academicJournals

Vol. 8(38), pp. 1848-1857, 11 October, 2013
DOI 10.5897/SRE10.550
ISSN 1992-2248 © 2013 Academic Journals
http://www.academicjournals.org/SRE

Scientific Research and Essays

Full Length Research Paper

Development of genetic algorithm toolbox using MATLAB in cutting tool path optimization

Nurhaniza Mohamad [1], M .K. A. Ariffin[1]*, Aidy Ali[1], F. Mustapha[2] and I. M. Salleh[3]

[1]Department of Mechanical and Manufacturing Engineering, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia.
[2]Department of Aerospace Engineering, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia.
[3]Universiti Kuala Lumpur, Malaysian Institute of Aviation Technology, 43900 Dengkil, Selangor, Malaysia.

Accepted 3 June, 2013

In this study, optimization of tool path is presented through simulation material removal using Finite Element Analysis (FEA) and control via Genetic Algorithm (GA). The computer code for GA is developed and is specifically designed using MATLAB programming. The developed GA toolbox was written in m-files script together with several built-in functions in order to import the text file generated by FEA software. This paper presents the solution for cutting tool path optimization which is the interaction toolbox that consists of the integration between MATLAB and ABAQUS. This developed toolbox facilitates the optimization process to be performed successfully on the tool path machining process. The optimization technique inherently assists in producing finishing product, with minimum chatter and static deflection problem.

Key words: Genetic algorithm (GA), MATLAB programming, intelligent cutting tool path.

INTRODUCTION

Machining operations of metal needs specific set of parameters to provide an excellent surface finish. However, there is an unnecessary unavoidable error during machining like chatter. Chatter is self excited vibration that causes problems in machining operations and will always be a cause of problems related with surface finish and tool life (Bandyopadhyay and Bhatacharya, 1991). It is one of the major factors disturbing the performance of the machining process. Chatter not only affects the cutting processes by limiting the efficiency but can also reduce the productivity, yield the poor surface finish of the workpiece and decrease cutting tool life (Budak, 2003). The machining operation with chatter presented will cause the unstable cutting process. This situation is typically undesirable because the chatter affects on the machine surface (Clancy and Shin, 2002; Qu et al., 2003). Besides that, the large peak value of the variable of the cutting force, which are spindle speed, depth of cut and feed rate will cause fracture of the tools or a few parts on the machine. There are several methods to avoid or reduce chatter problem during machining process such as changing the spindle speed of the cutting tool (Soliman and Ismail, 1997), active vibration control which applied to boring bars (Klein and Nachtigal, 1975) and milling machine (Dohner et al., 2004), study the possibility of increasing the stability of the cutting process by varying the tool pitch cutter (Altintas et al., 1999) and carefully planning the tool path to ensure that the thin walled section can be machined properly (Smith and Dvorak, 1998). However, this technique required a highly skilled machinist to program the tool path. For this study, this problem would be tackled using a suitable cutting tool path strategy. Tool path plays a very important and critical role in machining

*Corresponding author E-mail: khairol@eng.upm.edu.my.

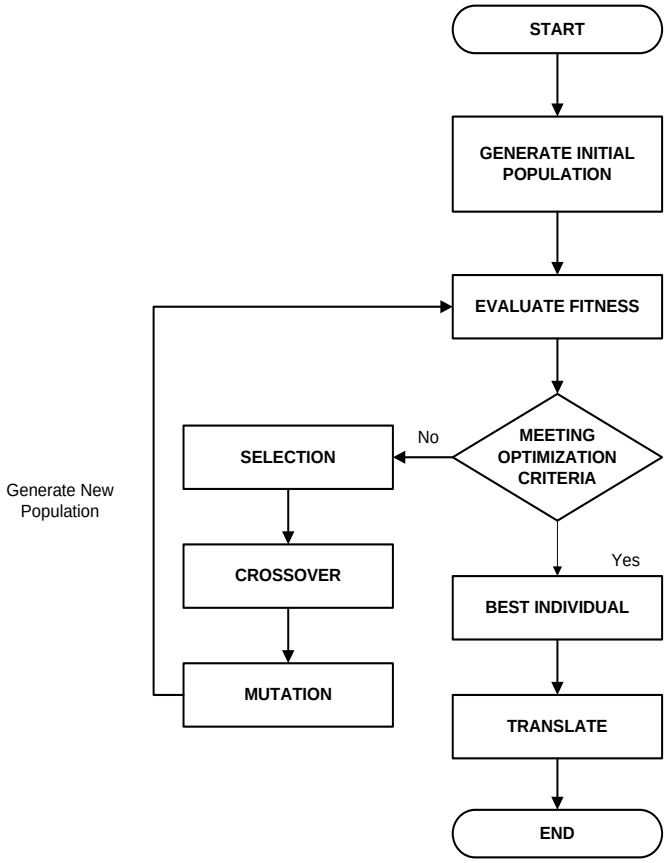


Figure 1. The flow of genetic algorithm Process (Cao and Wu, 1999).

process because it assists in reducing the workpiece vibration during machining. It also controls the metal removal during machining, surface finish and shape of the finished product (Ariffin et al., 2004). This problem is undertaken using the Finite Element Analysis (FEA) integrated with Genetic Algorithm (GA) and performed using an appropriate method or software such as MATLAB and Python (Lee and Kim, 2005).

Nowadays, there are several software introduced to facilitate users in many fields especially in engineering. This software sometime offers a concentrate practice such as for mathematical equation, FEA, Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM). For instance, MATLAB which is a high performance language for technical computing is one of the software that is usually used in engineering field (Ariffin et al., 2008). It combines computation, visualization and programming in a user friendly environment where problems and solutions are stated in well-known mathematical notation. MATLAB provides lots of toolboxes and comprised additional function for optimization to support an interactive and efficient situation for modeling and simulation (Cao and Wu, 1999; Moore, 2007).

Goldberg (1989) explained that GA plays a significant role to most researchers in many fields such as in intelligent design because of its influential capability and wide adaptation. GA is the optimization and search techniques based on the principles of natural selection and natural genetics. It can be used as an optional method to find the global optimum solution for optimization problem within a sensible time. Genetic algorithms have benefited from wide acknowledgment in a variety of domains. Few experimental studies proved that GA shows exciting effectiveness in practice and consistently outperforms both gradient techniques and different kinds of random search on lots of complex problems (Meng et al., 1999). The use of GA toolbox to optimize the cutting tool path is created using MATLAB programming and the use of GA toolbox as well as the code is present in this paper.

GENETIC ALGORITHM CONCEPTS

The concept of GA is explained in detail in many publications such as by Goldberg (1989) and Kaya (2006). It is based on the basic algorithm which started with generation of the random initial population of chromosomes. This population can be in binary, integer or permutation representation and generally performed as uniformly as possible (Eiben and Schoenauer, 2002). Then, the initial population is evaluated using an appropriate fitness function to measure the performance of the population or individuals in order to make them the better solution. After evaluation process, the population is tested whether it meets the optimization criteria or not. If the optimization criteria are satisfied, the process will stop and return the solution in current population. If the optimization criteria are not achieved, the new population is created by repeating the steps such as reproduction, selection, crossover and mutation until the new population is completed. Then, this new generated population is replaced for a further run of algorithm.

These steps are carried out continuously until the optimization or termination criteria are met. The flow of genetic algorithm concepts is shown in Figure 1.

COMPUTER IMPLEMENTATION OF THE ALGORITHM

In order to integrate GA and FEA, it is required to develop new software that can join the optimization technique. The new computer code is specifically designed using MATLAB programming software together with FEA software (ABAQUS). The new computer code consists of GA and FEA modules and is created completely using MATLAB programming. Figure 2 shows the outline of MATLAB and ABAQUS processes. The newly created computer code can be used to analyze the same problem as the previous commercial FEA software and is able to determine the best solution for a cutting tool path

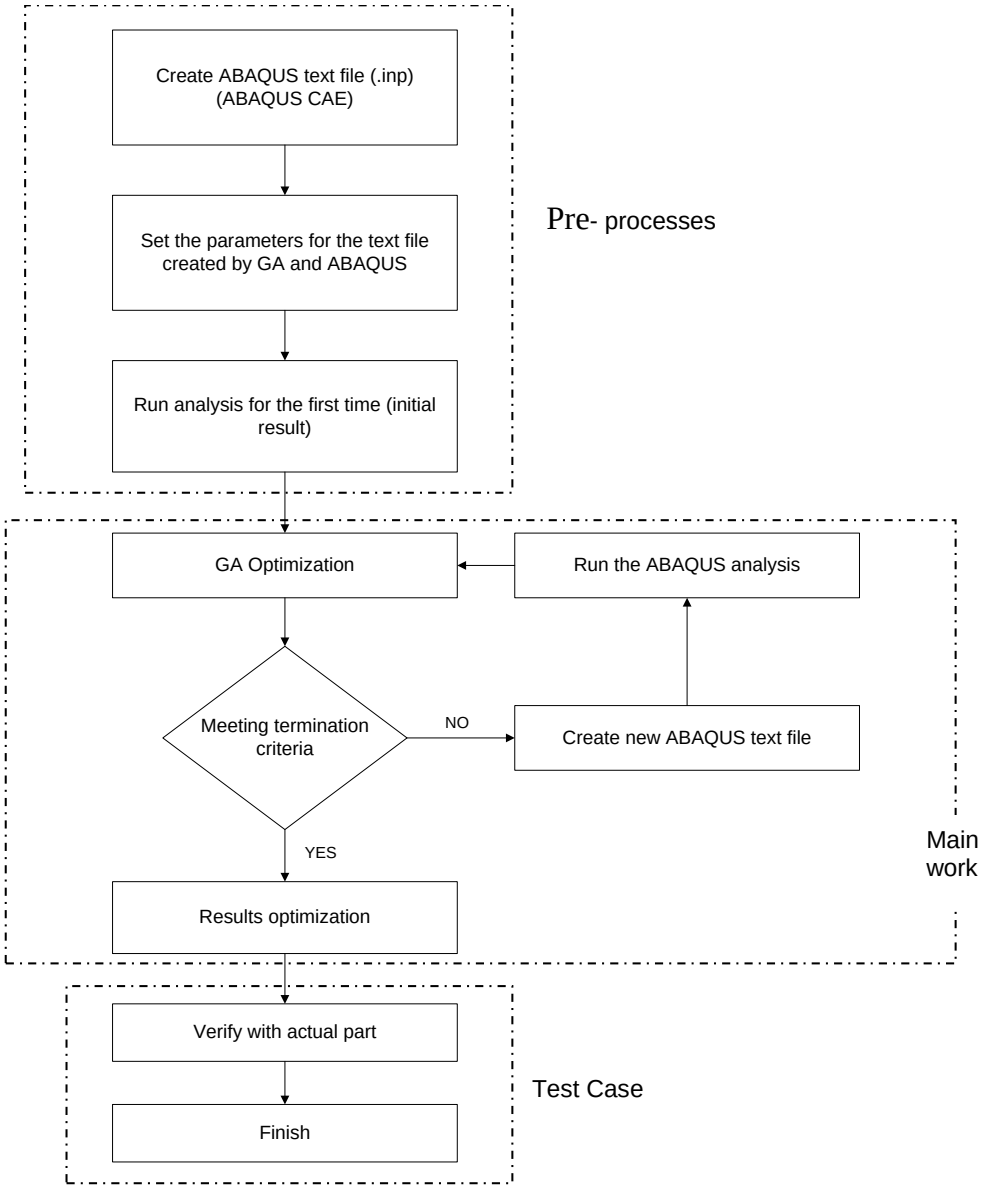


Figure 2. MATLAB and ABAQUS Processes.

strategy.

The integration between GA and ABAQUS is organized by MATLAB programming language where MATLAB is used for removal of the results of ABAQUS and tools to manipulate GA in order to find the optimum solution for the problem. The toolbox of GA is used to optimize the cutting tool path and is created using MATLAB programming which included a set of MATLAB command sequence that is called m-files and it is intended to apply the most important function in GA. MATLAB is run as a matrix processor to process numbers as an alternative of text string. This toolbox is mostly written in m-files and used several functions such as *fopen, fclose, fprintf, fscanf, findstr, num2str, str2num* and *sprintf* to import the text file and then convert it to the numerical value for optimization functions.

The integration between MATLAB and ABAQUS text file needs the user to realize the data structure and software requirements. For this research, the data structure is emphasized only on deflection and it displayed as *U* in the ABAQUS input file. The optimization process is done by MATLAB programming. From the optimization process, only the selected results are showed in the result file including all the required component of the analysis. The results obtained from the analysis are converted into MATLAB language and the results optimization process is done using GA until the aim value or best solution is achieved.

```
STEP: boltload-8
*Step, name=boltload-8
load apply on surface edge hole 8
*Static
1., 1., 1e-05, 1.
** LOADS
** Name: boltLoad-21   Type: Pressure
*Dsload
_PickedSurf120, P, 15.
** OUTPUT REQUESTS
*RESTART,WRITE,OVERLAY
*NODE PRINT, NSET=Set-11
U1,U2,U3
*NODE FILE, NSET=Set-11
U
*END STEP
```

Algorithm 1. Example of a section ABAQUS input file.

```
BEGIN
  INITIALIZE the population randomly
  EVALUATE the fitness function
  IF (not meet the optimization criteria)        DO
        SELECT parents
        CROSSOVER or RECOMBINE pairs of parents
        MUTATE the offspring obtained
        EVALUATE the new offspring
  OD
  ELSE (meet the optimization criteria)          DO
        SELECT the best individuals
        TRANSLATE the best solution
  OD
  END
```

Algorithm 2. Pseudocode of genetic algorithm.

DATA STRUCTURE OF ABAQUS RESULTS

The integration between MATLAB and ABAQUS software needs the user to realize the data structure and what the software need. For example, in ABAQUS data structure, user can choose to display the result either as deflection, stress, strain etc depending on what the user requires. For optimization process by MATLAB, only the selected result is shown in the result file. All the required component of the analysis is included in the result file. Algorithm 1 show a selected ABAQUS input file where this input file sets the deflection as the main component and is displayed as *U* in the ABAQUS input file.

```
% This function is to read the result from ABAQUS analysis

function myfile=read(filename);
fid=fopen(filename,'r');
B_C=fscanf(fid,'%c');

if (fid==-1);
   error(sprintf('problem opening the file "%s".',filename));
   return
end

read=B_C;
myfile=read;
fclose('all');
```

Algorithm 3. Example of MATLAB command for reading text file.

THE TOOLBOX STRUCTURE

The GA toolbox structure is built using MATLAB programming to implement the large variety of genetic algorithm methods. There are several general steps that consist of GA's operators such as initial population, evaluation, selection, crossover and mutation for the series of genetic algorithm. Algorithm 2 shows the general plan of genetic algorithm in pseudocode.

Initialization

Initialization or initial population is a set of individuals or possible solution that is usually formed in random. It is the first thing to do to decide the coding structure. This is the process of setting up the initial value for genetics parameters. In this paper, the problem has been coded in permutation representation which is appropriate for this kind of problem. For example, permutations of 50 sequences are represented to explain each individual in the population of interest which is referred to as chromosome. Algorithm 3 shows the command of MATLAB function for reading text file from ABAQUS.

The process selects the random population and creates the initial ABAQUS input file. Let *pop_size* and *chrom_length* as the number of individuals involved in population and the length of the permutation sequence respectively. *pop_size* refer to size of population while *chrom_length* refer to size of element. By using MATLAB programming, the entire data structure of the population is implemented by a matrix of size *(pop_size* x *chrom_length).* Algorithm 4 shows the command to generate the initial population. The population size is dependent on how many elements are required in one generation. Every function creates a set of unique number for every element possessed. The command operates following the instruction *popn = population (pop_size,chrom_length)*.

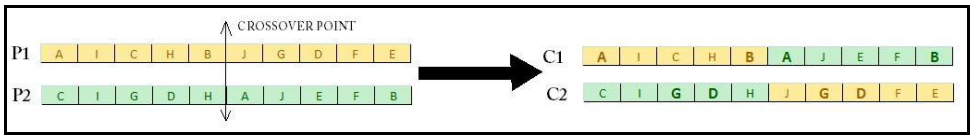


Figure 3. Crossover method for permutation.

```
popn=[];
while size(popn,1)<population_size
   temp=randperm(element_size);
   popn=[popn;temp];
   popn=unique(popn,'rows');
end
```

Algorithm 4. MATLAB command for initial population.

```
% Perform SUS
cumfit=cumsum(f_x);
trial=cumfit(Nind)/Nsel*(rand+(0:Nsel-1)');
Mf=cumfit(:,ones(1,Nsel));
Mt=trial(:,ones(1,Nind))';
[NewChrIx,ans]=find(Mt<Mf&[zeros(1,Nsel);Mf(1:Nind-1,:)]<=Mt);
```

Algorithm 5. The performance of SUS using MATLAB function (Chipperfield et al., 1993).

Evaluation or objective function

The evaluation or objective function is the most important step in genetic algorithm where it is the step to evaluate the fitness function. It is used to decide and value how individuals or chromosomes performed in the problem domain. Evaluation offers a measure of performance regarding a particular set of parameters. The evaluation process is done using ABAQUS to evaluate the fitness function. In this paper, the fitness function values are all from deflection results from ABAQUS. Each sequence generates the results where the result is used by GA for selection operation. Before starting the GA, the FEA program which uses ABAQUS evaluates the deflection for the sequences occupied by MATLAB. The lowest deflection represents the highest value. So, the calculation for the fitness function is 1/deflection where the evaluation process by GA intends to maximize the fitness function.

Selection

The selection operator determines which of the individuals stay alive and carry on for the next generation. The typical method used is Roulette Wheel Selection (RWS) which was described by Goldberg (1989). The best performance individual is selected more than once. Each member of the population is characterized by a slice that is directly proportional to its fitness. The probability of selection is shown in Equation 1.

$$p(x) = \frac{f(x)}{\sum f(x)} \quad (1)$$

Where: $p(x)$ = probability of selection
$f(x)$ = fitness value
$\Sigma f(x)$ = sum of all fitness value.

The selection operator implemented here is Stochastic Universal Sampling (SUS). SUS is another method frequently used in selection operation and it is opposite to the RWS. It is a method that gives the same portion of the selection and is equivalent to making only one spin of a wheel. The first step is to calculate the probabilities of each individual selected based on Equation 1. An outline of SUS is applied by obtaining a cumulative sum of the fitness vector, *f_x* where *f_x* is column vector that contains the fitness value of the individual in population. Then, the number of the individuals to be selected, and *Nsel,* used to generate equally spaced numbers between 0 and sum of *f_x.* The index of the individuals selected is determined by comparing the generated numbers with the cumulative sum of *f_x.* Algorithm 5 shows the MATLAB function of SUS.

Crossover

Crossover is one of the GA's operators that is, the mating process by the parents to produce a new set of chromosomes or children. Crossover involves crossing the genes from one chromosome to another chromosome and the function of chromosome is to generate new child or offspring from two parents of chromosomes by combining the information and characteristics extracted from both parents. In this paper, the problem has been coded as permutation representation. In this case, it is impossible to do a normal crossover operation because there might be two same numbers in one sequence and the operation becomes prohibited. As shown in Figure 3, the crossover operation does not work just easily crossed at specific point. The problem started when A and B

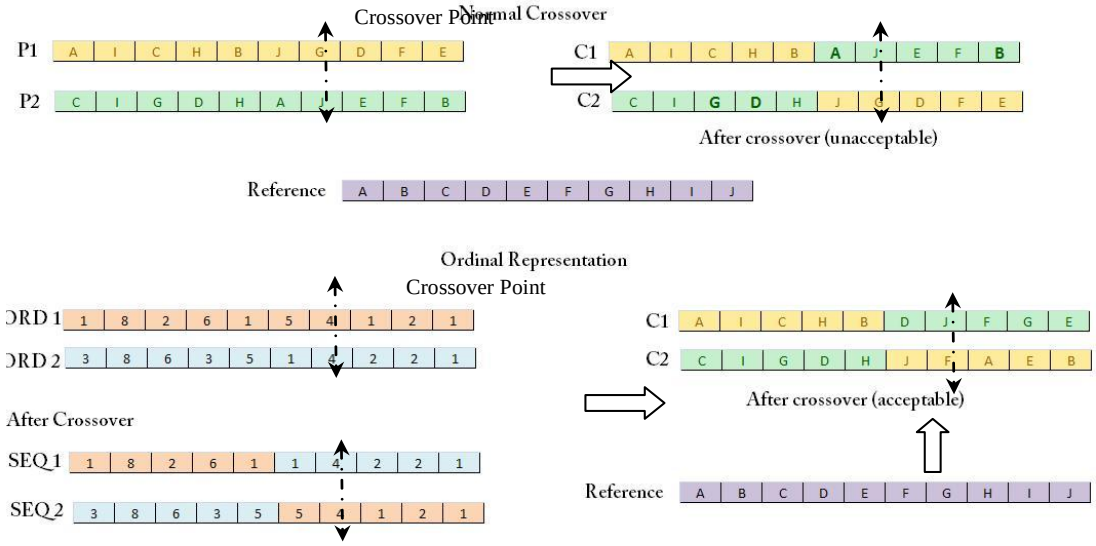


Figure 4. Example of ordinal representation for crossover purposes (Ariffin, 2006).

appear twice in child 1 and D and G appear twice in child 2. That means in machining term, this represents elements A and B for the first result and elements D and G for the second result will remove twice which is impossible. This problem can be solved by using ordinal representation which represents the solution for crossover purposes.

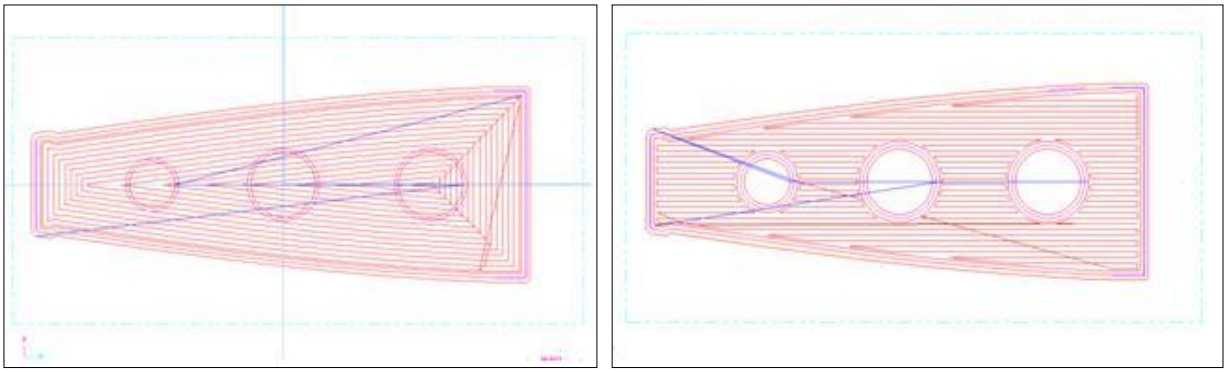
Ordinal representation

The purpose of ordinal representation is to make sure the conventional crossover operates for permutation representation. In the ordinal representation, there is a path order that act as reference sequence for each sequence in the ordinal representation. For example the reference sequence equals to (A B C D E F G H I J). According to Figure 3, if the Parent 1 and Parent 2 cross together in fifth point, then the offspring would be physically impossible. With the reference as indicated in Figure 4, the ordinal representation is described as follows: The first element in Parent 1 is *A* which matches with the first place in reference. After matching the *A* element, it is removed from the reference sequence. The next element in Parent 1 is *I*, where it is in the eighth place in reference sequence. *I* element is removed from reference sequence. The third element is *C,* which equals to the second place in reference sequence and then, *C* element is removed from reference sequence. The next element is *H,* which equals to the sixth place in reference sequence. After matching the element, *H* is removed from the reference sequence. These steps are continuous for the subsequent element in Parent 1 till the last element, *E.* Then, the same process is applied and repeated for Parent 2 until the complete ordinal representations are obtained (1 8 2 6 1 5 4 1 2 1) for ordinal 1 and (3 8 6 3 5 1 4 2 2 1) for ordinal 2. Now, the ordinal representation can cross at point number five and gives (1 8 2 6 1 1 4 2 2 1) for ordinal 1 and (3 8 6 3 5 5 4 1 2 1) for ordinal 2, which represent the removal elements of (A I C H B J G D F E) and (C I G D H A J E F B) as indicated in Figure 4. The essence of ordinal representation is to ensure that each number in each sequence only appears once after the crossover operation.

Mutation

Mutation is another GA's operator that modifies one or more genes of a selected chromosome from its initial condition where it only uses one parent to produce a single new offspring by applying some kind of randomized change to the representation. With new offspring and new gene values, the GA might be able to get a better solution than previous one. Each bit in each solution turns over with a very small probability. Normally, the probability of bit mutation ranges from 0.001 to 0.01.

The mutation operation must be done after the ordinal representation switched back to the permutation representation to ensure it creates a legal representation. One of the rules of mutation in ordinal representation is that the number left in the sequence must not be more than the place number. For instance, consider the ordinal representation of sequence A equals to (1 8 2 6 1 1 4 2 2 1). If mutation is randomly chosen and inverse between position 4 and 10, after the mutation occurs, the ordinal representation is changed to (1 8 2 1 1 1 4 2 2 6). As mentioned before, the ordinal representation must be referring to the reference sequence in order to return to



(A) (B)

Figure 5. NC Cutting (A) contour (B) zigzag tool path.

Table 1. Mechanical properties of Aluminum Alloy 7050.

Properties	Values
Modulus of Elasticity, *E*	71.7 GPa
Tensile Strength	524
Yield Strength	469
Poisson's Ratio, *u*	0.33
Density	2.83 g/cc

the permutation representation. After mutation, number 1 is positioned in the fifth place counting from the back place, and then it is acceptable. However, if the number 5 positioned on the first place from the back, it becomes illegal.

All the GA processes are terminated if the aim values are achieved or release certain practical number of generation.

CASE STUDY

By using the ABAQUS Computer Aided Engineering (CAE) software, the initial step is to model the workpiece with dimension of 600 mm × 300 mm × 25 mm. The materials used for this specimen is Aluminum Alloy 7050 with the mechanical properties as detailed in Table 1.

Then, the specimen is meshed using tetrahedral mesh or in ABAQUS code is C3D10M which represents 10 nodes for linear tetrahedron. After analysis, the meshing nodes and elements identified the critical area of the specimen. These nodes and elements are the input parameters that are used for optimization using this created toolbox. The objective of this optimization is to find the maximum stress of the loading sequence where the area of this maximum stress is considered as the critical area of the workpiece which needs to be machined first.

ABAQUS software is used to run the input file to obtain the .dat file. The input file is imported from ABAQUS CAE which contains element set and node set. The load is applied to the element set while the node set is utilized to extract fitness value. The fitness value is shown in each set where the fitness value is the maximum and minimum deflection values for each set.

The normal Numerical Control (NC) generated contour and zigzag tool paths are shown in Figure 5. Machining process is engaged with the process of selection on which pocket should be machined first without taking into account the workpiece stiffness.

RESULTS

In this paper, the requirement for genetic algorithm toolbox is solved using the specific function from MATLAB and ABAQUS itself. The specific function involved includes internal function of MATLAB and internal function of ABAQUS. Previously, this technique is applied to the model of wing aircraft holder and generates the result as detailed in Table 2. As mentioned in "Evaluation or objective function", the calculation of fitness value is equal to 1/deflection. Then, the maximum fitness value is shown in Figure 6. As a result for this study, it can be displayed in graph or table where the graph will notify the maximum value for evaluation process and if the entire steps are definitely solving the existing machining problem.

The selected area as shown in Figure 6 shows the critical analysis for this case study. This indicated the particular area where the joint is situated. For every sequences involved in this analysis, the maximum fitness value is shown in Table 2 column 2 row 11 for the first analysis and column 4 row 6 for second analysis.

Then, the simulation is extended to the new specimen with the parameters as detailed previously. The result obtained from ABAQUS is stored in .dat file. The workpiece elements meshing as shown in Figure 7 represent the material or element that would be removed

Table 2. Maximum deflection with the change of boundary condition.

Set	Maximum deflection 1st sequence	Node	Maximum deflection 2nd sequence	Node
Set 1	1.3391E-02	454	1.2318E-02	454
Set 2	2.2689E-02	456	2.0186E-02	454
Set 3	2.7633E-02	456	2.3472E-02	456
Set 4	2.9599E-02	456	2.3830E-02	456
Set 5	2.8129E-02	456	5.7427E-02	260
Set 6	3.4679E-02	260	3.8461E-02	1253
Set 7	4.5448E-02	260	2.1864E-02	456
Set 8	5.2617E-02	260	1.8618E-02	454
Set 9	5.6056E-02	260	1.1022E-02	454
Set 10	5.7427E-02	260	4.5557E-03	454

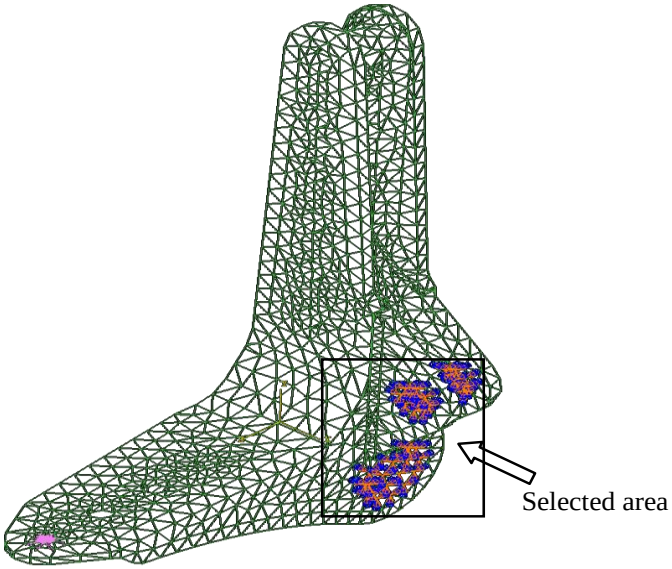


Figure 6. Maximum fitness value and node on the selected area.

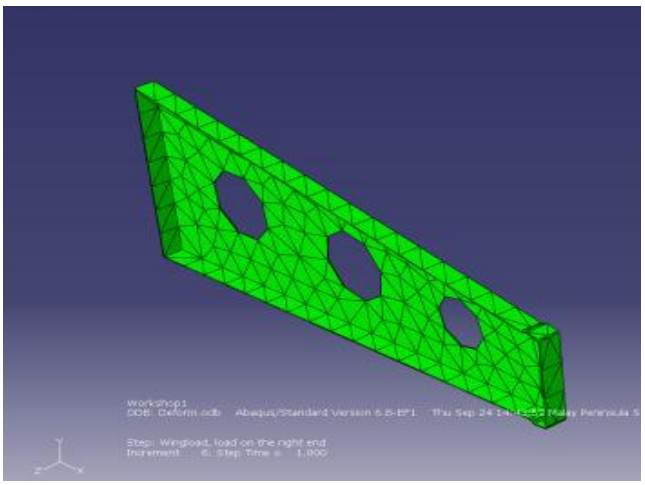


Figure 7. Workpiece element meshing.

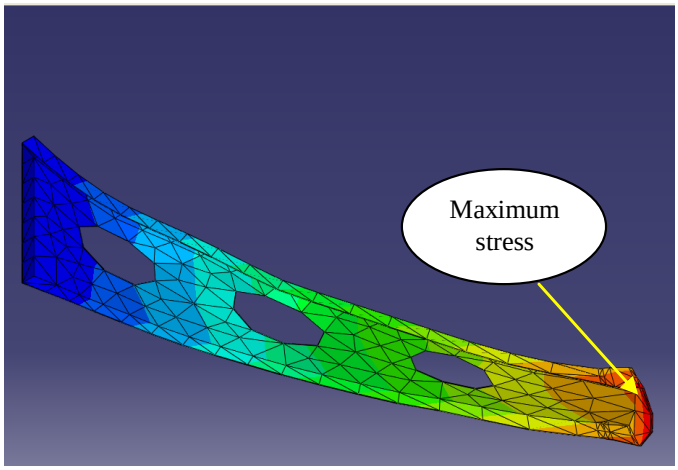


Figure 8. Stress distribution after milling process.

during simulation similar with cutting process. The material removal sequence is dependent on the sequence recommended by GA optimization which is present in this new created toolbox. Every time the element is removed, the deflection at the position of the element removal must be accounted for, as this suited the workpiece flexibility at the cutting location as indicated in Figure 8 where the figure shows the stress distribution on the workpiece after milling process. Figure 9 shows the energy plot obtained from the simulation of the workpiece where the workpiece deflection is proportional with energy.

DISCUSSION

The size of the initial population verifies the achievement of the GA where the larger initial population, the more potential solutions gain. The larger initial population tends to a thorough search, where all potential solutions are

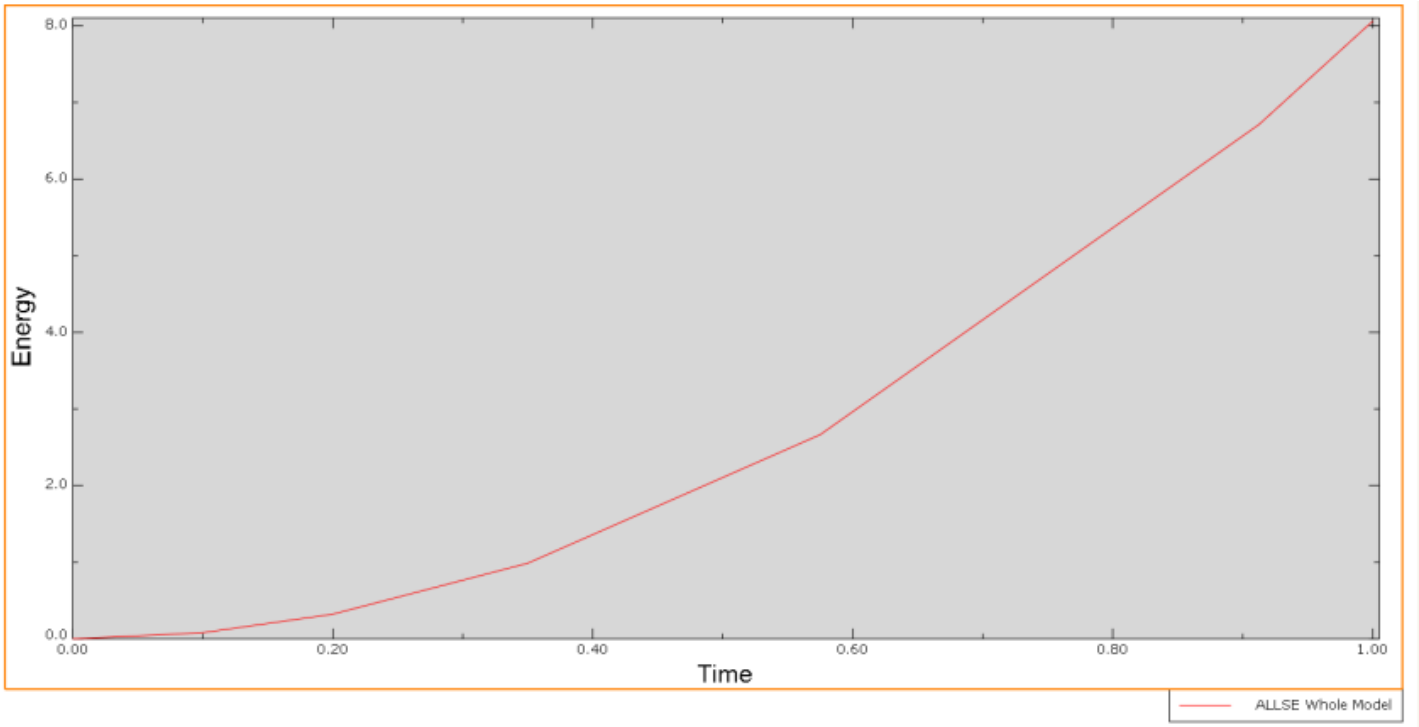


Figure 9. Energy method curve.

believed to be consequential in an important loss of time. The maximum generation number is time constraint where all the optimum fitness value can be found however the difference is only the time taken or how faster the result is found. The parameters used to obtain optimal solutions in reasonable time for different problems may differ such as changing the probability of crossover and mutation which is compatible with the particular problems.

Conclusion

Tool path optimization is one of the solutions to avoid problems occurring during machining such as workpiece chatter and static deflection. Workpiece vibration (stiffness) is identified as one of the factor that contributed to chatter and it can be sustained using an appropriate cutting tool path. This problem can be solved using GA which is controlled by MATLAB software combined with FEA which is performed by ABAQUS. In order to realize the intelligent tool path for machining workpiece, the FEA and GA are used as a tool to find the optimum value of workpiece deflection. The optimization was done by simulating the FE removal element to explain the workpiece interaction during machining.

The process of integration between ABAQUS and MATLAB software has been effectively done in order to realize the analysis of the workpiece stiffness using FE and GA to optimize the cutting tool path for avoiding chatter. It gives a new variation of CAD software especially in cutting tool path optimization. The genetic algorithm can give a better solution that is near to the best value of material removal sequence of workpiece within a reasonable time.

ACKNOWLEDGEMENT

The author would like to thank the Ministry of Higher Education Malaysia (MOHE) through University Putra Malaysia for supporting the work through Fundamental Research Grant Scheme vote number 5523411.

REFERENCES

Altintas Y, Engin S, Budak E (1999). Analytical stability prediction and design of variable pitch cutters. J. Manuf. Sci. Eng. 121(2):173-178.

Ariffin MKA (2006). Manufacturing of aerospace parts: AN intelligent tool path strategy. PhD Thesis, University of Sheffield, Sheffield.

Ariffin MKA, Faieza AA, Ismail N, Baharudin BTHT, Sulaiman S (2008). Integration of finite element analysis and MATLAB for genetic algorithm optimization. Proceedings of The International Conference on Advance Materials and Processing Technologies (AMPT), Manama, Bahrain.

Ariffin MKA, Sims ND, Worden K (2004). Genetic optimization of machine tool paths. 6th International Conference on Adaptive Computing In Design and Manufacture, Bristol: Springer-Verlag, pp. 125-136.

Bandyopadhyay BP, Bhatacharya RK (1991). Chatter reduction in machine tools. SAE Technical Paper Series, 910956.
Budak E (2003). An analytical design method for milling cutters with nonconstant pitch to increase stability, Part 1: Theory. J. Manuf. Sci. Eng. 125(1):29-34.
Cao YJ, Wu QH (1999). Teaching genetic algorithm using MATLAB. Int. J. Elec. Eng. Educ. 36(2):139-153.
Chipperfield A, Fleming PJ, Pohlheim H, Fonseca CM (1993). Genetic algorithm toolbox for use with MATLAB.
Clancy BE, Shin YC (2002). A comprehensive chatter prediction model for face turning operation including tool wear effect. Int. J. Mach. Tools Manuf. 42(9):1035-1044.
Dohner JL, Lauffer JP, Hinnerichs TD, Shankar N, Regelbrugge M, Kwan CM, Xu R, Winterbauer B, Bridger K (2004). Mitigation of chatter instabilities in milling by active structural control. J. Sound Vib. 269(1-2):197-211.
Eiben AE, Schoenauer M (2002). Evolutionary computing. Inf. Process. Lett. pp. 1-6.
Goldberg DE (1989). Genetic algorithms in search, optimization and machine learning. Reading, Mass.: Addison-Wesley Pub. Co.
Kaya N (2006). Machining fixture locating and clamping position optimization using genetic algorithms. Comp. Ind. 57(2):112-120.
Klein RG, Nachtigal CL (1975). A theoretical basis for the active control of a boring bar operation. J. Dyn. Syst. Meas. Control Trans. ASME 97(2):172-178.
Lee W, Kim HY (2005). Genetic algorithm implementation in Python. Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, Jeju Island, South Korea, pp. 8-12.
Meng QC, Feng TJ, Chen Z, Zhou CJ, Bo JH (1999). Genetic algorithms encoding study and a sufficient convergence condition of GAs. Proc. IEEE Int. Conf. Syst. Man Cybernet, Tokyo, Japan. 1:I-649–I-652.
Moore H, MATLAB for engineers (2007). Pearson Prentice Hall: Upper Saddle River.
Qu T, Khajepour A, Lin DC, Behdinan K (2003). Finite element modeling and stability analysis of chatter in end milling machining. Trans. Can. Soc. Mech. Eng. 27:205-221.
Smith S, Dvorak D (1998). Tool path strategy for high speed milling aluminum workpieces with thin webs. Mechatronics 8(4):291:300.
Soliman E, Ismail F (1997). Chatter detection by monitoring spindle drive current. Int. J. Adv. Manuf. Tech. 13(1):27-34.