CHAPTER 2

Tools and Techniques Used in
Machining Process Optimization

2.1  Response Surface Methodology

Experimental design is an important tool in machining operations for improving the performance of the machining process. It is a series of tests in which changes are made to the input variables of a process. The input variables are varied across the entire range of values permissible for operation. The changes in the output response are then observed and identified, Davim (2003). The initial step during optimization of machining processes is to develop a mathematical model of the required machining operation. The statistical experimental design mostly preferred in optimization experiments are termed as Response Surface Designs. RSM is a technique useful for correlating the control variables with machining criteria and the objective is to optimize the response. In addition to trials at the extreme level settings of the variables, response surface designs also contain trials in which one or more of the variables are set at the midpoint of the study range. Thus, the designs provide information on direct effects, pair wise interaction effects and curvilinear variable effects, Bhatti et al. (2011), Chiang et al. (2006).

If the response can be defined by a linear function of independent variables, then the approximating function is a first-order model. Lack of fit of the first-order model occurs when
the response surface is not a plane. If there is a significant lack of fit of the first-order model, then more highly structured model such as the second-order model are adopted in order to locate the optimum.

A second-order model is useful in approximating a portion of true response surface with parabolic curvature, Gopalakannan et al. (2012). The second-order model includes all terms in the first-order model, plus all quadratic terms like \( b_{ii}x_{iu}^2 \) and all cross product terms like \( b_{ij}x_{iu}x_{ji} \).

It is usually expressed as shown in equation 2.1 below,

\[
Y_u = b_0 + \sum_{i=1}^{n} b_i x_{iu} + \sum_{i=1}^{n} b_{ii} x_{iu}^2 + \sum_{i<j}^{n} b_{ij} x_{iu} x_{ju} + e_u
\]  \( \ldots 2.1 \)

where, \( Y_u \) is the corresponding response, \( X_{iu} \) are coded values of the \( i^{th} \) variables, \( b_0, b_i, b_{ii} \) and \( b_{ij} \) are the regression coefficients, \( e_u \) is experimental error of the \( u^{th} \) observation, \( n \) is number of independent variables. The second-order model is flexible, because it can take a variety of functional forms and approximates the response surface locally. Therefore, this model is usually a good estimation of the true response surface. The method of least squares can be applied to estimate the coefficients \( b_j \) in a second-order model.

**2.1.1 Experimental Design using Rotatable CCD**

A popular design available for fitting a second-order model is the central composite design (CCD). CCD is useful in RSM for building a second order (quadratic) model for the response variable without the requirement for a complete three-level factorial experiment. It consists of factorial points, central points and axial points.

CCD is often developed through a process of sequential experimentation. The number of center points “\( nc \)” at the origin and the distance “\( a \)” of the axial runs from the design center are two
parameters in the CCD design. Figure 2.1 illustrates the graphical view of a central composite design for $q = 2$ factors.

CCD can run in incomplete blocks. A block is a set of relatively homogeneous experimental conditions so that an experimenter divides the observations into groups that are run in each block. An incomplete block design may be conducted when all treatment combinations cannot be run in each block. In order to protect the shape of the response surface, the block effects need to be orthogonal to treatment effects. This can be done by choosing the correct "a" and "nc" in factorial and axial blocks. Also, "a" and "nc" can be chosen so that the CCD is not blocked. If the precision of the estimated response surface at some point $x$ depends only on the distance from $x$ to the origin and not on the direction, then the design is said to be rotatable. When the rotatable design is rotated about the center, the variance will remain same. Since the reason for using response surface analysis is to locate unknown optimization, it makes sense to use a rotatable design that provides equal precision of estimation of the surface in all directions.
The central composite design is popular of the many classes of RSM designs due to the following properties:

A CCD can be run sequentially. It can be partitioned into two subsets of points; the first subset estimates linear and two-factor interaction effects while the second subset estimates higher order effects. The second subset need not be run when analysis of the data from the first subset points indicates the absence of significant curvature effects.

CCDs are very efficient, providing much information on experiment variable effects and overall experimental error in a minimum number of required runs.

CCDs are very flexible. The availability of several varieties of CCDs enables their use under different experimental regions of interest and operatability.

Besides this design is rotatable. The rotatability and the small number of necessary experiments make CCD design very well suited for estimating the coefficients in a second order model. If the distance from the center of the design space to a factorial point is ±1 unit for each factor, the distance from the center of the design space to an axial point is ±α with |α| > 1. The precise value of α depends on certain properties desired for the design and on the number of factors involved. Similarly, the number of center point runs the design is to contain also depends on certain properties required for the design.

2.2 Genetic Algorithm

2.2.1 Biological Analogy

In the biological world, chromosomes are strings of DNA which serves as a model for the whole living organism. A chromosome consists of genes which are blocks of DNA and it is the smallest
unit which exhibits all the properties of living organisms. Each gene encodes a trait. During reproduction, chromosomes of the parents are utilized to reproduce chromosomes of the offspring. This evolution is accomplished through the operations of crossover and mutation. Thus, the offspring chromosomes incorporate major chunks of genetic material coming from both the parents that is an inheritance as well as information exchange between the parent chromosomes. Hence, each new offspring produced inherit traits which make them more adaptable to the changing environment.

The inspiration for GAs came from nature and survival of the fittest. In a population, each individual has a set of traits that determine how well suited it is to the environment. Survival of the fittest implies that the ‘fitter’ individuals are more likely to survive and have greater probability of passing their ‘good’ features to the next generation. In biological reproduction, if the best features of each parent are inherited by their offspring, a new individual will be created that should have an improved probability of survival. This is the process of evolution. The DNA can be thought of as a string of genes, with each gene or combination of genes representing a particular feature. Reproduction is the ‘crossover’ of two DNA strings to produce a new offspring that has genes from both parents. Mutation can also occur where a particular gene is not an exact copy of either parent.

Analogous to the chromosomes, GA uses strings which are feasible individual solution point for a given optimization problem, Goldberg (1989). The crucial step in GA is the breeding process. Information is exchanged across the population members during the course of breeding. Breeding comprises of a two step process which are crossover and mutation. During crossover, the two strings generated by the encoding process are laid side by side and a cutting point is randomly selected along the length of the defining strings. The string fragments located right of
the cutting point are then interchanged and spliced onto the original fragments located left of the cutting point. The second step is mutation. For the string produced by the crossover process, a few randomly selected digits are replaced by a randomly selected new digit value. The resulting fragments are then decoded into two pairs whose fitness are then evaluated simply by computing the functional value.

Both crossover and mutation operations involve purely stochastic components, such as the choice of cutting point, site of mutation and new value of the mutated digit. This is where the variability needed to sustain the GA is obtained. The encoding/decoding process is just one of many possible such schemes.

### 2.2.2 GA Building Blocks

GA is designed to search for and breed good solutions called building blocks to a problem in a highly parallel fashion. Holland (1975) invented the idea of schemas/schemata to formally conceptualize the notion of ‘building blocks’. Each schema is a similarity template for a bit string of length \( l \), which describes a subset of strings with similarities at certain string positions. The schemata are comprised of ones and zeros defining the bits and an asterisk ‘*’ that act as wildcards within the string. Therefore, the schema can be represented as below

\[
H = 1\,*\,*\,*\,0\,*
\]

The above schema denotes all bit strings that begin with one and have a 0 at the 5\(^{th}\) bit position. Examples of bit strings that satisfy this condition are 111001, 100001, 111100, 111101 etc. These are called instances of the schema.

Other traits of a schema include the ‘defined bits’ and ‘defining length’. The defined bits are the number of bits defined within the schema. In the above example, there are 2 defined bits. The defining length is the distance between the two most outer defined bits. In the above example,
the defining length is 4. The motivation for considering important similarities is to get more information to help guide the search process.

During reproduction more highly fit strings have higher probabilities of selection. Thus, there is always a supply of an ever increasing number of samples to the observed best similarity patterns. However, crossover and mutation are to be exercised to produce new strings. Crossover does affect if it cuts the schema. Considering these two schemata 1****0 and **111*, the first is likely to be disrupted by crossover than the second. Mutation at normal does not disrupt a particular schema. Thus, it can be concluded that highly fit, short defining length schemata or building blocks are propagated by giving exponentially increasing samples to the observed best. All of this iteration goes in parallel with no special record or memory other than the population of ‘n’ strings. This processing advantage called implicit parallelism is unique to GA, Goldberg (1989).

2.2.3 Operation of Genetic Algorithm

The process involved in GA optimisation problems is based on that of natural evolution and broadly works as follows,

1. Randomly generate an initial population of potential solutions.
2. Evaluate the suitability or ‘fitness’ of each solution.
3. Select two solutions biased in favour of fitness.
4. Crossover and mutation performed on the solutions at a random point on the string to produce two new solutions.

Figure 2.2 show a general flow diagram of operation of a GA.
2.2.4 Different Operators in GA

(1) Population Size

Population size is the total number of candidate solutions in any one generation. In natural evolution the total population size is governed by what is sustainable by the environment and similarly in GAs the larger the population size the more computationally intensive is the search. In nature, the bigger the gene pool the more diverse is the genetic makeup of the population. The advantage of this diversity is that there will be no dominant gene that may be susceptible to a particular disease and hence result in the elimination of the whole species. However, with large populations, the search for the global optimal solution can be a slow process. Again if the
population size is small, then a strong individual quickly becomes dominant and the diversity of the gene pool is restricted. The chance of evolving the global optimum is limited and would depend on mutation introducing new genes to diversify the search. As new solutions are generated it is common to keep the population size constant by eliminating individuals. The advantage computers have over nature is that good individuals do not have to die and can be retained for indefinite reproduction. The retention of certain fit individuals is known as ‘elitism’.

(2) Encoding

In optimisation problems a set of parameters is sought that will give the best solution to a particular problem. In order to implement a GA these parameters must be encoded into a string so that crossover and mutation can be applied. GAs work with a coding of the parameter set and not the parameter themselves. Coding similarities are exploited by GA to help guide the search process. As a result they are largely unconstrained by limitations like continuity, derivative existence, unimodality etc.

(i) Binary encoding

In binary encoding, every chromosome is a string of bits (0 or 1). Figure 2.3 shows an example of chromosomes with binary encoding.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1011001011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>111111000</td>
</tr>
</tbody>
</table>

Figure 2.3  Example of chromosomes with binary encoding

(ii) Permutation encoding

In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence. Figure 2.4 exhibits examples of chromosomes with permutation coding.
(iii) Value Encoding

In value encoding, every chromosome is a string of some values. Values can be anything connected to problem. Figure 2.5 shows the example of chromosomes with value encoding.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1.2324 5.3243 0.4556 2.3293</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>ABDJEIFJHDIERJFDLDFLF</td>
</tr>
<tr>
<td>Chromosome C</td>
<td>(back), (back), (right), (forward),</td>
</tr>
</tbody>
</table>

Figure 2.5 Example of chromosomes with value encoding

(iv) Tree encoding

In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. Figure 2.6 portrays the example of chromosomes with tree encoding.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>$(+ x (/ 5 y))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>$(do until step wall)$</td>
</tr>
</tbody>
</table>

Figure 2.6 Example of chromosomes with tree encoding
(3) Evaluation of Fitness function

Assume a problem of maximizing the function \( f(x) = x^2 \), where ‘x’ is permitted to vary between 0 and 31. The variable ‘x’ is coded as a binary unsigned integer of length 5. In a base 2 arithmetic, there are only two digits to work with, 0 and 1, and as an example the number 10011 decodes to the base 10 number as,

\[
1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 2 + 1 = 19
\]

With a five bit (binary digit) unsigned integer, numbers between 0 (00000) and 31 (11111) can be obtained. A population of size 4 is selected randomly by tossing a fair coin 20 times. Table 2.1 shows the decoded ‘x’ values along with the fitness or objective function values \( f(x) \).

<table>
<thead>
<tr>
<th>String No.</th>
<th>Initial Population</th>
<th>‘x’ value</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 1 1 0 1</td>
<td>13</td>
<td>169</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 0 0</td>
<td>24</td>
<td>576</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 0 0</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 1 1</td>
<td>19</td>
<td>361</td>
</tr>
</tbody>
</table>

Decoding the third string (01000), there is a single one in the \( 2^3 = 8 \)’s position. Hence for string 01000, obtained value of ‘x’ = 8. To calculate the fitness function, the value of ‘x’ is squared obtaining the resulting fitness value \( f(x) = 64 \). Other ‘x’ and \( f(x) \) values are obtained similarly, Goldberg (1989).

(4) Selection

The procedure for choosing individuals (parents) on which to perform crossover in order to create new solutions is known as selection. The logic is that the fitter individuals are more prominent in the selection process, with the hope that the offspring they create will be even fitter still. In roulette wheel, each individual is assigned a slice of a wheel, the size of the slice being
proportional to the fitness of the individual. The wheel is then spun and the individual opposite the marker becomes one of the parents. In tournament selection several individuals are chosen at random and the fittest becomes one of the parents.

It is a process of selecting individual strings according to their objective function values. The objective maybe a measure of some parameters associated with machining processes. According to Darwin’s evolution theory, the best ones should survive and create new offspring. A few methods to select the best chromosomes are, e.g. roulette wheel selection, rank selection, steady state selection and elitism are briefed as follows:

(i) Roulette wheel selection

In this type of selection, every chromosome has its place big accordingly to its fitness function as displayed in Figure 2.7. Selection of chromosomes is random. Hence, chromosomes with high fitness function values like chromosome 1 have more chances to be selected.

![Figure 2.7] A roulette wheel

However, when the fitnesses differ very much for example, if the best chromosome fitness is 90% of the entire roulette wheel, then the other chromosomes will have very few chances to be selected.

(ii) Rank selection

Rank selection permits all the chromosomes to have a chance to be selected. It first ranks the population and then every chromosome receives fitness from this ranking. The worst will have
fitness 1, second worst 2, soon and the best will have fitness N (number of chromosomes in population). Figures 2.8 and 2.9 depict how the situation changes after changing fitness to order number.

![Figure 2.8](image)

**Figure 2.8** Situation before ranking

![Figure 2.9](image)

**Figure 2.9** Situation after ranking (graph of order numbers)

But, this method can lead to slower convergence, because the best chromosomes do not differ so much from the other ones.

(iii) Steady-state selection

This selection process is based on the idea that a big part of chromosomes should survive to the next generation. During the process of every generation a few good chromosomes with high fitness values are selected for creating a new offspring. Some bad chromosomes having low fitness values are removed and the new offspring is placed in their place. The rest of the population survives to give the new generation.
(iv) **Elitism**

In this method, the best chromosome (or a few best chromosomes) is first copied to new population. The rest is done in the traditional way. Elitism increases the performance of GA by preventing the loss of best found solution.

(5) **Crossover**

Along with mutation, crossover is the operator that creates new candidate solutions. A position is randomly chosen on the string and the two parents are ‘crossed over’ at this point to create two new solutions. Multiple point crossover is where this occurs at several points along the string. A crossover probability \(P_c\) is often given which enables a chance that the parents descend into the next generation unchanged. Crossover is performed in two steps. First, the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes crossing over as follows. An integer position \(k\), say \(k = 5\), as shown in Figure 2.11 along the string is selected uniformly at random between 1 and the string length less one \([1, l-1]\). Thereafter two new strings are created by swapping all the characters between positions \(k + 1\) and \(l\) inclusively.

(i) **Single point crossover**

In this method, a single crossover point is selected. The binary string from the beginning of chromosome to the crossover point is copied from the first parent and the rest is copied from the second parent.

\[
\text{Parent A} \quad + \quad \text{Parent B} \quad = \quad \text{Offspring}
\]

\[
11001011 \quad + \quad 11011111 \quad = \quad 11001111
\]

*Figure 2.10 Single point crossover*
(ii) Two point crossover

Here two crossover points are selected. The binary string from the beginning of chromosome to the first crossover point is copied from the first parent, the part from the first to the second crossover point is copied from the second parent and then the rest is copied from the first parent. Figure 2.11 depicts the mechanism of two point crossover.

![Two point crossover diagram](image1)

(iii) Uniform crossover

In this type of crossover, the bits are randomly copied from the first or from the second parent. Figure 2.12 shows the mechanism of uniform crossover.

![Uniform crossover diagram](image2)

(iv) Arithmetic crossover

This method uses some arithmetic operation to make a new offspring. Figure 2.13 exhibits the basic mechanism of arithmetic crossover.

![Arithmetic crossover diagram](image3)
(6) Mutation

After crossover, each bit of the string has the potential to mutate, based on a mutation probability ($P_m$). Mutation is needed because even though reproduction and crossover can effectively search and recombine extant notions, occasionally, there is a probability of losing some potentially useful genetic material (1’s or 0’s at particular locations). In GA, the mutation operator protects against such an irrecoverable loss. Mutation with small probability is the occasional random alteration of the value of a particular string position. In the binary coding, this is simply changing a ‘1’ to a ‘0’ and vice versa. Mutation often prevents premature loss of important notions.

2.2.4 Parameters used in GA

There are two basic parameters in GA i.e. crossover probability and mutation probability. Crossover probability determines the occurrence of crossover. For a 100% crossover probability, all the offspring will be made by crossover. If it is 0%, the whole new generation will be made from exact copies of chromosomes of the parents. Crossover is performed in a hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better.

Mutation probability defines how often the parts of chromosome will be mutated. If there is no mutation, the offspring will be taken after crossover without any change. If mutation is performed, part of the chromosome will be changed. The whole chromosome will be changed for a mutation probability of 100% and likewise there is no change for a mutation probability of 0%. Mutation prevents GA from falling into local extreme, but it should not occur very often, else the GA will in fact change to random search.

The population size is also an important parameter while operating GA. Population size defines the number chromosomes in the population in one generation. A very few chromosomes in the
population leads to GA having a very few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA will slow down. Research has shown that after some limits which depend mainly on encoding and the given problem, it is not useful to further increase the population size as the speed of GA remains stagnant.

There are a number of stopping criteria to stall further operation of GA 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, Eiben et al. (2011), Mishra et al. (2011), Yusup et al. (2012).

2.3 Multi Objective Optimization

2.3.1 Multi-objective Genetic Algorithm

GA is a suitable optimization technique to solve multiobjective optimization problems since the search process in this approach deal simultaneously with a set of possible solutions called population. This feature of GA enables to find several members of the Pareto optimal set in a single run of the algorithm which gives it leverage over traditional mathematical programming techniques. The traditional techniques are less preferred for having to perform a series of separate runs to obtain the Pareto optimal set. This ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. Additionally, GA is less susceptible to the shape or continuity of the Pareto front i.e., they can easily deal with discontinuous or concave Pareto fronts. Most multi-objective GA does not

The first multi-objective GA, called Vector Evaluated Genetic Algorithms (or VEGA), was proposed by Schaffer (1985). In this method, the selection mechanism of GA is modified so that at each generation a number of sub-populations are generated by performing proportional selection according to each objective function in turn. Thus, for a problem with k objectives and a population size of M, k sub-populations of size M/k each would be generated. These sub-populations are shuffled together to obtain a new population of size M which is further processed by GA by applying the crossover and mutation operators in the usual way.

Pareto-based Techniques method is suggested to solve the problems with Schaffer’s VEGA. It uses non dominated ranking and selection to move the population towards the Pareto front. This approach requires a ranking procedure and a technique to maintain diversity in the population else the GA will tend to converge to a single solution because of the stochastic noise involved in the process.

Although several variations of Goldberg’s proposal have been proposed several researchers have used what is called “pure Pareto ranking”. The idea in this case is to follow Goldberg’s (1989) proposal as stated in his book. Non dominated sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb (1994) is based on several layers of classifications of the individuals. Non-dominated individuals are allotted a certain dummy fitness value and then are removed from the population. The process is repeated until the entire population has been classified. To maintain the diversity of the population, classified individuals are shared in decision variable space with their dummy fitness values. An improved version NSGA-II uses elitism and a crowded
comparison operator that keeps diversity without specifying any additional parameters, Zio and Bazzo (2011).

### 2.3.2 Pareto Optimal Solution

Real-life problems like the optimization of machining processes have number of objectives under consideration which often conflict with each other. At such circumstance, optimizing with respect to a single objective often results in degraded results with respect to the other objectives. Due to this adverse effect, it is impossible to search for a perfect multi-objective solution that simultaneously optimizes all the objective function. A practical solution to such multiobjective problem would be to investigate a set of solutions, each of which satisfies the objectives at an acceptable level and without being dominated by any other solution. The set of all feasible non-dominated solutions in the solution space is referred to as the Pareto optimal set, Su et al. (2008).

A Pareto optimal solution is not dominated by any other solution in the solution space. Thus, there is no scope for a Pareto optimal solution to be improved with respect to any objective without worsening at least one other objective. For a given Pareto optimal set, the corresponding objective function values in the objective space is referred as the Pareto front.

Thus, restricting attention to the set of choices that are Pareto-efficient, an operator can make tradeoffs within this set while using a machining process rather than considering the full range of every parameter. A Pareto front is shown in figure 2.14. Objective $f(1)$ is mapped along $y$-axis and objective $f(2)$ is mapped along the $x$-axis. All the points between these two axes are feasible solution points but the Pareto optimal solution is given by the points joined by the curved line.

The prime objective of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. However, for a large number of Pareto optimal solutions it is impossible to identify the entire Pareto optimal set due to its enormous size. Therefore, a more feasible
approach to multi-objective optimization is to investigate a set of solutions that represent the Pareto optimal set as much as possible, Shajan et al. (2005), Srinivas et al. (1994).

While searching the solution set for problems with conflicting objectives, a multi-objective optimization approach should achieve the following three conflicting goals,

(i) The best-known Pareto front should be as close as possible to the true Pareto front. Ideally, the best-known Pareto set should be a subset of the Pareto optimal set.

(ii) Solutions in the best-known Pareto set should be uniformly distributed and diverse over the Pareto front. This is required in order to provide the decision maker a true picture of trade-offs.

(iii) The best-known Pareto front should capture the whole spectrum of the Pareto front. This requires investigating solutions at the extreme ends of the objective function space.
2.3.3 Techniques in Multi-Objective GA

(1) Fitness Functions

Pareto-ranking approaches utilize the concept of Pareto dominance while evaluating fitness or assigning selection probability to solutions. The population is ranked according to a dominance rule. Each solution is assigned a fitness value based on its rank in the population and not its actual objective function value. Wherein all objectives are assumed to be minimized, a lower rank corresponds to a better solution, Schaffer (1985).

(2) Diversity

A diverse population is to be maintained in multi-objective GA to obtain solutions which are uniformly distributed over the true Pareto front. Without any preventive measure to maintain population diversity, the population tends to form relatively few clusters in multi-objective GA. This phenomenon is called genetic drift. This anomaly is overcome by fitness sharing.

Fitness sharing aims to encourage the search in unexplored sections of a Pareto front by artificially reducing fitness of solutions in densely populated areas. This is achieved by identifying densely populated areas thereafter a fair penalty method is used to penalize the solutions located in such areas. In this procedure, $\sigma_{\text{share}}$ defines a neighborhood of solutions in the objective space. The solutions in the same neighborhood contribute to each other’s niche count. Hence, a solution in a crowded neighborhood will have a higher niche count reducing the probability of selecting that solution as a parent. As a result, niching limits the proliferation of solutions in one particular neighborhood of the objective function space. Deb and Goldberg (1993) reported that fitness sharing on the objective function space usually performs better than one based on the decision variable space.
One of the disadvantages of the fitness sharing based on niche count is that the user has to select a new parameter $\sigma_{\text{share}}$. To address this problem, Deb and Goldberg (1994) and Fonseca and Fleming (1998) developed a systematic approach to estimate and dynamically update $\sigma_{\text{share}}$. Though computational effort is required to calculate niche counts yet the benefits of fitness sharing surpass the burden of extra computational effort in many applications.

(3) **Crowding Distance**

The crowding distance approach aims to obtain a uniform spread of solutions along the best known Pareto front without using a fitness sharing parameter. The main advantage of the crowding approach is that a measure of population density around a solution is computed without requiring a user-defined parameter, Altenberg (1994). In the NSGA-II, the crowding distance measure is used as a tie-breaker for the next selection phase. Among two randomly selected solutions in the same non-dominated front, the solution with a higher crowding distance is selected. Otherwise, the solution with the lowest rank is selected, Zio and Bazzo (2011).

(4) **Elitism**

In multi-objective GA, all non-dominated solutions discovered are considered as elite solutions. However, implementation of elitism is not as straightforward as in single objective optimization. This is mainly because of the large number of possible elitist solutions. Multi-objective GA using elitist strategies tend to outperform their non-elitist counterparts. Multi-objective GA uses the following two strategies to implement elitism (i) maintain the elitist solutions in the population, and (ii) store elitist solutions in an external secondary list and reintroduce them to the population, Lianga et al. (2011).
(5) **Constraint Handling**

Optimization problems often include constraints that must be satisfied. Multiobjective GA use the following different constraint handling strategy: (i) discarding infeasible solutions, (ii) if possible, customizing genetic operators to always produce feasible solutions, and (iii) repairing infeasible solutions.

Comparative studies on multi-objective GA agree that elitism and diversity preservation mechanisms improve performance of GA. However, implementing elitism and diversity preservation strategies usually require substantial computational effort and computer memory. Also, the evaluation of objective functions may take considerable time in real-life problems. Therefore, advanced data structures are introduced to reduce execution time and resource requirement of multi-objective GA. One of the trends in this avenue is parallel and distributed processing.