4th International Conference on Mechanical Engineering, December 26-28, 2001, Dhaka, Bangladesh/pp.VII 227-233

OPTIMIZATION OF PROCESS PLANNING PARAMETERS FOR ROTATIONAL COMPONENTS BY GENETIC ALGORITHMS

Nafis Ahmad* and A.F.M. Anwarul Haque
Department of IPE, BUET, Dhaka

Abstract—In CAPP systems process parameter optimization is one of the key areas for research and development. Traditional techniques have very limited scope because of the complexity of the optimization problem. Due to the rapid development of computer technology Genetic Algorithms (GAs), which are robust search algorithm, have been found to be suitable and efficient tools for optimization in such cases. In this work process planning parameters for machining rotational components are optimized by a Genetic Algorithm Optimization Toolbox developed in Matlab environment. Here machining time is considered as the objective function and constraints are machine capacity, limits of feed rate, depth of cut, cutting speed etc. Machining time is minimized through a series of generations while some genetic operators are applied at each generation. The result of the work shows how a complex optimization problem is handle by a genetic algorithm and converges very quickly.

Keyword- CAPP, GA, Optimization

INTRODUCTION

Optimization of process planning is one of the foremost targets of Manufacturing Systems. Numbers of research works are performed for generating optimum process plan. The optimum process plan may be on the basis of time or cost or on the basis of some weighted combination of these two. Tool selection, machine selection, process selection and tool path selection, process parameter selection are the most important areas for optimization in process planning. Process parameter optimization is the final stage of a CAPP system. Determination of optimum parameters is one of the vital stages of process planning since the economy of machining operation plays the most important role in increasing productivity and competitiveness. Genetic algorithm is one of the most efficient tools for optimization of such problems. This paper presents the application of GA in process planning parameters optimization.

GA AND OTHER SEARCH ALGORITHMS

Many works have so far been done to optimize these parameters by using different optimization techniques like goal programming, multistage dynamic programming, linear programming, geometric programming, branch and bound algorithm etc. But all of them face great difficulties when the number of variables increases, because the problem becomes combinatorially explosive and hence computationally complex [1]. Different researchers used different techniques to optimize process parameters but all of those techniques have their own limitations.

Direct search methods include function evaluation and comparisons only. Gradient search methods need values of function and its derivatives, and their computerizations are also problematic. They are more difficult than the direct search methods, but they can yield more accurate for some computational efforts.

Derivative-based mathematical optimizations are not manageable for optimizing functions of discrete variables. Dynamic programming that may be applied to problems whose solution involves a multistage decision process, can handle both continuous and discrete variables. Contrary to many other optimization methods it can yield a global optimum solution. However if the optimization problem involves a large number of independent parameters with a wide range of values (as in the case of optimization of cutting parameters), the use of dynamic programming is limited. As the numbers of variables and constraints increases, the optimum has a tendency to grow flatter with less probability that the realizable optimum will be a mathematical optimum, and hence computational effort increases considerably.

Geometric programming is a useful method that can be used for solving nonlinear problems subject to nonlinear constraints, especially if the objective function to be optimized is a polynomial with fractional and negative exponents, while the constraints may be incorporated in the solution techniques. It is more powerful than other mathematical optimization techniques when the problem is restricted by one or two constraints. However if the degree of difficulty increases, the formulated problem might be more complicated than the original problem. Geometric programming can only handle continuous variables.

*Email: nafis@IPE.BUET.EDU

The solution to the optimization problems, which includes real value variables, can be obtained using numerous methods. There is no efficient all-purpose optimization method available for nonlinear programming problems like process parameter optimization. The computational time and cost involved in the determination of optimal parameters commonly depends on the complexity or simplicity of the model. Some models can produce accurate solutions by making rigorous computation, which is not economic in terms of computation time and cost. Sometimes the solution from these models may not be optimal. Some other models may develop solutions far from the optimum in a fast manner. Therefore a compromise between the high accuracy of a rigorous solution and low accuracy of an oversimplified solution should be made.

Genetic Algorithms (GAs) are robust search algorithms that are based on the mechanics of natural selection and natural genetics. They combine the idea of "survival of the fittest" with some of the mechanics of genetics to form a highly effective search algorithm. Genetic algorithms belong to a class of stochastic optimization techniques known as evolutionary algorithms. Among the three major types of evolutionary algorithms (genetic algorithms, evolutionary programming, and evolution strategies) genetic algorithms are the mostly widely used. GAs are most often used for optimization of various systems, especially complex problems such as those involving manufacturing systems analysis.

GA AND NATURAL EVOLUTION PROCESS

Genetic Algorithms (GAs) are search strategy which are able to search very large solution spaces efficiently by providing a concise computational cost, since they use probabilistic transaction rules instead of deterministic ones. They are easy to implement and are increasingly used to solve inherently intractable problems quickly. Although GAs are heuristic procedures themselves, they test a wealth of samplings from different regions of the search space for fitness simultaneously, and sort out and exploit regions of interest very quickly [1].

The idea behind genetic algorithm is based on the natural evolution phenomena. Rabbits are taken as example: at any given time there is a population of rabbits. Some of them are faster and smarter than the other rabbits. These faster, smarter rabbits are less likely to be eaten by foxes, and therefore more of them survive and make more rabbits. Of course, some of the slower, dumber rabbits will survive just because they are lucky. This surviving population of rabbits starts breeding. The breeding results in a good mixture of rabbits' genetic material: some slow rabbits breed with fast rabbits, some fast with fast, some smart rabbits with dumb rabbits, and so on. As a resulting baby rabbits will (on average) be faster and smarter than those in the original population because more faster, smarter rabbits survived the foxes. (It is a good thing that the foxes are undergoing a similar process- otherwise the rabbits might become too fast and smart for the foxes to catch any of them). In the similar fashion, in an artificial genetic algorithm, a crude population is refined through a series of generations while some genetic operators work on the population.

TYPICAL GA PROCEDURE

GAs start with an initial set of random solutions called the population. There is no strict rule to determine the population size. Population sizes of 100-200 are common in GA research. Through the steps described below, the population will eventually converge. Larger population size ensures greater diversity but requires more computer resource. Once the population size is chosen, the initial population is randomly generated.

If the population has 20 strings of 10 bits then 10.20=200 bits must be set to either 0 or 1. The computer sets the value of 200 bit positions with simulated coin toss. Any string with decimal equivalent greater than the maximum limits is discarded and replaced with another randomly chosen string that meets the constraint. For example, range of a parameter is from 1000 to 1100 and an individual of the population is randomly taken as 0111 or 1101, which falls outside the above range. This individual will be discarded and another individual will be random taken and checked whether it falls within the range. This process continues until all the individual of the population are within the specific range.

Once the chromosomes are coded as bit strings, the genetic algorithm manipulates these strings using three genetic operators —reproduction, crossover, and mutation. The chromosomes are said to evolve through successive generations. In each generation, the fitness of each chromosome is evaluated; chromosomes with a higher fitness value are more likely to be selected.

Reproduction takes the current population of bit strings (that have already been evaluated and given a fitness value), makes copies of the strings with better fitness values, and places these strings in a "mating pool". The reproduction operator may be implemented in algorithmic form in different ways. The easiest way is to create a biased **roulette wheel** slot sized in proportion to each current string in the population. Another selection technique is normalized geometricselection which is a ranking selection function based on the normalized geometric distribution.

After the selection, the strings are paired up, and a percentage of the pairs trade parts of their strings. This is known as **crossover**. Different crossover techniques are used in GA. Simple crossover involves two parents and crossover points are selected randomly. If two parents to be used for generating new chromosomes are

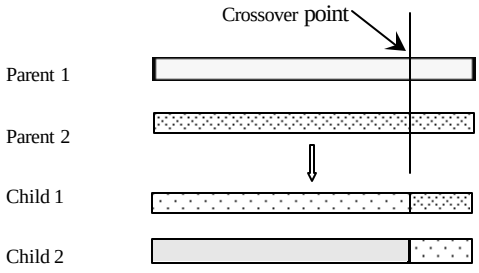

Figure 7.2 : Crossover diagram

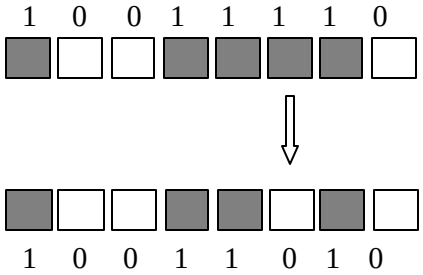

Figure 7.3 : Mutation

{Parent 1: *0 1 1 0 1* } and {Parent 2: 1 0 1 1 0 } and if a crossover point is chosen randomly as 4 the following children will be produced: {Child 1: *0 1 1 0* 0} and {Child 2: 1 0 1 1 *1*}

Here first four digits of child-1 (i.e. *0 1 1 0*) are from parent-1 and the rest of the digits (i.e. 1) from parent-2. Similarly first 4 digits of child-2 are from parent-2 and the rest of the digits from parent-1. Figure 1 is the graphical representation of the crossover function.

These newly formed strings by crossover operator are then subjected to a random screening, where random bits in random strings are picked and modified. This is known as **mutation**. Mutation introduces random variations into the population. It zaps a '0' to a '1' and vise versa in a binary string. Each bit position for every member of the population is examined. The computer randomly decides whether mutation should occur or not. Mutation is usually performed with low probability; otherwise it will defeat the order building being generated through selection and crossover. Mutation attempts to bump the population gently onto a slightly better course.

As an example consider a string as shown in figure 2. Shaded and clear boxes represent two different options for a bit position in a string:'1' and '0'. If the sixth position of the string is randomly chosen as mutation point, '1' at that position will be replaced by '0' by mutation operator.

After this step, the remaining strings form the next generation of bit strings. They are evaluated, given a fitness value, and again subjected to reproduction, crossover, and mutation. This combined process of exploiting knowledge about a search space (reproduction) and exploring a search space (crossover, mutation) is what drives the performance of a genetic algorithm. Over time, bad bit strings disappear from a population, while good bit strings live on and reproduce with other good strings to form even better strings.

CODING THE CROMOSOMES

The individuals comprising the population are known as chromosomes. In most genetic algorithm applications, the chromosomes are coded as a series of zeroes and ones, or a binary bit string. This usually involves discretization of the search space into a certain number of points that can be represented by a certain length of bit string. For binary bit strings, this would mean that a search space would need to be split into 2^n points, represented by a bit string of length n. For example a 3 bit string may be maximum binary number of 3 ones: 111. Its decimal equivalent is 2^3-1=7. So the search space is from 0-7 (i.e. 000. 001, 010, 011, 100, 101, 110 and 111 in binary system). Encoding schemes transform points in a parameter space into bit string representation. For example a point (11,6,9) in a three dimensional parameter space can be represented as a concatenated binary string, in which each coordinate value is encoded as a ***gene*** composed of four binary bits using binary coding.

1011 0110 1001
11 6 9

In process planning parameter optimization where real valued variables are involved, the variables are controlled simultaneously within their ranges.

PRPBLEM STATEMENT

Rotational parts, which have surfaces symmetric to the part axis, are usually machined by lathes machine. Depending on the required surface finish, rough turning or other finishing operations are required. But initially rough turning operation creates the shape of the surface from the blank by removing a materials and the major part of the machining time is usually required for rough turning operation. For this reason cutting parameters such as feed rate, depth of cut, cutting speed etc. are optimized only for rough turning operation.

Again, a machine shop may have several lathes with different power and rpm. So, it is also necessary to identify the machine and rpm that will require minimum time for machining a specific surface. Though total time includes machinig time, setup time, approach and overtravel of the cutting tool, in most of the cases, machining time is responsible for the major part of total cost. Other cost is not as significant as machining time. So, in this optimization problem the cutting parameters are determined by minimizing the machining time.

PROBLEM FORMULATION

It is already mentioned that only cylindrical surfaces are considered in this work. These types of surfaces are machined by turning operation to attain the required shape of the surfaces. An example is presented in figure 4. For a horizontal cylindrical surface machining time(T_m) for turning operation depends on

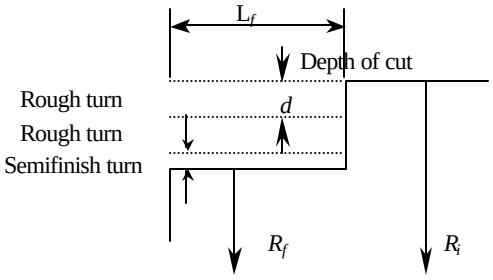

Figure 3 Rough turning of a

the total length of the surface (L) to be machined, feed rate (f) and rotational speed (N_w) of the work piece [3] i.e. machining time is:

$$T_m = \frac{L}{f \times N_w} \qquad (1)$$

If the length of a feature is L_f and number of pass is n_{pass} then

$L = L_f \times n_{pass}$. So,

$$T_m = \frac{L_f \times n_{pass}}{f \times N_w} \qquad (2)$$

In figure 3 depth of cut, initial radius and final radius are denoted by d, R_i and R_f respectively. Therefore,

Total cut = $R_i - R_f$

Semifinish cut = *Remainder of* $((R_i - R_f), d)$

$n_{pass} = \{(R_i - R_f) - \text{Remainder of } ((R_i - R_f), d)\}/d \qquad (3)$

To minimize the rough turning, the material removal rate should be as high as possible. As material removal rate is proportional to depth of cut, feed rate and cutting speed, these parameters should be increased for a higher material removal rate. But these parameters cannot be increase indefinitely due to limitation of maximum allowable force on the cutting tool and also maximum power limit of the machine tool in some cases. Power (P), and cutting force (Fc) can be calculated by equations 4 and 5 [3]. These two values are checked against their limits (i.e. P_{max} and F_{max}) when depth of cut, feed rate and cutting speed is optimized.

Power, $P = F_c \times V \qquad (4)$

Cutting force, $F_c = C_f \times f^a \times d^b \qquad (5)$

Here C_f, a, b are constants. Here V, f and d are independent variables.

Rough turning operation are usually performed at some low cutting speed with high depth of cut and high feed rate as the metal removal rate is more important than surface finish.

From the machine database we can collect the available spindle speeds. The effective limits of rpm of the spindle is chosen according to the following equation,

$N_{max} = Max\ (available\ speeds) \qquad (6)$

$N_{min} = Min\ (available\ speeds) \qquad (7)$

If the actual rpm of the spindle is N_w, can be calculated by equation,

$$N_w = \frac{1000 \times V}{2 \times p \times R}$$

This rpm is checked against the limits (N_{min} and N_{max}) during the optimization process.

As a result the optimization model becomes

$$\text{Minimize, } T_m = \frac{L_f \times n_{pass}}{f \times N_w} \qquad (8)$$

Subject to

$f_{min} < f < f_{max} \qquad (9)$

$V_{min} < V < V_{max} \qquad (10)$

$d_{min} < d < d_{max} \qquad (11)$

$F_c < F_{max}$ *i.e.* $C_f \times f^a \times d^b < F_{max} \qquad (12)$

$N_{min} < N < N_{max}$ *i.e.* $N_{min} < \frac{1000 \times V}{2 \times p \times R} < N_{max} \qquad (13)$

$P_m < P_{max}$ *i.e.* $C_f \times f^a \times d^b \times V < P_{max} \qquad (14)$

This is a nonlinear optimization problem where feed rate, cutting speed and depth of cut are independent and real valued parameters. Limits of these three parameters depend on the workpiece and tool material combination. Here workpiece and tool materials are Low C free machining steel and uncoated carbide. The fourth constraint is maximum allowable force on cutting tool. Typical values are taken for the last three constraints i.e. cutting force, rpm and power of machines. As these parameters are not independent, penalty method is used [2] to keep them within their ranges.

PLOTS AND TABLES

Figure 4 shows the changes of the average and best fitness over the generations. Figures 5-13 are related to the parameters at different generation.

RESULT FROM MATLAB PROGRAM

As stated in earlier the first step of GA is to create an initial population. Individual values for different parameters in the initial population are chosen within their ranges at random. Figure 5, 8 and 11 are the initial values of depth of cut, feed rate and cutting speed. It is clear from these figures that initial population is spread over the whole solution space instead of being localized because initial population is created randomly. This diversity of the population increases the region under search to find the global optima.

From the initial set of data, GA starts to converge very quickly by using some genetic operators such as reproduction, crossover, and mutation (discussed in section7.3). Figure 6, 9 and 12 describe the intermediate situation of the cutting parameters. These parameters are plotted after running the program for 20 generation. It is assumed that from the initial random positions the parameters are moving toward some specific values.

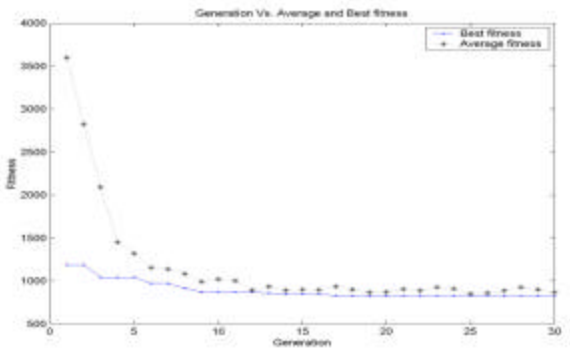

Figure 4: Generation vs. Average and best fitness

After 40 generation final values of these parameters are plotted in figure 7, 10 and 13. These three figures demonstrate that when GA reaches to or very close to optimum solution, all the values of a specific parameter in the final population becomes similar. Some exception exist i.e. some values are away from the optimum at each generation because GA checks whether it is proceeding towards a local optima or global by mutation operator.

It is also clear from figure 4 that the convergence rate at an earlier stage is much more higher than that of the later stage. Average fitness and best fitness values decrease very rapidly in the initial stages [1-20 generation]. As the number of generation increases, rate of changes in these two fitness values decrease rapidly and programs are so designed that genetic algorithm terminates when no significant improvement occurs in the solution. Usually maximum number of generation is set before the program starts. Almost all the individuals in a population become similar at the final stage [figure 7,10 and 13.

In table 1 comparisons of the parameters are presented after 40 generation and 100 generation with population size 50 and 100 respectively. It explains the advantage of high population size and maximum number of generation for the optimization problem.

Total machining time, which is the objective function in this minimization problem is improved from 131 minutes to 114 minutes i.e. almost 13% over the 60 generation. Here cutting force on the tool, which reaches near the maximum limit (2kN) at 40^{th} generation and same as the maximum limit at the 100^{th} generation is the main constraint. Due to the lower limit of cutting force, limit of the machine power becomes redundant.

In this case depth of cut and cutting speed are decreased while feed rate increased to minimize the total machining time. Number of passes is also increased as the depth of cut decreases over the generation. By genetic algorithm the most suitable machine is also selected for this specific machining operation. Here among three different machines, third one is selected because the nearest of the required rpm is available in machine 3.

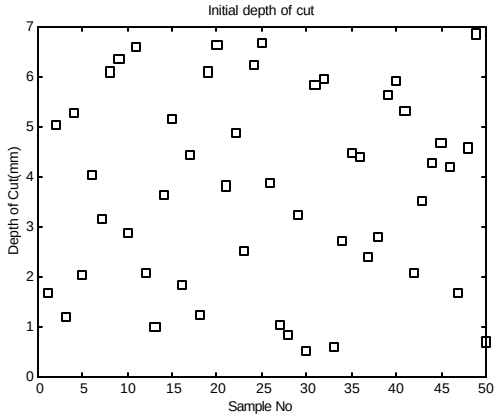

Figure 5: Initial depth of cuts

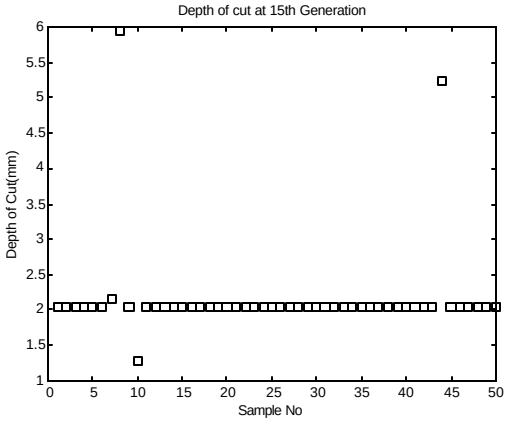

Figure 6: Depth of cuts at 20^{th} generation

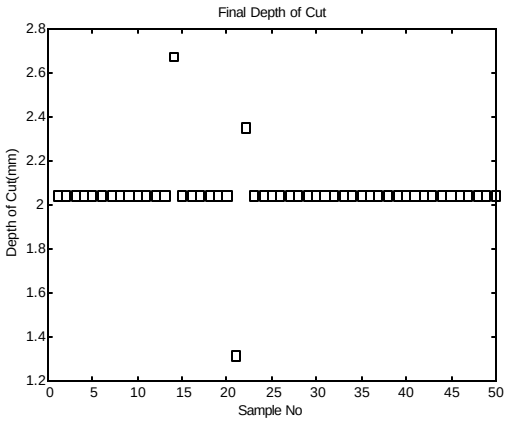

Figure 7: Final depth of cuts(40 generation)

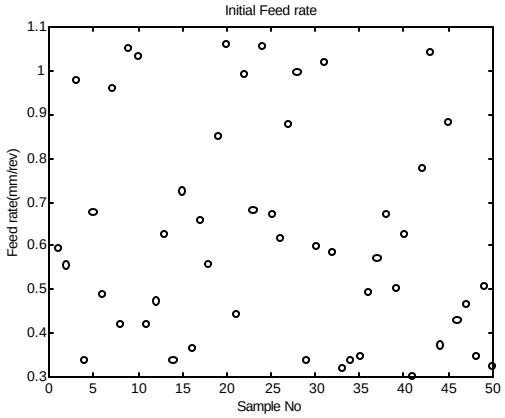

Figure 8: Initial feed rates

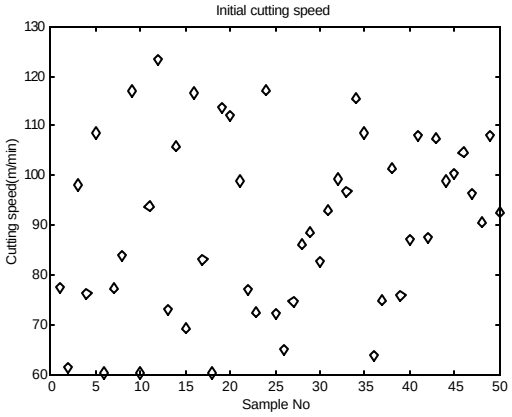

Figure 11: Initial cutting speeds

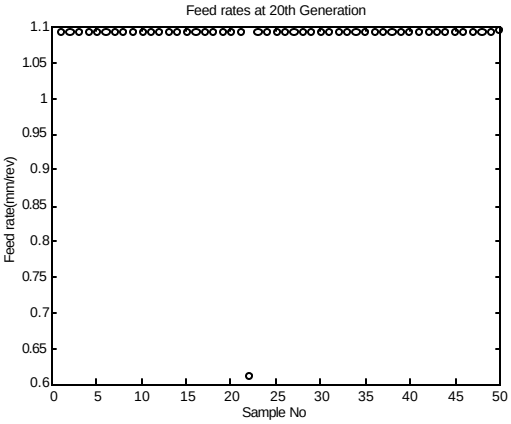

Figure 9: Feed rates at 20th generation

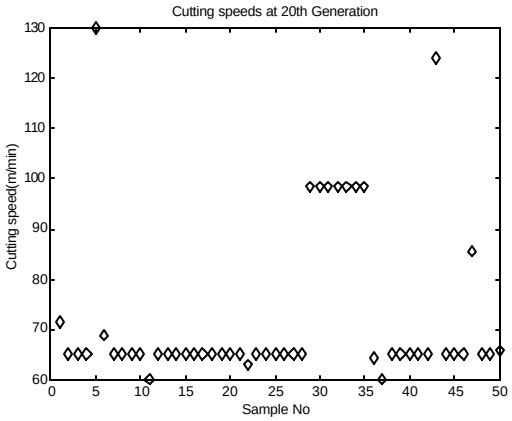

Figure 12: Cutting speeds at 20th generation)

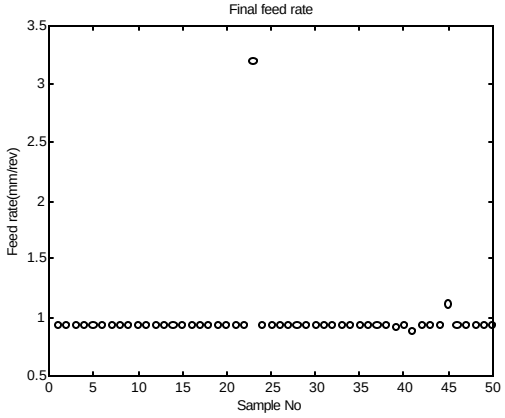

Figure 10: Final feed rates (40 generation)

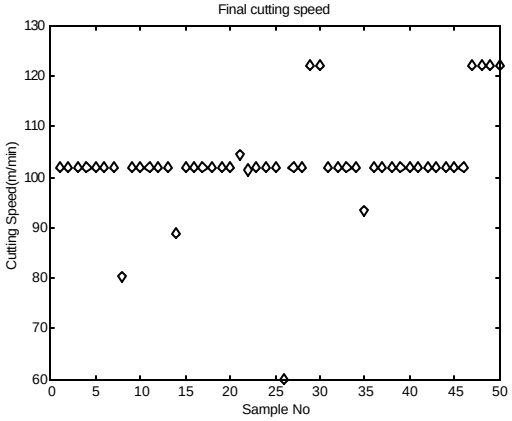

Figure 13: Final cutting speeds (40 generation)

Table 8.1: The results after 40 and 100 generation

Parameters	After 40 generation (Population size 50)	After 100 generation (Population size 100)
Depth of cut (mm)	2.0409	1.285
Feed rate (mm/rev)	0.4480	0.751
Cutting speed (m/min)	102.0994	99.353
Cutting force (kN)	1.9995	2.0000
Power (kW)	3.4024	3.7003
Rpm	162.4962	176.6797
Machine	3	3
No. of pass	48	82
Time	131.8657	114

CONCLUSION

Optimization of process parameters is one of the important task of the CAPP systems. The impact of AI techniques in CAPP had proven by many research projects. GA is promoted as one of the promising AI techniques to be used for solving nonlinear and combinatorial problems involved in process planning. With the GA-base optimization system developed in this work, it would be possible to increase machining efficiency by using optimal cutting parameters.

REFERENCES

1. Dereli, T. and Filiz, H.I., " Optimization of Process Planning functions by Genetic algorithm", Computers and Industrial Engineering, Vol. 36, pp281-308, 1999
2. Goldberg. D. E., "Genetic algorithms in search, optimization & machine learning", Addison-Wesley 1989. reading
3. Lawrence E. Doyel, "Manufacturing Processes and Materials for Engineers", 1985 Prentice-Hall International, Third edition, pp 486-500
4. Ahmad, N.,"A dynamic model of Computer Aided Process Planning for rotational components", M.Engg. thesis, 2001, BUET
5. Tulkoff, J., " Computer Aided Process Planning", Production Hand Book, 4th Edition, 1987
6. Wang H.P. and Jain K.L., "Computer Aided Process Planning", Elsevier 1991 pp-356-364