the fitness. If there is no improvement it searches in a random direction. If improvement in fitness is found in the above procedure, the regions position vector is updated.

Figure 4.4 Distribution of ants for local and global search
The Pheromone deposited by the ant is proportional to the increase in fitness. The age of the region is another important parameter in the ACO algorithm. The size of the ant movement in the local search depends on the current age.

- Initially all the regions are assigned with the pheromone value of 1.0
- It better results are obtained the pheromone of region i is modified by
  \[
  \text{Pheromone}_{i(\text{new})} = \frac{P_{r(\text{new})} - P_{r(\text{old})}}{P_{r(\text{old})}} + \text{Pheromone}_{i(\text{old})}
  \]
- Now again the average pheromone for each region is calculated and the procedure is repeated as many number of times as they are local ants.
- The termination criterion is the total number of iterations.

4.5 ARTIFICIAL NEURAL NETWORK (ANN)

4.5.1 Introduction

Artificial Neural Network is one of the computational tools for modeling when the fundamental data relationship is unknown and also it can recognize interconnected relationship between input data sets and corresponding target values. ANN mimics the learning process of the human brain and can process problems involving non linear and complex data even if the data are imprecise and noisy. ANN is a computational structure, which consists of simple computational units called neurons, they are highly interconnected. ANN are parallel computational model comprised of densely interconnected adaptive processing units. A very important feature of these networks is their adaptive nature, where learning by example replacing programming in solving problems.
Is new function value improved?

Yes

Move ants to new solution and update trial value of weakest solutions

Sort according to Function value

Print best Function value

No

Is the termination criteria reached?

Yes

Results Output

Create initial population

Send 90% of ‘G’ global ants for crossover and mutation

Send 10% of ‘G’ global ants for Trial diffusion

Update trial value of weakest solution

Select region and send ‘L’ local ants

No

Figure 4.5 Flowchart of ants colony optimization algorithm
4.5.2 ANN Model

ANN model built with layers of units, and thus are termed multilayer ANN. Generally ANN model constructed with following layers,

- First layer consists of input units called independent variable.
- Last layer contains output units these are called dependent variable.
- All other units in the model are called hidden units and comprise hidden layer.

Functions governing the behavior of a unit in a particular layer for whole model

1. The input function
2. The output/activation function

- The input function: Input into a node is weighted sum of outputs from nodes connected to it. The input function normally given by following equation.

\[
net_i = \sum_j W_{ij} X_j + \mu_i \quad (4.5)
\]

- net\(_i\) = the result of the net inputs impacting on unit i
- \(W_{ij}\) = weights connecting neuron j to neuron i.
- \(X_j\) = output from unit j
- \(\mu_i\) = threshold for neuron i.
• Activation function: Each unit takes its net input and applies an activation function to it. The threshold function is useful in situations where the inputs and outputs are binary encoded. The general threshold function is sigmoid, such as

\[ g(\text{netinput}) = \tanh(\text{netinput}) \]

Various steps in developing neural network modeling

Step 1: Variable selection for modeling.

Step 2: Formation of data set. Dataset is divided into three distinct sets called training, testing and validation sets.

Training data set: Largest data set and is used by neural network to learn patterns present in the data.

Testing data set: Evaluate the generalization ability of a supposedly trained network.

Validation data set: final check on the performance of the trained network.

Step 3: Neural Network architecture construction defines its structure including number of hidden layers, number of hidden nodes, number of output nodes and activation function.

Step 4: model building

4.5.3 Back Propagation Algorithm

Back propagation algorithm is one of the supervised learning algorithm, which uses the data to adjust the network’s weights and thresholds so as to minimize the error in its predictions on the training set.
The back propagation algorithm consists of four steps

**Step 1:** compute how fast the error changes as the activity of an output unit is changed. This Error Derivative (EA) is the difference between the actual and desired activity.

\[
EA_j = y_j - d_j \tag{4.6}
\]

- \(y_j\) = actual activity
- \(d_j\) = desired activity

**Step 2:** compute how fast the error changes as the total input received by an output unit is changed. This quantity EI is the answer from step 1 multiplied at the rate at which the output of a unit changes as its total input is changed.

\[
EI_j = \frac{\partial E}{\partial X_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y_j}{\partial X_j} = EA_j y_j (1 - y_j) \tag{4.7}
\]

**Step 3:** compute how fast the error changes as a weight on the connection into an output unit is changed. This quantity (EW) is the answer from step 2 multiplied by the activity level of the unit from which the connection emanates.

\[
EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial X_j} \times \frac{\partial X_j}{\partial W_{ij}} = EI_j y_i \tag{4.8}
\]

**Step 4:** compute how fast the error changes as the activity of a unit in the previous layer is changed. This crucial step allows backpropagation to be applied to multi-layer networks. When the activity of a unit in the previous layer changes, it affects the activities of all the output units to which it is connected. So to compute the overall effect on the error, we add
together rall these separate effects on output units. But each effect is simple to calculate. It is the answer in step 3 multiplied by the weight on the connection to that output unit.

\[
EA_j = \frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial X_j} \times \frac{\partial X_j}{\partial y_i} = \sum_j EI_j W_i
\] (4.9)

By using step 2 and 4, we can convert the EA of one layer of units into EA for the previous layer. The procedure can be repeated to get the EA for as many previous layers as desired. Once we know the EA of a unit, we can use step 2 and 3 to compute the EW on its incoming connections.

4.6 DESIGN OF EXPERIMENT

4.6.1 Introduction

Generally the experiments are to be conducted for investigate about the performance of particular system or process. In engineering visualize the process as a combination of machines and methods. Experimental design is important tool for improving the performance of manufacturing process. The experiments are designed based on factors and its levels called factorial design. Factorial designs are used primarily for screening significant factors, but can also be used sequentially to model and refine a process. Number of factorial design concepts available with design of experiment concepts. These are 2-level factorial design, irregular fraction designs, general factorial designs, D-optimal design and Taguchi design. Based on the above discussion this work uses the Taguchi method to measure the performance characteristics of process.

4.6.2 Taguchi method

Taguchi method is concerned with quality engineering and product development. Taguchi method is important to evaluate the two kinds of losses the loss due to function and one due to harmful effects. According to Taguchi
its purpose is to adjust the parameter levels so that the objective characteristic will not vary much even if the system and environmental parameter change. It is search for the parameter levels at which the characteristics is stable. And reformulates this term is “finding levels of the process variables that achieve a mean performance at a targeted value and at the same time reducing variability around that value”. Based on Taguchi statement two kinds of factors that affect the characteristic of the product, these are design factors and noise factors. In that design factor is known as control factors, are those factors that can be controlled easily during the manufacturing process. Noise factors on the other hand, are factors that are hard to control, such as environmental or their levels will vary within a certain range during the production process.

4.6.3 Taguchi experimental design

Taguchi’s approach to parameter design provides the design engineer with a systematic and efficient method for determining near optimum design parameters for performance and cost. The objective is to select the best combination of control parameters so that the product or process is most robust with respect to noise factors. The purpose of this experiment is to determine the optimal levels for the control factors in conjunction with reducing the product’s performance around the target value. The Taguchi method utilizes orthogonal arrays from design of experiments theory to study a large number of variables with a small number of experiments. Using orthogonal arrays significantly reduces the number of experimental configurations to be studied.

Orthogonal arrays are not unique to Taguchi. However, Taguchi has simplified their use by providing tabulated sets of standard orthogonal arrays and corresponding linear graphs to fit specific projects. A typical tabulation is shown in Table 4.1.
Table 4.1 $L_9(3^4)$ orthogonal array

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

In this array, the columns are mutually orthogonal. That is, for any pair of columns, all combinations of factor levels occur; and they occur an equal number of times. Here there are four parameters A, B, C, and D, each at three levels. This is called an "L 9" design, with the 9 indicating the nine rows, configurations, or prototypes to be tested. Specific test characteristics for each experimental evaluation are identified in the associated row of the table. Thus, L 9 means that nine experiments are to be carried out to study four variables at three levels. The number of columns of an array represents the maximum number of parameters that can be studied using that array. Note that this design reduces $81 (3^4)$ configurations to 9 experimental evaluations.

The Taguchi method can reduce research and development costs by improving the efficiency of generating information needed to design systems that are insensitive to usage conditions, manufacturing variation, and deterioration of parts. As a result, development time can be shortened significantly; and important design parameters affecting operation, performance, and cost can be identified. Furthermore, the optimum choice of
parameters can result in wider tolerances so that low cost components and production processes can be used. Thus, manufacturing and operations costs can also be greatly reduced.

The figure 4.6 provides a brief overview of the process followed by Taguchi’s approach to parameter design.

**Fig 4.6 Flowchart for taguchi method**