Chapter 7 - Genetic algorithm based optimization of machining process

GA is a search algorithm based on the hypothesis of natural selections and natural genetics also the GA is a parallel and global search technique that emulates natural genetic operations [104]. GA can find a global solution after sufficient iterations, but has a high computational burden. Recently, a global optimization technique using GA has been successfully applied to various areas of power system such as economic dispatch [105,106], unit commitment [107,108], reactive power planning [109-111], and power plant control [112,113]. GA-based approaches for optimization of machining parameters have several advantages. Naturally, they can not only treat the discrete variables but also overcome the dimensionality problem. In addition, they have the capability to search for the global optimum or quasi optima within a reasonable computation time. To enhance GA’s computational efficiency, an improved evolutionary direction operator (IEDO) modified from [114] and a migration operator [115] are embedded in GA to form the IGA. On the contrary, studies on evolutionary algorithms have shown that these methods can be efficiently used to eliminate most of the above-mentioned difficulties of classical methods [116].

The selection of optimal cutting parameters, like depth of cut, feed and speed, is a very important issue for every machining process. In workshop practice, cutting parameters are selected from machining databases or specialized handbooks, but the range given in these sources are actually starting values, and are not the optimal values [106]. In any optimization procedure, it is a crucial aspect to identify the output of chief importance, the so-called optimization objective or optimization criterion.

In this section, an improved genetic algorithm (IGA), which can overcome the aforesaid problems of the conventional GA to some extent, is developed to obtain the optimal parameters in turning processes. The proposed IGA incorporates the following two main features. First, an artificial creation scheme for an initial population is devised, which also takes the random creation scheme of the conventional GA into account. Second, a stochastic crossover strategy is developed. In this scheme, one of the three different crossover methods is randomly selected.
from a biased roulette wheel where the weight of each crossover method is determined through pre-performed experiments. The stochastic crossover scheme is similar to the stochastic selection of reproduction candidates from a mating pool. The IGA requires only a small population, and it is more efficient than GA. The results of the IGA are compared with those of the conventional simple genetic algorithm.

Genetic algorithms are very different from most of the traditional optimization methods. Genetic algorithms need design space to be converted into genetic space. So, genetic algorithms work with a coding of variables. The advantage of working with a coding of variable space is that coding discretizes the search space even though the function may be continuous. A more striking difference between genetic algorithms and most of the traditional optimization methods is that GA uses a population of points at one time in contrast to the single point approach by traditional optimization methods. This means that GA processes a number of designs at the same time. As we have seen earlier, to improve the search direction in traditional optimization methods, transition rules are used and they are deterministic in nature but GA uses randomized operators. Random operators improve the search space in an adaptive manner.

Three most important aspects of using GA are:

1. definition of objective function
2. definition and implementation of genetic representation
3. definition and implementation of genetic operators.

Once these three have been defined, the GA should work fairly well beyond doubt. We can, by different variations, improve the performance, find multiple optima (species if they exist) or parallelize the algorithms.

Genetic algorithms are computerized search and optimization algorithms based on the mechanics of natural genetics and natural selection: Prof. Holland of University of Michigan, Ann Arbor, envisaged the concept of these algorithms in the mid-sixties and published his work [117].
Thereafter, a number of students and other researchers have contributed to the development of this field.

To date, most of the GA studies are available by Davis [118], Goldberg [119], Michalewicz [120] and Deb [121] and through a number of conference proceedings. The first application towards structural engineering was carried by Goldberg. He applied genetic algorithms to the optimization of a ten-member plane truss. P. Ju [122] applied genetic algorithm for the design of Static Compensator in an integrated power system. Apart from structural engineering there are many other fields in which GA’s have been applied successfully. It includes biology, computer science, image processing and pattern recognition, physical science, social sciences and neural networks. In this chapter, we will discuss the basic concepts, representatives of chromosomes, fitness functions, and genetic inheritance operators with example and how this will be adopted for the power system stability low frequency damping problem.

7.1. Genetic Algorithm Based Optimization

First, coding scheme is to be defined and the initial population is produced. The computation with genetic operators is used to evaluate fitness with respect to the objective function [79]. Fig 7.1 shows the GA based optimization procedure.

The genetic algorithm (GA) has gained momentum in its application to optimization problems. Unlike strict mathematical methods, the GA does not require the condition that the variables in the optimization problem be continuous and different; it only requires that the problem to be solved can be computed. So, the GA has an apparent benefit to adapt to irregular search space of an optimization problem [79]. Therefore, in this approach, the GA has been used for optimization of surface roughness for the machining process. The basic operators in the GA include reproduction, crossover, and mutation. The input gains and output gain are taken as individuals in GA, and are represented by a binary string of length 200 with 50 bit for each individual.
Initialization of Random Generation of P Chromosomes

Generation = 1

Evaluate Fitness Function for the Chromosomes

Selection, Crossover

Fitness = Fitness$_{\text{max}}$

Mutation

Fitness = Fitness$_{\text{max}}$

Yes → End

Generation = Generation + 1

Yes → Generation = Gen$_{\text{max}}$

No → End

Fig. 7.1. GA Optimisation Algorithm
Steps in Genetic Algorithm

- Create the initial population.
- Evaluate the fitness of each individual.
- Select the best individuals and perform recombination.
- Mutate the new generation.
- If termination condition is not reached, go back to step 2.

The calculation can be terminated for example when a certain fitness level is reached or after a certain number of iterations is performed. Also, if it seems that the solutions will not get any better for a long time, it can be deduced that it is best to stop the calculation.

**Selection**

The main idea behind the selection mechanism is better individuals get higher chance. There are many methods reported such as Roulette Wheel selection, Stochastic Universal sampling and Tournament selection, etc. In this approach, Tournament selection method which is one of the most widely used selection schemes. In tournament selection a specified number of individuals are selected from the current population size. The best individuals out of the best individuals get copy in a mating pool. The selection of individuals can be performed either with replacement or without replacement. In selection with replacement the individuals selected for the current tournament are candidate for other tournaments. On the other hand, if selected without replacement the individuals once selected are not candidates for other tournaments. Tournament selection can be implemented very efficiently as no sorting of the population is required. The advantages of Tournament selection are No premature convergence, No stagnation, No global reordering is required and explicit fitness is not needed.
**Crossover**

Crossover is a mechanism, which creates new individuals by combining parts from two individuals. Crossover is explorative; it makes a big jump to an area somewhere “in between” two (parents) areas. Single point, multi point and uniform crossovers are available. In this work, simulated binary crossover (SBX) proposed by Deb and his students has been used. SBX crossover creates children solutions in proportion to the difference in parent solutions.

**Mutation**

Mutation is a mechanism, which creates new individual by making changes in a single individual. Mutation is explorative, it creates random small deviations, thereby staying near (in the area of) the parent. Only mutation can introduce new information. In this work polynomial mutation has been applied.

**Stopping criteria**

There are many no of stopping criteria are reported such as Maximum number of generation, Maximum number of functional evaluation, Convergence criteria, computation time etc. In this work, Maximum number of generations has been used as stopping criteria.

In order to optimize the present problem using genetic algorithms (GAs), The fitness function for the surface roughness is taken as the constrained optimization problem is stated as follows:
From the observed data for surface roughness, the response function has been determined using RSM and fitness function, defined as Minimize,

\[
R_a = -4.89 + 2.49F - 38.0D + 0.599V + 3.27R - 5.38F \times D + 0.0140F \times V - 18.2F \times R + \\
0.0097D \times V + 15.8D \times R - 0.232V \times R + 80.5F^2 + 16.5D^2 - 0.00318V^2 \tag{7.1}
\]

subject to

\[
39.269 \text{ m/min} \leq V \leq 94.247 \text{ m/min}
\]

\[
0.059 \text{ mm/rev} \leq F \leq 0.26 \text{ mm/rev}
\]

\[
0.4 \text{ mm} \leq D \leq 1.2 \text{ mm}
\]

\[
0.4 \text{ mm} \leq R \leq 1.2 \text{ mm}
\]

\[
x_{il} \leq x_i \leq x_{iu}
\]

where \(x_{il}\) and \(x_{iu}\) are the upper and lower bounds of process variables \(x_i\). \(x_1, x_2, x_3, x_4\) are the cutting speed, feed, depth of cut and nose radius respectively. In order to optimize the present problem using GAs, the following parameters have been selected to obtain optimal solutions with less computational effort.

**Population size** = 50

**Maximum number of generations** = 1000

**Total string length** = 50

**Crossover probability** \((P_c)\) = 0.9

**Mutation probability** \((P_m)\) = 0.01

Initially, a set of 50 random pairs of the coefficients are created, discarding the unstable cases. These 50 pairs of coefficients are converted into binary codes to construct the initial population termed as “old pop.” From this grouped population and by using the usual GA operators, equal numbers of new populations are generated. A specific probability of each operator is fixed, keeping the “mutation” probability sufficiently small. The crossover and mutation probabilities are taken as 0.9 and 0.01, respectively [79]. To select two strings of population for either mutation or crossover, the roulette wheel technique is used [79]. The technique specified that for selection, a random number between 0 and 1 is multiplied with the sum of fitness of all the “old
pop” strings. When this value is greater than or equal to the cumulative fitness of the $i^{th}$ string, this string is selected from the “old-pop.” In this manner, two strings (mate-1 and mate-2) are selected to the mating pool. Using the GA operators, two new strings (child-1 and child-2) are created out of these mates. This process is continued until 50 new strings of population are generated. Out of the original 50 strings and newly created 50 strings (a total of 100 strings), the most-fit 50 population strings are retained. These strings are replaced into the “old-pop” to represent the second generation “old- pop.” In this manner, 1000 generations are continued, before the algorithm converges into the fit unique solution. The binary data in the solution are decoded to provide the optimized machining parameters. The detailed flow chart is shown in Fig. 7.2.
Start

Initialization Pop size = 50
$P_c = 0.09$, $P_m = 0.01$, No. of bits = 50
No. of generations = 1000
No. of variables = 4

Evaluate Objective function

No

$J = \text{Pop}$

Yes

Genetic Operations
Selection, Crossover, Mutation

No

$I = \text{No. of generations}$

Yes

End

Fig. 7.2. The detailed flow chart of GA Optimisation Algorithm
7.1.1. Simulation Studies and Performance Evaluation

The CGA code was developed using MATLAB. The input machining parameter levels were fed to the CGA program. The CGA program uses different types of crossover and mutation operators to predict the values of tool geometry and cutting conditions for minimization of surface roughness. Table 7.1 shows the minimum value of surface roughness with respect to input machining parameters for CGA. It is possible to determine the conditions at which the turning operation has to be carried out in order to get the optimum surface finish. The genetic evolution history is described in figure 7.3 for CGA. The given problem is converted to a maximization problem and solved using CGA.

Table 7.1 – Output values of the genetic algorithm with respect to input machining parameters

<table>
<thead>
<tr>
<th>Machining Parameters</th>
<th>CGA Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed, F(mm/rev)</td>
<td>0.161564</td>
</tr>
<tr>
<td>Depth of cut, D(mm)</td>
<td>0.583087</td>
</tr>
<tr>
<td>Cutting Velocity, V(m/min)</td>
<td>39.985</td>
</tr>
<tr>
<td>Nose Radius, R(mm)</td>
<td>0.967974</td>
</tr>
<tr>
<td>Min. Surface Roughness, Ra(microns)</td>
<td>1.62085*10^{-10}</td>
</tr>
</tbody>
</table>
7.2. Improved Genetic Algorithm (IGA)

The main advantage of the IGA approach is that the “curse of dimensionality” and a local optimal trap inherent in mathematical programming methods can be simultaneously overcome. This section describes the proposed IGA. First, a brief overview of the IGA is provided then the solution procedures of the proposed IGA are stated.

The IGA is a parallel direct search algorithm that uses \( N_p \) vectors of variables in the nonlinear programming problem, namely, \( X^G = \{X^G_i, i = 1, \ldots, N_p\} \) as a population in generation \( G \). For convenience, the decision vector (chromosome) \( x_i \), is represented as \( (x_{ij_1}, x_{ij_2}, \ldots, x_{ij_{n_{ci}}}) \). Here, the decision variable (gene), \( x_{ij} \), is directly coded as a real value within its bounds.
7.2.1. Improved Evolutionary Direction Operator (IEDO)

The main shortcoming of the evolutionary direction operator (EDO) [114] is that it creates a new chromosome from three arbitrary chromosomes in each generation, making this search operator blind. The improvement of the IEDO is to choose three best solutions in each generation to implement the improved evolutionary direction operation, and then obtain a new solution that is superior to the original best solution. The IEDO is introduced below.

A chromosome which carries a set of solutions with \( n_c \) optimizing parameters may be expressed as \( x_j = C_1, C_2, ..., C_{n_c} \). Each \( C_p \) represents a continuous decision variable, and is limited by its lower and upper bounds (\( C_p^{\text{MIN}} \) and \( C_p^{\text{MAX}} \)). Three sets of optimal chromosomes are obtained after a generation. These three preferred chromosomes are ascended according to their fitness and called the “low,” “medium,” and “high” chromosomes, respectively.

Three inputs (preferred) and the output (created) chromosomes are denoted below.

Inputs:

- “low” chromosome, \( z_l = \{C_{l1}, C_{l2}, ..., C_{lnc}\} \), with Fitness \( F_l \)
- “medium” chromosome, \( z_m = \{C_{m1}, C_{m2}, ..., C_{mnc}\} \), with fitness \( F_m \)
- “high” chromosome, \( z_h = \{C_{h1}, C_{h2}, ..., C_{hnc}\} \), with fitness \( F_h \)

Output chromosome, \( \{C_{o1}, C_{o2}, ..., C_{onc}\} \), with fitness, \( F_{\text{new}} \)

The IEDO can significantly reduce the effort in searching for the optimal solution because it enhances the local searching capability for GA. Fig. 7.4 shows the flowchart of the minimum optimization for IEDO.
Fig. 7.4 Flow chart of operation for the improved evolutionary direction operator

1. Start of the IEDO
   \[ T_s = 1, \; D_1 = D_2 = 1, \; N_L = 4 \]

2. Choose three preferred fitness, and obtained \( C_{lp}, C_{mp} \) and \( C_{hp} \)

3. Compute \( C_{p}^{\phi} \) by employing
   \[
   C_{p}^{\phi} = C_{lp} + D_1 \cdot (C_{lp} - C_{mp} + D_2 \cdot (C_{lp} - C_{hp})
   \]

4. \( C_{op} = \max \{ \min (C_{p}^{\phi}, C_{p}^{\text{MAX}}), C_{p}^{\text{MIN}} \}, \; p = 1, ..., n_c \)

5. Evaluate the new fitness, \( F_{\text{new}} \)

6. If \( F_{\text{new}} < F_h \)

7. If \( F_{\text{new}} = F_i = F_m \)

8. Replace \( C_{lp} \) by \( C_{c_{op}} \), if \( F_{\text{new}} < F_i \)

9. Replace \( C_{mp} \) by \( C_{c_{op}} \), if \( F_i < F_{\text{new}} < F_m \)

10. Replace \( C_{hp} \) by \( C_{c_{op}} \), if \( F_m < F_{\text{new}} < F_h \)

11. \( T_s = T_s + 1 \)

12. \( D_1 = D_1 \times (-0.5) \)

13. \( D_2 = D_2 \times (-0.5) \)

14. End of the IEDO

End of the IEDO

\[ C_{p}^{\phi} = C_{p}^{\phi} + N_r \]
Step 1. The magnitudes of the two evolution directions is set to 1 (i.e., $D_1 = 1, D_2 = 1$). Then, set the initial index of the IEDO to 1 ($T_s = 1$), and set the number of the IEDO loop to 4 (i.e., $N_L = 4$).

Step 2. Choose three preferred fitness values ($F_l, F_m$ and $F_h$) and find their associated chromosomes ($Z_l, Z_m$ and $Z_h$). Then, obtain three preferred decision variables ($C_{lp}, C_{mp}$, and $C_{hp}$, $p = 1; \ldots ; n_c$) from these three preferred chromosomes.

Step 3. Compute $C^\phi_p$ by

$$C^\phi_p = C_{lp} + D_1.(C_{lp} - C_{mp}) + D_2.(C_{lp} - C_{hp})$$

Starting from the base point $C_{lp}$ and using two difference vectors, $D_1.(C_{lp} - C_{mp})$ and $D_2.(C_{lp} - C_{hp})$, the next evolutionary direction and the next evolutionary step-size can be determined by this parallelogram. The point $C^\phi_p$ can be then created along the evolutionary direction with the evolutionary step-size.

Step 4. $C_{op} = \max[\min(C^\phi_p, C_p^{MAX}), C_p^{MIN}]$, $p = 1, \ldots , n_c$. The value of $C_{op}$ must be kept within its set bounds.

Step 5. Evaluate the new fitness $\tilde{F}_{new}$ of the newly created output chromosome.

Step 6. If $F_{new} < F_h$, then go to next step; otherwise, go to Step 8.

Step 7. If $F_{new} = F_l = F_m$, add a random number $\sim [0,1]$ to $C^\phi_p$ and go to Step 4, then recomputed $F_{new}$; otherwise, go to Step 9.

A random number is added to prevent the algorithm from falling into a local optimum.

Step 8. Let $D_1 = D_1*(-0.5), D_2 = D_2*(-0.5)$, then go to Step 10.

Use the opposite direction and reduce the half step-size to search the new solution.
Step 9. Replace $C_{lp}$ by $C_{cop}$, if $F_{new} < F_i$. Replace $C_{mp}$ by $C_{op}$, if $F_i < F_{new} < F_m$. Replace $C_{hp}$ by $C_{op}$, if $F_m < F_{new} < F_h$, and go to Step 10.

To search a minimum extreme, use three cases of replacements mentioned above to choose the best three individuals, given as $Z_l$, $Z_m$, and $Z_h$ for the IEDO operation.

Step 10. If the last iterative loop of the IEDO is reached, then go to Step 11; otherwise, $T_s = T_s + 1$, and go to Step 2.

Step 11. Terminate the IEDO operation.

7.2.2. Reproduction, Crossover, and Mutation

Three preferred individuals generated by the IEDO are selected for reproduction. Reproduction probabilities of the three chosen individuals are set as follows: the first preferred unit 35%; the second preferred unit 25%, and the third preferred unit 15%. The remainder 25% of population is generated using the randomly created feasible individual. A binomial mutual crossover is adopted to raise the local diversity of individuals. For a small population (e.g. $N_p = 50$), the crossover probability is set to 0.9 which is enough to create new individuals and to avoid high diversity resulting in divergence of the population. The purpose of mutation is to introduce a slight perturbation to increase the diversity of trial individuals after crossover, preventing trial individuals from clustering and causing premature convergence of solution. The probability of mutation is set to 0.01.

7.2.3. Migration

A migration is included in the IGA to regenerate a newly diverse population, which prevents individuals from gradually clustering and thus greatly increases the amount of search space explored for a small population. The migrant individuals are generated based on the best ind
individual, $x^G_{i+1} = (x^G_{1+1}, x^G_{2+1}, ..., x^G_{n+1})$ by non-uniformly random choice. Genes of the $i^{th}$ individual are regenerated according to

$$
X^{G+1}_{k} = \begin{cases} 
X^{G+1}_{k} + \rho(X^{L}_{k} - X^{G+1}_{k}), & \text{if } r_1 < \frac{X^{L}_{k} - X^{G+1}_{k}}{X^{L}_{k} - X^{G+1}_{k}} \\
X^{G+1}_{k} + \rho(X^{U}_{k} - X^{G+1}_{k}), & \text{otherwise}
\end{cases}
$$

(7.2)

Where $k = 1, ..., n_c; i = 1, ..., N_p, r_i$ and $\rho$ are random numbers in the range of [0,1]. The migration may be performed if only the best fitness has not been improved for over 500 generations running, and the migrant population will not only become a set of newly promising solutions but also easily escape the local extreme value trap.

The procedure used in the optimization using improved genetic algorithm is shown in Fig.7.5. The problem of optimization of the turning process can be described as minimizing the surface roughness subject to a set of constraints as shown in equation (7.3).

In order to optimize the present problem using improved genetic algorithms (IGAs), the constrained optimization problem is stated as follows:

From the observed data for surface roughness, the response function has been determined using RSM and fitness function, defined as

Minimize,

$$
R_a = -4.89 + 2.49F - 38.0D + 0.599V + 3.27R - 5.38F \cdot D + 0.0140F \cdot V - 18.2F \cdot R + 0.0097D \cdot V + 15.8D \cdot R - 0.232V \cdot R + 80.5F^2 + 16.5D^2 - 0.00318V^2
$$

(7.3)
subject to

\[39.269 \text{ m/min} \leq V \leq 94.247 \text{ m/min}\]

\[0.059 \text{ mm/rev} \leq F \leq 0.26 \text{ mm/rev}\]

\[0.4 \text{ mm} \leq D \leq 1.2 \text{ mm}\]

\[0.4 \text{ mm} \leq R \leq 1.2 \text{ mm}\]

\[x_{\text{il}} \leq x_i \leq x_{\text{iu}}\]

where \(x_{\text{il}}\) and \(x_{\text{iu}}\) are the upper and lower bounds of process variables \(x_i\). \(x_1, x_2, x_3, x_4\) are the cutting speed, feed, depth of cut and nose radius respectively. In order to optimize the present problem using IGAs, the following parameters have been selected to obtain optimal solutions with less computational effort.

Maximum number of generations = 1000

Total string length = 50

Crossover probability (\(P_c\)) = 0.9

Mutation probability (\(P_m\)) = 0.01
7.3. Simulation Studies and Performance Evaluation

The possibility of a SS 420 machining optimization procedure using genetic algorithm is investigated in this work. The optimisation based on genetic algorithm has proved to be very
useful in dealing with discrete variables defined on a population of cutting condition obtained from the experiment. The search for the optimum was based on the minimization of an objective function. It was found that the GA can be a powerful tool in experimental machining optimization of scientific interest and large industrial applications. However, the optimization by GA technique requires a good setting of its own parameters, such as population size, number of generations, etc.

An improved evolutionary direction operator (IEDO) is embedded in GA to form the IGA so as to enhance GA’s computational efficiency. The IEDO can significantly reduce the effort in searching for the optimal solution because it enhances the local searching capability for GA. IGA is the algorithm based on mechanics of natural selection and natural genetics, which are more robust and more likely to locate global optimum and a local optimal trap inherent in mathematical programming methods can be overcome. The IGA code was developed using MATLAB. The genetic evolution histories are described in figure 7.6 for IGA and table 7.2 shows the minimum values of surface roughness with respect to input machining parameters.

Moreover, the proposed IGA approach has the following merits: simple concept; easy implementation; greater effectiveness than previous methods; better efficiency than the conventional genetic algorithm (CGA); robustness of algorithm; applicable to the larger-scale system; and the requirement for only a small population to prevent the dimensionality problem. The comparative results demonstrate that the proposed algorithm has the advantages mentioned above for solving the optimization problem.

7.4. Summary

Both CGA and IGA is applied for the optimization of machining problem. The proposed IGA provides better solutions than the conventional GA. The improved genetic algorithm incorporating a stochastic crossover technique and an artificial initial population scheme is developed to provide a faster search mechanism. The main advantage of the IGA approach is that the “curse of dimensionality” and a local optimal trap inherent in mathematical programming methods can be simultaneously overcome. The IGA equipped with an improved evolutionary direction operator and a migration operation can efficiently search and actively explore solutions.
Moreover, by incorporating all the improvements, it was found to be robust in providing optimum solution within a reasonable computation time and yield better solutions. Contrary to the dynamic programming, computation time of the proposed IGA is linearly proportional to the number of stages.

Table 7.2 – Output values of Improved genetic algorithm with respect to input machining parameters

<table>
<thead>
<tr>
<th>Machining Parameters</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed, F(mm/rev)</td>
<td>0.0755067</td>
</tr>
<tr>
<td>Depth of cut, D(mm)</td>
<td>0.564062</td>
</tr>
<tr>
<td>Cutting Velocity, V(m/min)</td>
<td>42.7725</td>
</tr>
<tr>
<td>Nose Radius, R(mm)</td>
<td>0.649459</td>
</tr>
<tr>
<td>Min. Surface Roughness, Ra(microns)</td>
<td>4.88498*10^{-14}</td>
</tr>
</tbody>
</table>
Fig. 7.6 Genetic evolution of IGA