COMPARISON OF PARTICLE SWARM OPTIMIZATION AND GENETIC ALGORITHM ON MATERIAL REMOVAL RATE IN THE TURNING OPERATION OF EN-41B ALLOY STEELS

T. Sreenivasa Murthy[1] , R. K. Suresh[2] , G. Krishnaiah[3]

[1] Asst. Professor - Department of Mechanical Engineering , MJR College of Engineering and Technology, Piler.

[2] Asst. Professor (Sr.) - Department of Mechanical Engineering , Srikalahasteeswara Institute of Technology, Srikalahasti.

[3] Retd. Professor , Department of Mechanical Engineering, SVU College of Engineering, S. V. University. Tirupati.

tsreem@gmail.com

Abstract

The material removal rate (MRR) is an important indicator of the efficiency and cost-effectiveness of the machining process. In this paper full factorial L_{27} Taguchi standard Orthogonal Array is chosen for the design of experiments and experiments are conducted with dry cutting condition to consider effects of Speed, Feed and Depth of cut on the MRR (response). For each of these parameters three different levels have been identified and used to perform the turning parameters for maximization of material removal rate on conventional lathe. The material selected for machining was EN41B with cermet cutting tool. Further, two different evolutionary algorithms such as, genetic algorithm and particle swarm optimization were used to predict the optimum process parameters for maximum material removal rate. Finally, the evolutionary algorithms thus used have been compared in terms of performance.

Keywords:

1. Introduction

Process parameters composed of cutting speed, feed and depth of cut (for turning operation), have essential effects on the machining productivity and cost. The selection of cutting parameters has long depended on the skills and experience of machine tool operators or handbooks, and conservative cutting parameters are usually selected. This situation would cause significant productivity loses and lead to a costly machining operation. The determination of optimum cutting parameters is a combinatorial optimization problem and is usually realized by applying optimization algorithms. These algorithms include neural network [1], geometric programming [2], simulated annealing [3], genetic algorithm (GA) [4], particle swarm optimization (PSO)[5], etc. GA was considered as a suitable algorithm for solving any type of machining process optimization problem [6]. PSO is discovered through simulation of the social behavior of bird flocking for food. It was used for optimization of continuous nonlinear functions. In PSO, the variables to be optimized need not to be encoded. Because of the convenience of realization and promising optimization ability in various problems, it has been paid more and more attention [7].In this paper, Process parameters optimizaton by using GA and PSO were discussed comprehensively.

2. Machining Optimization Model

Machining process is basically a manufacturing process for shaping of metal parts by removing unwanted material. During machining, one should satisfy efficiency and cost-effectiveness of the machining process with an objective of minimum machining time. The resulting mathematical models

for obtaining optimal combination of machining parameters including cutting speed, feed rate and depth of cut, subject to various constraints are nonlinear and non-convex in nature.

2.1 Objective function

When developing optimization models, objective functions are determined by optimization criteria. Here in present study, the maximum material removal rate is adopted as optimization criteria.

2.2 Process parameters

Speed, Feed and Depth of cut: The maximum allowable feed has a pronounced effect on both the optimum spindle speed and production rate. Feed changes have a more significant impact on tool life than depth of cut changes. The system energy requirement reduces with feed, since the optimum speed becomes lower. Therefore, the largest possible feed consistent with the allowable machine power and surface finish is desirable, in order for a machine to be fully used. It is often possible to obtain much higher metal removal rates without reducing tool life by increasing the feed and decreasing the speed. In general, the maximum feed in a roughing operation is limited by the force that the cutting tool, machine tool, work piece and fixture are able to withstand. The maximum feed in a finish operation is limited by the surface finish requirement and can often be predicted to a certain degree, based on the surface finish ad tool nose radius. Cutting speed is a vital component of tool life equation. When compared with depth of cut and feed, the cutting speed has only a secondary effect on chip breaking, when it varies in the conventional speed range. There are certain combinations of speed, feed and depth of cut which are preferred for easy chip removal which are mainly dependent on the type of tool and work piece material. Charts providing the feasible region for chip breaking as a function of feed versus depth of cut are sometimes available from the tool manufacturers for a specific insert (or) tool, and can be incorporated in the optimization systems.

2.3 Constraints

During machining, some constraints are imposed on machining processes and parameters, which effect the optimal selection of machining conditions and therefore need to be handled carefully while optimizing the machining model.

The parameters depth of cut, feed rate and cutting speed are bounded by upper and lower limits, specified by machinist or tool maker. These bounds are defined as:

(i) $Speed\ \min \le Speed \le Speed\ \max$
$360 \le Speed \le 580$
(ii) $feed\ \min \le feed \le feed\ \max$
$0.05 \le feed \le 0.09$
(iii) $doc\ \min \le doc \le doc\ \max$
$0.05 \le doc \le 0.15$

The Material Removal Rate (MRR) Measurement From the initial and final weight of job MRR is calculated and the relation is given below:

MRR = (Initial Wt - Final Wt) / Time Taken.

The dependent variable is Material removal rate.During turning, the Material removal rate, *MRR*, must not be less than the specified value of Material removal rate. The Material removal rate constraint follows the following relation

$$-1.85 + 0.713 \ln(s) + 1.61 \ln(f) + 0.331 \ln(doc) = \ln(MRR)$$

3. Methodology

In the present Evolutionary and swarm based algorithms; GA, and PSO techniques are considered for determining optimal conditions for turning process. Both the methods are discussed in detail in this section.

3.1 Genetic Algorithms

The workability of genetic algorithms (GAs) is based on Darwinian's theory of survival of the fittest. Genetic algorithms (GAs) may contain a chromosome, a gene, set of population, fitness function, breeding, mutation and selection.

GAs introduced by Holland [8] are based on the mechanics of natural selection and can search the entire search space to find the global optimum, and so have been used as a powerful tool for optimizing functions with numerous applications. They require only an `*objective function*' to guide their search. There is no requirement to formulate a mathematical equation or any *a priori* knowledge for the objective function. For complex systems, GAs have many advantages over traditional analytical approaches [9].

The data processed by GA includes set of strings (or chromosomes) with a finite length in which each bit is called an allele (or a gene). A selected number of strings is called a population and the population at a given time is a generation. Generation of the initial population of strings is done randomly. New chromosomes can be generated using genetic operators. The genetic operators operate on the genes to replace their place within the chromosome. Since the binary alphabet offers the maximum number of schemata per bit of information of any coding [10], a binary encoding scheme is traditionally used to represent the chromosomes using either zeros or ones.

Crossover, inversion and mutation are the three main genetic operators used for global searches, although there are other types of genetic operators that can yield good results. *Simple crossover* involves two parents, and crossover points are selected randomly. If two parents to be used for generating new chromosomes are {*Parent A:*1 0 0 1 0 } and {*Parent B:* 1 0 1 1 0 } and a crossover point was chosen randomly as 2; this produces the following children: {*Child A*: 1 0 | 1 1 0 } and {*Child B*: 1 0 | 0 1 1 }.

In the case of a modified crossover operator referred to as a partially matched crossover (PMX) [10] two parents are randomly picked from the population, and two crossover points are randomly chosen. These two points define where the crossover is to take place. The genes between the crossover points are replaced between the parents and the children are generated. If the same parents as above (*Parent A* and *Parent B*) are used for generating new chromosomes with the PMX operator and two crossover points were chosen randomly as 2 and 4; this produces the following children: {*Child C*: 1 1 | 1 1 | 1 } and {*Child D*: 1 0 | 0 1 | 0 }. *Inversion* operates on a single parent. It reverses the order of the element between two randomly chosen points in the parent: {*Parent A*: 1 | 0 0 | 1 1 }. Assuming that the two random inversion points are 1 and 3, the child generated by the inversion operator on the parent is: {*Child E*: 0 0 1 1 1 }.

Mutation operation involves a single parent. An index into the parent is randomly picked, and the gene at that position becomes the first gene in the new chromosome. From this picked position on, the parent is wrapped around to produce the child. This operation keeps some of the parent characteristics. If the parent is: {*Parent A*: 1 0 | 0 1 1 }, and the pick position is 2,then this operator produces: {*Child F*: 0 1 1 1 0 }.

GA is a search strategy ideally suited to parallel computing and most effectively applied to problems in which small changes result in very nonlinear behavior in the solution space [11]. GAs are able to search very large solution spaces efficiently by providing a concise computational cost, since they use probabilistic transition rules instead of deterministic ones. They are easy to implement and are increasingly used to solve inherently intractable problems called NP-hard problems.

The optimizing routines to handle NP-hard problems increase quickly with increasing problem size. Therefore, more emphasis is given on the development of heuristic procedures that usually do not claim to reach either a local or global optimum and on obtaining near optimal solutions within a reasonable computation time. It results in the restriction of the search space in some way, leaving some parts totally untouched. Although GAs are heuristic procedures themselves, they test a wealth of sampling from different regions of the search space for fitness simultaneously, and sort out and exploit regions of interest very quickly [12]. GAs are well suited to solving complicated and multi-variable optimization problems [13].Although GAs are originally developed for combinatorial optimization problems, they have been also used with success in numerical optimization problems [14]. A detailed discussion on the consideration of GAs as valid approaches to numerical optimization and the reasons can be found in Michalewicz and Janikow [15] and Schaffer *et al*.[16]. Further information on the use of GAs for engineering design and optimization can be found in Goldberg [10] and Gen and Cheng [17] .

Algorithmically, the basic genetic algorithm (GAs) is outlined as follows:

Step I [Start] Generate random population of chromosomes, that is, suitable solutions for the problem.

Step II [Fitness] Evaluate the fitness of each chromosome in the population.

Step III [New population] Create a new population by repeating following steps until the new population is complete.

a) [Selection] Select two parent chromosomes from a population according to their fitness. Better the fitness, the bigger chance to be selected to be the parent.

b) [Crossover] With a crossover probability, cross over the parents to form new offspring, that is, children. If no crossover was performed, offspring is the exact copy of parents.

c) [Mutation] With a mutation probability, mutate new offspring at each locus.

d) [Accepting] Place new offspring in the new population.

Step IV [Replace] Use new generated population for a further run of the algorithm.

Step V [Test] If the end condition is satisfied, stop, and return the best solution in current population.

Step VI [Loop] Go to step 2.

The genetic algorithms performance is largely influenced by crossover and mutation operators.

The block diagram representation of genetic algorithms (GAs) is shown in Figure.1.

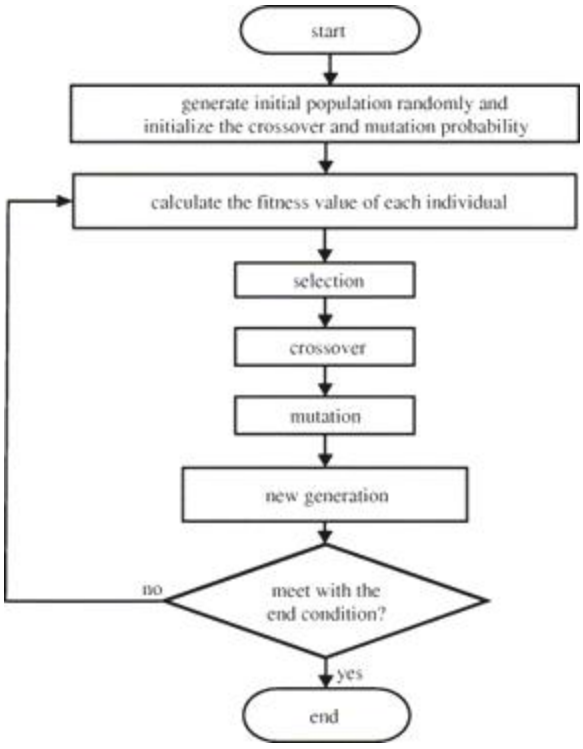

Figure 1. Block Diagram of Genetic Algorithm

3.2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization is derived from the behavior of social groups like bird flocks or fish swarms. Although the "survival of the fittest" principle is not used in PSO, it is usually considered as an evolutionary algorithm.

PSO is firstly introduced by Eberhart and Kennedy [18] and used for optimization of continuous nonlinear functions. The PSO algorithm conducts search using a population of particles which correspond to individuals in a genetic algorithm [19]. A population of particles is initially randomly generated. Each particle represents a potential solution and has a position represented by a position vector. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector. At each time step, a function representing a quality measure is calculated by using as input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far in a vector. Furthermore, the best position among all the particles obtained so far in the population is kept track as output. In addition to this global version, another local version of PSO keeps track of the best position among all the topological neighbors of a particle. At each time step, by using the individual best position, and global best position, a new velocity for particle is updated.

The procedure of PSO is illustrated as follows.

i. **Initialization.** Randomly generate a population of the potential solutions, called "particles," and each particle is assigned a randomized velocity.

ii. **Velocity Update.** The particles then "fly" through search hyperspace while updating their own velocity, which is accomplished by considering its own past flight and those of its companions.

The particle's velocity and position are dynamically updated by the following equations:

$$v_{id}^{NEW} = w_i \cdot v_{id}^{OLD} + C_1 \cdot r_1 \cdot \left(x_{pd} - x_{id}^{OLD}\right) + C_2 \cdot r_2 \cdot \left(x_{gd} - x_{id}^{OLD}\right) \quad \text{-- (1)}$$

$$x_{id}^{NEW} = x_{id}^{OLD} + v_{id}^{NEW} \quad \text{----- (2)}$$

where the acceleration coefficients C_1 and C_2 are two positive constants; w_i is an inertia weight and r_1, r_2 is a uniformly generated random number from the range [0, 1] which is generated every time for each iteration. Eberhart and Shi [20] and Hu and Eberhart [21] suggested using $C_1 = C_2 = 2$. Equation (1) shows that, when calculating the new velocity for a particle, the previous velocity of the particle (v_{id}), their own best location that the particles have discovered previously (x_{id}) and the global best location (x_{gd}) all contribute some influence on the outcome of velocity update. The global best location (x_{gd}) is identified, based on its fitness, as the best particle among the population. All particles are then accelerated towards the global best particle as well as in the directions of their own best solutions that have been visited previously. While approaching the current best particle from different directions in the search space, all particles may encounter by chance even better particles en route, and the global best solution will eventually emerge. Equation (2) shows how each particle's position (x_{id}) is updated in the search of solution space.

The block diagram representation of Particle Swarm Optimization algorithm (PSO) is shown in Figure.2.

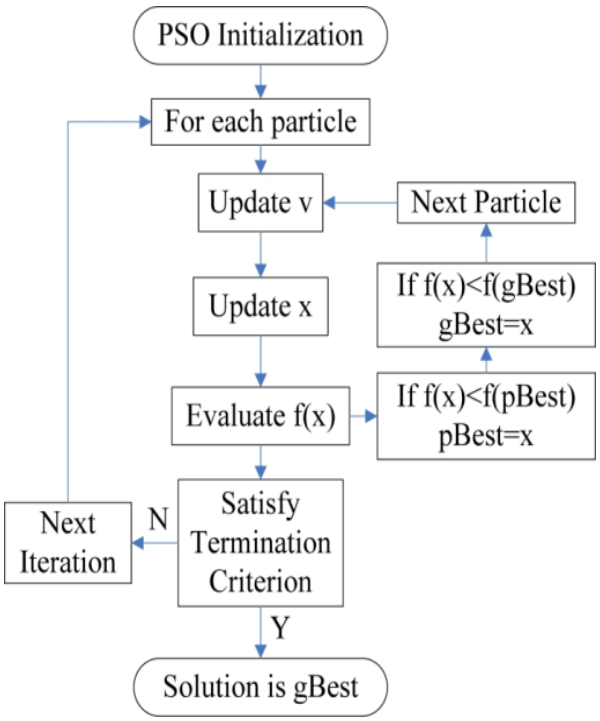

Figure 2. Block Diagram of Particle Swarm Optimization Algorithm

4. Results and Discussions

The aim was to optimize the speed, feed and depth of cut for obtaining maximum material removal rate for turning process. MATALAB GA Toolbox is used for implementation of Genetic Algorithm and for particle swarm optimization; program is developed and implemented for single objective turning process using MATLAB software. The GA and PSO was implemented using same set of variables, constraints and boundary conditions. The program is run with different combinations of generations between 5 and 50. The GA and PSO programs are run on Intel Pentium Core I5 Processor with 2 GB RAM. Table 1 shows the results of GA and Table 2 shows the results of PSO implementation on the designed problem (objective function). Using PSO implementation, the maximum MRR is 0.172 compiled with 43rd generation at run no. 6. The maximum MRR using GA is 0.182 compiled with 43rd generation at run no.6.

Table 1: GA Results (Turning)

Run No.	No. of Generations	Speed	Feed	Depth of cut	MRR
1	6	570	0.058	0.135	0.178
2	29	512	0.078	0.14	0.162
3	20	547	0.055	0.14	0.145
4	50	478	0.086	0.14	0.134
5	35	480	0.087	0.14	0.171
6	43	473	0.071	0.14	0.182

Table 2: PSO Results (Turning)

Run No.	No. of Generations	Speed	Feed	Depth of cut	MRR
1	6	457	0.071	0.09	0.124
2	29	552	0.07	0.11	0.138
3	20	522	0.09	0.14	0.159
4	50	512	0.07	0.15	0.149
5	35	442	0.07	0.10	0.131
6	43	555	0.09	0.14	0.172

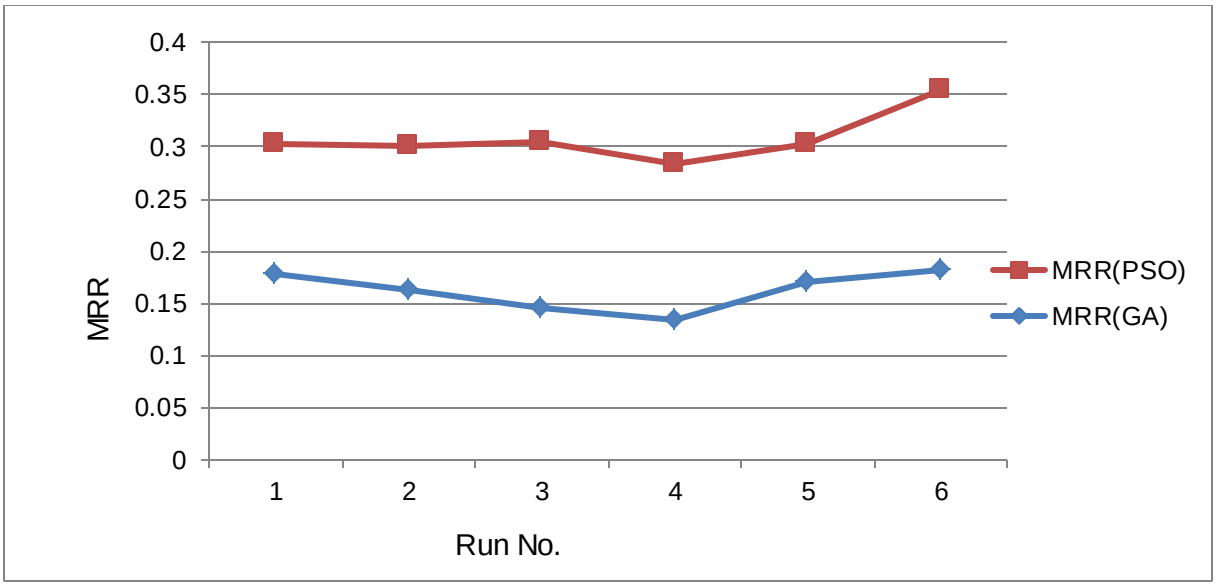

Figure 3. GA and PSO Material removal Rate Comparison.

Figure 3. shows the comparison of the objective function, a function of maximization of material removal rate, found after Genetic Algorithm and Particle Swarm optimization implementation (Ref. Table 1 and Table 2).

5. Conclusion

In this paper, the cutting parameters optimization by using non-traditional algorithms; genetic algorithm (GA) and particle swarm optimization (PSO) have been employed to find the optimal machining parameters during the machining operation of EN 41B steel with cermet tool. According to the results, conclusions were drawn as follows:

1. The optimized parameters by GA are cutting speed 473, feed 0.07, and depth of cut 0.14 which produce MRR around 0.182 .
2. The optimized parameters by PSO are cutting speed 555, feed 0.07, and depth of cut 0.14 which produce MRR around 0.172.
3. PSO in cutting parameters optimization can converge quickly to a consistent combination of spindle speed, feed and depth of cut.
4. The genetic algorithm (GA) results are found better in terms of the objective function as compared with particle swarm optimization (PSO) results.

References

[1] U. Zuperl, F. Cus, B. Mursec and T. Ploj: J. of Mater. Process, Tech. Vol.157-158 (2004), pp.82.
[2] B. Gopalkrishan and A. K. Faiz: Int. J. of Prod. Res, Vol.29 (1991), pp.1897.
[3] Kalyanmoy Deb: Optimization for Engineering Design: Algorithms and examples (Prentice Hall, USA 1995).
[4] D. E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning (Addison Wesley-Longman, UK 1989).
[5] M.P. Song and G.C. Gu: Proceedings of the Third International Conference on Machine Learning and Cybernetics (Shanghai, China 2004), pp.2236.
[6] R. Saravanan, P. Asokan and M. Sachithanandam: Int. J. Adv. Manuf. Tech, Vol.17 (2001), pp.471.
[7] A. Noorul Haq, K. Sivakumar, R. Saravanan and K. Karthikeyan: Int. J. Adv. Manuf. Technology, Vol. 27 (9-10) (2006), pp. 865.
[8] Holland, J. H.,. Adaptation in Natural and Artificial Systems (1975) (New York: The University of Michigan Press).
[9] Hashimoto, Y., Applications of artificial neural networks and genetic algorithms to agricultural systems. Computers and Electronics in Agriculture, 18(2) (1997), 71- 72.
[10] Goldberg, D., Genetic Algorithms in Search, Optimization and Machine Learning (1989) (Reading: Addison Wesley).
[11] Kamhawi, H. N., Leclair, R. S. and Philip, C. L., Feature sequencing in the rapid design systemusing a genetic algorithm. Journal of Intelligent Manufacturing, 7 (1996), 55- 67.

[12] Ulusoy, G., Serifoglu, S. F. and Bilge, U., A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Proceedings of the 1st International Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey, (1996), pp. 438- 461.

[13] Chen, C. J. and Tseng, C. S., The path and location planning of workpieces by genetic algorithms. Journal of Intelligent Manufacturing, 7(1996), 69-76.

[14] Karaboga, D., Design of fuzzy logic controllers using tabu search algorithm. Biennial Conference of the North American Fuzzy Information Processing Society (1996), University of California, Berkeley, USA.

[15] Michalewicz, Z. and Janikow, C. Z., Genetic algorithms for numerical optimization. Statistics Computing, 1(1991), 75- 91.

[16] Schaffer, J. D., Eshelman, L. J. and Offutt, D., 1991, In G. Rawlins (ed.), Spurious Correlations and Premature Convergence in Genetic Algorithms (San Mateo, CA: Morgan Kaufmann (1991)), pp. 102-112.

[17] Gen, M. and Cheng, R., Genetic Algorithms and Engineering Design (1997), (New York: Wiley).

[18] Eberchart R and Kennedy J . A new optimizer using particle swarm theory, Proceedings of the International Symposium on Micro Machine and Human Science(1995), pp. 39–43

[19] Jouni Lampinen and Rainer Storn. Differential Evolution In Godfrey C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 123–166. Springer-Verlag, Berlin Heidelberg, (2004).

[20] Eberhart, R. C. and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. *Proceedings of the Congress on Evolutionary Computation*. (2001), Seoul, Korea, 94-97.

[21] Hu, X. and R. C. Eberhart. Tracking dynamic systems with PSO: where's the cheese?" *Proceedings of The Workshop on Particle Swarm Optimization*. (2001), Indianapolis, IN, USA.