Int J Adv Manuf Technol (2008) 38:884–895
DOI 10.1007/s00170-007-1144-3

ORIGINAL ARTICLE

Real-coded genetic algorithm for machining condition optimization

Sung Soo Kim · Il-Hwan Kim · V. Mani · Hyung Jun Kim

Received: 15 December 2006 / Accepted: 25 June 2007 / Published online: 6 September 2007

Abstract In this paper, we consider the machining condition optimization models presented in earlier studies. Finding the optimal combination of machining conditions within the constraints is a difficult task. Hence, in earlier studies standard optimization methods are used. The non-linear nature of the objective function, and the constraints that need to be satisfied makes it difficult to use the standard optimization methods for the solution. In this paper, we present a real coded genetic algorithm (RCGA), to find the optimal combination of machining conditions. We present various issues related to real coded genetic algorithm such as solution representation, crossover operators, and repair algorithm in detail. We also present the results obtained for these models using real coded genetic algorithm and discuss the advantages of using real coded genetic algorithm for these problems. From the results obtained, we conclude that real coded genetic algorithm is reliable and accurate for solving the machining condition optimization models.

S. S. Kim · H. J. Kim
Department of Industrial Engineering,
Kangwon National University,
Chunchon 200-701, South Korea

S. S. Kim
e-mail: kimss@kangwon.ac.kr

I.-H. Kim
Department of Electronic and Tele Communication Engineering,
Kangwon National University,
Chunchon 200-701, South Korea
e-mail: ihkim@kangwon.ac.kr

V. Mani (✉)
Department of Aerospace Engineering, Indian Institute of Science,
Bangalore 560 012, India
e-mail: mani@aero.iisc.ernet.in

Keywords Machining conditions · Multi-pass turning · Optimization · Real-coded genetic algorithm · Single-pass turning

1 Introduction

Process planning is an important issue in the modern competitive manufacturing environment. A good process plan will reduce the cost of manufacture and thus increase the profits of the organization. With the availability of modern computers and information technology, the process planning activity can be computerized. This is known as computer aided process planning (CAPP) in the area of manufacturing.

In this study, we consider the problem of obtaining optimal combination of machining conditions (parameters) that forms an important part in computer aided process planning. This problem is a non-linear constrained optimization problem. The objective function in this problem is that of minimizing the cost of production. In general, the constraints are the physical machining parameters, such as the cutting speed, feed rates, depth of cut, cutting force, cutting power, tool life, temperature, surface finish, surface roughness and horse power. The constraints depend on the type of machining operation used in the manufacturing process. In earlier studies [1–10], different machining models are presented as non-linear constrained optimization problems. In these studies standard constrained optimization methods are used to obtain the solution. We will discuss these models in detail later in this paper.

In this paper, we present a real coded genetic algorithm (RCGA) to find the optimal combination of machining conditions. In our study, we consider five machining condition optimization models presented in [1]. Finding the optimal combination of machining conditions within the constraints is a difficult task. Hence, in earlier studies standard optimization methods are used. The non-linear nature of the objective function, and the constraints that need to be satisfied makes it difficult to use the optimization methods for the solution.

Genetic algorithms (GA) is a well-known search algorithm that mimic the evolution in nature. In the initial studies on genetic algorithms [11–14], the decision variables (or solutions) are represented as binary strings. The disadvantage with this binary string representation for real valued optimization problems is that the length of the chromosome increases with the number of decision variables and in turn affects the efficiency and convergence of the genetic algorithm. In genetic algorithm it is possible to use the real value representation for decision variables instead of binary string representation. Genetic algorithms using real value representation for solutions are called real coded genetic algorithms (RCGA). This type of solution representation is also known as floating-point representation, real number representation, or continuous representation. It is shown in [14] that for numerical optimization problems, floating point representation of solutions performs better than binary representation because they are more precise, more consistent, and lead to faster convergence. There is an increasing interest in solving real-world optimization problems using real coded genetic algorithms [15–22]. The physical constraints in the real-world optimization problem, restricts the search space and the feasible regions may be disjointed. Real coded genetic algorithms are population based search algorithms and are the same as binary coded genetic algorithms except for the fact that the solution is represented as real numbers. The algorithm starts with a population of solutions and these solutions are improved in each generation by means of selection, crossover (recombination), and mutation (if necessary) as in binary coded genetic algorithms. It is shown in [16] that the binary coded genetic algorithm does not offer the same reliability and accuracy as the real coded genetic algorithm. It is also shown that the binary coded genetic algorithm requires higher computation time than the real coded genetic algorithm. Various issues related to solving constrained optimization problems using real coded genetic algorithms are presented in [18–20]. In real coded genetic algorithms crossover operator (recombination operator) is regarded as the main search operator. The crossover operator directs the search toward the neighborhood of the parents. There are many type of crossovers operators presented in earlier studies. An extensive study of different crossover operators is presented in [21]. A matlab based genetic algorithm for function optimization is discussed in [22].

Contributions of this paper In this paper, we consider the machining condition optimization models presented in earlier studies. We obtain the optimal machining parameters using a real coded genetic algorithm. We present various issues related to real coded genetic algorithm such as solution representation, crossover operators, repair algorithm in detail. We also present the results obtained for these models using real coded genetic algorithm and discuss the advantages of using real coded genetic algorithm to these problems.

2 Mathematical model

In this section, we first describe the mathematical model presented in earlier studies.

Model.1 This model is used in multi-pass turning operation of mild steel work-piece using a carbide tool presented in [4] and considered in [1] and [3]. The objective in this model is to minimize the production cost in dollars/piece. The objective function is:

$$Min.\ Cost = n * \left\{ \begin{array}{l} 3141.59V^{-1}f^{-1}d^{-1} + 2.879 \\ \times 10^{-8}V^{4}f^{0.75}d^{-0.025} + 10 \end{array} \right\} \tag{1}$$

In the above equation, n is the number of passes and d is the depth of cut. The allowable range for d is $1.20 \le d \le 2.75\ mm$. The results for $n=2$ and $d=2.5$ are given in [1] and [3]. For comparing the results obtained from real coded genetic algorithm with earlier studies, we also use $n=2$ and $d=2.5$. Hence, the two decision variables are V the cutting speed and f the feed rates. The allowable range for V and f are given as:

$$50 \le V \le 400\, m/min \tag{2}$$

$$0.30 \le f \le 0.75\, mm/rev \tag{3}$$

There are four physical constraints in this model. These constraints are on cutting force (F_c), cutting power (P_c), tool life (TL), and temperature (T). These constraints are $F_c \le 85\ kg$, $P_c \le 2.25\ kW$, $25 \le TL \le 45\ min$ and $T \le 1000°C$. These constraints are functions of V and f and the functional relationships are:

$$F_c = \left(28.10V^{0.07} - 0.525V^{0.5}\right)d \times f\left\{1.59 + 0.946\frac{1+x}{\left\{(1-x)^2 + x\right\}^{0.5}}\right\} \tag{4}$$

$$P_c = \frac{0.746F_cV}{4500} \tag{5}$$

$$TL = 60\left\{\frac{10^{10}}{V^5 f^{1.75} d^{0.75}}\right\} \quad (6)$$

$$T = 132 V^{0.4} f^{0.2} d^{0.105} \quad (7)$$

where $x = \left\{\frac{V}{142} exp(2.21f)\right\}^2$.

Model.2 This model is used for the single pass turning operation presented in [5] and considered in [1] and [3]. The objective in this model is to minimize the production cost in dollars/piece. The objective function is:

$$Min.\ Cost = 1.25V^{-1}f^{-1} + 1.8 \times 10^{-8} V^3 f^{0.16} + 0.2 \quad (8)$$

There are three physical constraints in this model. These constraints are on surface finish (*SF*), feed rate (*f*), and the horse power (*HP*). These constraints are *SF*≤100 *μin*, *f*≤ 0.01 *in/rev*, and *HP*≤2*hp*. These constraints are functions of cutting speed (*V*) and feed rate (*f*) and the functional relationships are:

$$SF = 1.36 \times 10^8 V^{-1.52} f^{1.004} \quad (9)$$

$$f \leq 0.01 \quad (10)$$

$$HP = 3.58 V^{0.91} f^{0.78} \quad (11)$$

Model.3 This model is used for single pass turning of a medium carbon steel workpiece using a carbide tool presented in [6] and considered in [1] and [3]. The objective in this model is to minimize the production cost in dollars/piece. The objective function is:

$$Min.\ Cost = 452V^{-1}f^{-1} + 10^{-5}V^{2.33}f^{0.4} \quad (12)$$

There are two physical constraints in this model. These constraints are on cutting power (P_c) and surface finish (R_a). These constraints are P_c≤5.5, and R_a≤2 *μm*. These constraints are functions of cutting speed (*V*) and feed rate (*f*) and the functional relationships are:

$$P_c = 10.6 \times 10^{-2} V f^{0.83} \quad (13)$$

$$R_a = 2.2 \times 10^4 V^{-1.52} f \quad (14)$$

Model.4 This model is used for single pass turning presented in [7] and considered in [1] and [3]. The objective in this model is to minimize the production cost in dollars/piece. The objective function is:

$$Min.\ Cost = 1.2566V^{-1}f^{-1} + 1.77 \times 10^{-8} V^3 f^{0.16} + 0.2 \quad (15)$$

There are three physical constraints in this model. These constraints are on feed rate (*f*), surface finish (*SF*), and horse power (*HP*). These constraints are *f*≤0.1 *in/rev*, *HP*≤ 4 *hp*, and *SF*≤50 *μin*. These constraints are functions of cutting speed (*V*) and feed rate (*f*) and the functional relationships are:

$$HP = 2.39 V^{0.91} f^{0.78} d^{0.75} \quad (16)$$

$$SF = 204.62 \times 10^6 V^{-1.52} f^{1.004} d^{0.25} \quad (17)$$

In the earlier study [1], the value of depth of cut (*d*) is used as 0.2 *in*. In our study also, we use the same value as it is convenient for comparison of results.

Model.5 This model is used in multi-pass turning operation of a medium carbon tool presented in [8] and considered in [1] and [3]. The objective in this model is to minimize the production cost in yens/piece. The objective function is:

$$Min.\ Cost = \sum_{i=1}^{n} \left\{ \begin{array}{l} 3927V_i^{-1}f_i^{-1} + 1.95 \\ \times 10^{-8} V_i^{2.88} f_i^{-1} exp(5.884 f_i) d_i^{-1.117} \\ +60 \end{array} \right\} \quad (18)$$

In the above equation, n is the number of passes and d_i is the depth of cut. The sum of depths of cut of the n passes used to remove the total depth A of material and so $\sum_{i=1}^{n} d_i = A$. The allowable ranges for feed rates (*f*), cutting speeds (*V*), and depth of cut *d* are given as:

$$0.001 \leq f \leq 5.6\, mm/rev \quad (19)$$

$$14.13 \leq V \leq 1005.3\, m/min \quad (20)$$

$$0 \leq d \leq A\, mm \quad (21)$$

There are four physical constraints in this model. These constraints are on cutting force (F_c), stable cutting regions related to cutting surface, surface roughness (H_{max}), and power consumption (P_c). These constraints are F_c≤170 *kg*, fV^2≥2230.5, 0.356 f^2≤H_{max} (H_{max} ranges from 0.01 to

0.06 *mm*), and P_c=7.5 *kW*. These constraints are functions of V and f and the functional relationships are:

$$F_c = 290.73V^{-0.1013}f^{0.725}d \tag{22}$$

$$P_c = \frac{F_cV}{4896} \tag{23}$$

In the earlier study [1], this model is solved for removing a depth of 2 mm and a surface roughness of H_{max}= 0.006 *mm*. In our study also, we use the same values as it is convenient for comparison of results.

3 Real coded genetic algorithm

Genetic algorithms differ from conventional optimization methods in the following ways:

1. Genetic algorithms work with a coding of parameter set and not the parameters themselves.
2. Genetic algorithms search from a population of solution points instead of a single solution point.
3. Genetic algorithms uses fitness function information and does not use derivatives or auxiliary knowledge in their search for optimal/best solution.
4. Genetic algorithms use probabilistic transition rules, i.e., randomized operators and not deterministic rules for information exchange among the strings.

The construction of a real-coded genetic algorithm for the machining condition optimization problems described in the earlier section involves the following issues: solution representation, population initialization, selection function, design of genetic operators, fitness function and termination criterion. Now, we will describe these issues involved in applying real-coded genetic algorithm to our machining condition optimization problems (Model.1–Model.5).

Solution representation The most common solution representation in genetic algorithms is binary representation. In fact, for the machining condition optimization problems, earlier studies [1, 2] used a binary representation for the decision variables. It is shown in an earlier study [16] that a natural representation of solutions is more efficient and produces better results. The advantages of real coded genetic algorithm (or real representation) are:

– The binary coding of decision variables discretizes the solution space as a set of decision points, but the decision variables are continuous in the solution space. This preassigned set of decision points may result in local optimum.
– The length of the chromosome in the binary representation is another major disadvantage. The length of the chromosome increases exponentially with the required accuracy. This will affect the efficiency of the genetic algorithm. In the earlier studies [1, 2], the decision variables are encoded as 32 bit and 12 bit binary numbers, respectively.
– The mapping between solution space and binary space creates problems for the crossover operator used in genetic algorithms. In the discrete variable case, the genetic operators may produce invalid offsprings.

In this paper, the solution is represented as a string of real numbers. The number elements in the string is the number of decision variables. The number of decision variables for Models.1–4 are two namely cutting speed (V) and feed rate (f). For Model.5 the number of decision variables are four and they are cutting speed (V), feed rate (f), and depth of cut d_1, d_2 in two passes. A valid string (solution representation) in our problems should satisfy the corresponding constraints for the models. This should be kept in mind while generating the initial population of solutions and design of genetic operators.

Initial population As mentioned earlier, genetic algorithms searches many solutions in the search space in parallel. This is due to the fact that the genetic algorithms search from a population of solution points rather than a single solution point. The method of generating initial population affects the convergence of the problem. The size of initial population is problem dependent. One way of generating initial population is to generate randomly in the intervals for the decision variables. Because of the constraints, we may generate invalid solutions. We can design a repair algorithm to make the invalid solutions to valid solutions. Another way is to preprocess the constraints and obtain the initial population. By preprocessing we mean obtaining the initial population from feasible regions. We will explain the population initialization for our models in the Appendix in detail.

Selection function In genetic algorithms, selection of solutions from the existing population (solutions) to produce new solutions for the next generation plays an important role. In literature [11–14] there are several selection schemes such as roulette wheel selection and its extensions, scaling techniques, tournament, elitist models, and ranking methods are presented. In our study we have used roulette wheel selection method.

Genetic operators Genetic operators such as crossover and mutation provide the basic search mechanism in binary coded genetic algorithms. In real coded genetic algorithms crossover operator (recombination operator) is regarded as the main search operator. The crossover operator directs the search toward the neighborhood of the parents. The crossover operation is a method of sharing information between two solutions to the problem. Many researchers have focussed their attention in designing effective crossover operators for real coded genetic algorithms [21]. We will discuss below some of the crossover operators used in real coded genetic algorithms.

Two point crossover (TPX) Let C_1 and C_2 are the two solutions selected for crossover operations. The solutions are of length M, when the number of decision variables are M. This operator first select two crossover points randomly i, j and $i<j$.

$$C_1 = \left\{c_1^1, c_2^1, \cdots, c_i^1, c_{i+1}^1, \cdots, c_j^1, c_{j+1}^1, \cdots, c_M^1\right\} \tag{24}$$

$$C_2 = \left\{c_1^2, c_2^2, \cdots, c_i^2, c_{i+1}^2, \cdots, c_j^2, c_{j+1}^2, \cdots, c_M^2\right\} \tag{25}$$

Two new solutions H_1 and H_2 are obtained as

$$H_1 = \left\{c_1^1, c_2^1, \cdots, c_i^2, c_{i+1}^2, \cdots, c_j^2, c_{j+1}^1, \cdots, c_M^1\right\} \tag{26}$$

$$H_2 = \left\{c_1^2, c_2^2, \cdots, c_i^1, c_{i+1}^1, \cdots, c_j^1, c_{j+1}^2, \cdots, c_M^2\right\} \tag{27}$$

Simple crossover (SCX) Here only one crossover point i is selected randomly and the second crossover point is M. Two new solutions H_1 and H_2 are obtained as

$$H_1 = \{c_1^1, c_2^1, \cdots, c_i^2, c_{i+1}^2, \cdots, c_M^2\} \tag{28}$$

$$H_2 = \{c_1^2, c_2^2, \cdots, c_i^1, c_{i+1}^1, \cdots, c_M^1\} \tag{29}$$

Uniform crossover (UCX) This operator first select two crossover points randomly i and j. Two new solutions H_1 and H_2 are obtained as

$$H_1 = \left\{c_1^1, c_2^1, \cdots, c_i^2, c_{i+1}^1, \cdots, c_j^2, c_{j+1}^1, \cdots, c_M^1\right\} \tag{30}$$

$$H_2 = \left\{c_1^2, c_2^2, \cdots, c_i^1, c_{i+1}^2, \cdots, c_j^1, c_{j+1}^2, \cdots, c_M^2\right\} \tag{31}$$

Arithmetical crossover Two new solutions H_1 and H_2 are obtained as

$$H_1 = \left\{h_1^1, h_2^1, \cdots, h_i^1, h_{i+1}^1, \cdots, h_j^1, h_{j+1}^1, \cdots, h_M^1\right\} \tag{32}$$

$$H_2 = \left\{h_1^2, h_2^2, \cdots, h_i^2, h_{i+1}^2, \cdots, h_j^2, h_{j+1}^2, \cdots, h_M^2\right\} \tag{33}$$

where h_i^1 and h_i^2 are

$$h_i^1 = \lambda c_i^1 + (1-\lambda)c_i^2 \tag{34}$$

$$h_i^2 = \lambda c_i^2 + (1-\lambda)c_i^1 \tag{35}$$

where $\lambda \in [0,1]$.

Geometrical crossover Two new solutions H_1 and H_2 are obtained as

$$H_1 = \left\{h_1^1, h_2^1, \cdots, h_i^1, h_{i+1}^1, \cdots, h_j^1, h_{j+1}^1, \cdots, h_M^1\right\} \tag{36}$$

$$H_2 = \left\{h_1^2, h_2^2, \cdots, h_i^2, h_{i+1}^2, \cdots, h_j^2, h_{j+1}^2, \cdots, h_M^2\right\} \tag{37}$$

where h_i^1 and h_i^2 are

$$h_i^1 = c_i^{1^w} * c_i^{2^{(1-w)}} \tag{38}$$

$$h_i^2 = c_i^{2^w} * c_i^{1^{(1-w)}} \tag{39}$$

where $w \in [0,1]$.

BLX-α crossover Two new solutions H_1 and H_2 are obtained as

$$H_1 = \left\{h_1^1, h_2^1, \cdots, h_i^1, h_{i+1}^1, \cdots, h_j^1, h_{j+1}^1, \cdots, h_M^1\right\} \tag{40}$$

$$H_2 = \left\{h_1^2, h_2^2, \cdots, h_i^2, h_{i+1}^2, \cdots, h_j^2, h_{j+1}^2, \cdots, h_M^2\right\} \tag{41}$$

where h_i^1 and h_i^2 are randomly chosen from the interval $[C_{min} - I\alpha,\ C_{max} + I\alpha]$, where

$$C_{max} = Max\{c_i^1, c_i^2\} \tag{42}$$

$$C_{min} = Min\{c_i^1, c_i^2\} \tag{43}$$

$$I = C_{max} - C_{min} \tag{44}$$

The above crossover operators are able to do exploration and exploitation. Exploration is that they generate additional diversity and exploitation is to use the current diversity to produce better offsprings. We will explain these above crossovers, and the exploration and exploitation in detail (for our machining condition optimization) with a numerical example in the Appendix.

Mutation operators are used to introduce diversity in the population. We present the mutation operation used in our study.

Heuristic mutation Let C_1 be the solution selected for mutation operation.

$$C_1 = \left\{c_1^1, c_2^1, \cdots, c_i^1, c_{i+1}^1, \cdots, c_j^1, c_{j+1}^1, \cdots, c_M^1\right\} \tag{45}$$

In this operator, a single mutation point is randomly selected. The offspring produced in this operation is H_1 and is

$$H_1 = C_1 + \eta_m \lambda \delta \tag{46}$$

where η_m is the mutation rate, $-1 \leq \lambda \leq 1$, and δ is $I/2$, where I is given in BLX-α crossover.

Non-uniform mutation Let C_1 be the solution selected for mutation operation. In C_1, let the element selected for mutation be c_i. The offspring generated is H_1 and is

$$H_1 = \left\{c_1^1, c_2^1, \cdots, c_i^{1'}, c_{i+1}^1, \cdots, c_j^1, c_{j+1}^1, \cdots, c_M^1\right\} \tag{47}$$

In the above expression, the value of $c_i^{1'}$ is randomly selected from the two possibilities given below.

$$c_i^{1'} = c_i^1 + \Delta\left(t, c_i^{1U} - c_i^1\right) \tag{48}$$

$$c_i^{1'} = c_i^1 - \Delta\left(t, c_i^1 - c_i^{1L}\right) \tag{49}$$

where c_i^{1U} and c_i^{1L} are upper bound and lower bound for the element c_i^1. This mutation operator is a function of the generation in which it operates. In the initial generations (t is small), this operator searches the space uniformly and searches the space locally at later generations (t is large). This function $\Delta(t, dx)$ is

$$\Delta(t, dx) = dx.r.(1 - t/T)^d \tag{50}$$

where r is a random number from the interval [0.1], T is the maximum number of generations, and d is a parameter determining the degree of dependency (usually assumed as 2).

Fitness function Fitness is the driving force in genetic algorithms. The only information used in the execution of genetic algorithms is the observed value of fitness of the solution present in the population. The objective in our machining condition optimization problem is to determine the decision variables such that the production cost is a minimum. The calculation of fitness is easy. The string (solution representation) gives the value of decision variables. Once the value of decision variables are known the cost can obtained from the objective function. Since, the genetic algorithm maximizes the fitness function, the fitness (F) is defined as

$$F = -\{minimum\ Cost\} \tag{51}$$

The decision variables in the string may violate the constraints for the given problem (solution is infeasible). In that case, first the repair algorithm is applied to the solution and then evaluates the fitness. Another method of handling the constraints violation is based on a penalty function. In this method, the fitness is

$$F = -\{minimum\ Cost\} + r\left\{\sum_{i=1}^{K}\{g_i(x)\right\} \tag{52}$$

In the above expression, K is the number of constraints and $g_i(x)$ is whether the constraint i is satisfied or not. If the constraint i is satisfied then $g_i(x)$ is zero and if the constraint i is not satisfied then $g_i(x)$ is one. The penalty parameter (r) is a large value. In this way, the infeasible solutions are assigned a very large objective function value.

Termination criteria In genetic algorithms, the evolution process continues until a termination criterion is satisfied. The most frequently used termination criteria are population convergence criteria and a specified maximum number of generations. In this study, a specified maximum number of generations is used as termination criterion.

4 Results and discussions

The real-coded genetic algorithm for obtaining the optimal machining conditions is done on a Pentium-IV machine. The real-coded genetic algorithm used the following parameters: S_c crossover probability 0.25, S_m mutation probability 0.1, roulette wheel selection, the maximum number of generation is 1000, and the population size is 50. The following steps are carried out in real-coded genetic algorithm for obtaining the optimal machining conditions for all the models presented earlier.

Step 1. Initialization: An initial population of solutions of size N (N=50) is generated using population initialization.

Step 2. Evaluation: The fitness of each solution is calculated according to the fitness function. The fitness function is negative of the objective function (minimum cost) for the given solution.

Step 3. Selection: Perform selection using roulette wheel selection, to select solutions for genetic operations.

Table 1 Optimal machining conditions Model.1

Parameter	Simulated annealing	Continuous simulated annealing	Genetic algorithm	Generalized reduced gradient	Real-coded genetic algorithm
V^*	148.215	148.219	147.710	151.55	147.925
f^*	0.3617	0.3617	0.3614	0.375	0.3616
Min Cost	79.544	79.542	79.569	–	79.554

Step 4. Genetic operations: Perform crossover and mutation operations based on probability of crossover and mutation. Here we may get more solutions than the population size (N). Perform reproduction operation using elitist strategy to obtain N best solutions.

Step 5. Repeat the steps 2, 3, and 4 until the algorithm converges.

We can see that the real-coded genetic algorithm uses the survival of fittest strategy by passing the good solutions to the next generation of solutions, and combining different solutions to explore new solutions. In this manner, the algorithm converges to the optimal solution. An analysis of convergence for real-coded genetic algorithm is discussed in [23]. We now present the numerical results obtained for the machining models presented in the earlier section.

The results obtained using real-coded genetic algorithm for Model.1 is shown in Table 1. For comparing the results obtained from real coded genetic algorithm with earlier studies, we also use n=2 and d=2.5. Hence, the two decision variables are V the cutting speed and f the feed rates. The optimal values of cutting speed V^* and feed rate f^* are shown in Table 1. We also present the results obtained in earlier studies using different optimization algorithms for this model. The convergence of real-coded genetic algorithm for Model.1 is shown in Fig. 1.

In the above results for Model.1, given in Table 1, generalized reduced gradient method does not find the optimal solution. The values of V (151.55) and f (0.375) given in [3] are not feasible solutions. This can be easily verified, because the tool life constraint is violated. The tool life constraint is $25 \leq TL \leq 45$. With these values of V (151.55) and f (0.375) the value of TL=21.0069.

The results obtained using real-coded genetic algorithm for Model.2 to Model.5 are shown in Tables 2, 3, 4, and 5 respectively, along with the results obtained in earlier studies.

The advantage with real-coded genetic algorithms is that it starts with solutions generated randomly and modify the solutions in successive generations, and upon termination we obtain the optimal solution. For Model.1, we have the real-coded genetic algorithm with three types of crossovers. The crossovers used are average crossover, arithmetical crossover with λ=0.25, and geometric crossover with λ=0.25. The real coded genetic algorithm for each model is run 100 times (runs) with these three crossover methods. The results obtained for each crossover after 100 times (runs) is shown in Table 6 and feasible values in Table 7.

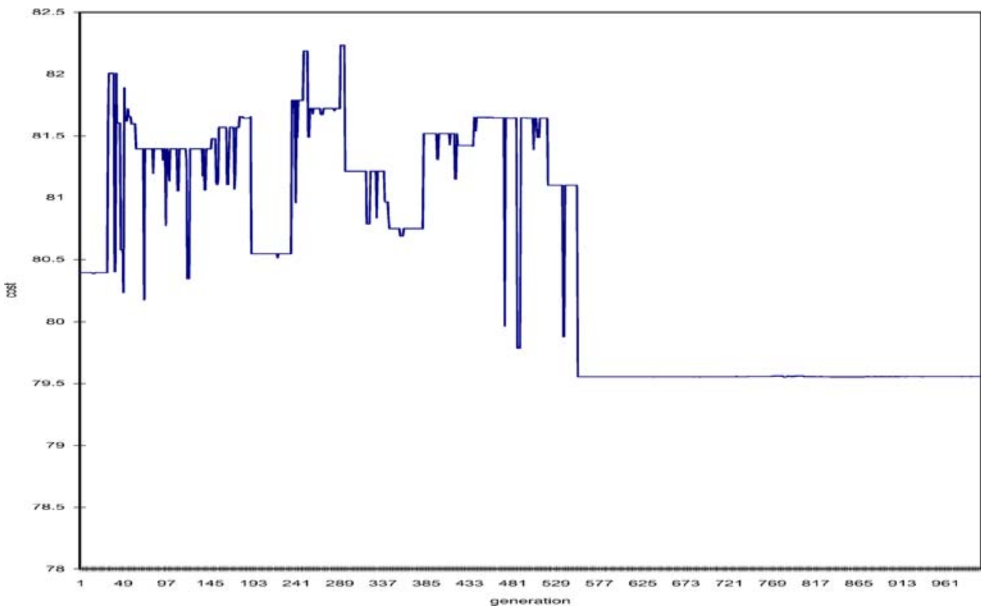

Fig. 1 Convergence of real-coded genetic algorithm for Model.1

Table 2 Optimal machining conditions Model.2

Parameter	Simulated annealing	Continuous simulated annealing	Genetic algorithm	Generalized reduced gradient	Real-coded genetic algorithm
$V*$	143.908	143.9140	145.068	143.90	143.9037
$f*$	0.001439	0.001439	0.001423	0.0014	0.001439
Min Cost	6.2550	6.2551	6.2758	6.26	6.255718

5 Discussions

Now, we will discuss various issues in applying real-coded genetic algorithm to machining condition optimization problem.

We have presented many types of crossover and mutation operators. Each crossover operator produces two new solutions and each mutation operator produce one new solution. It is natural to ask which crossover and mutation operator is to be used for solution. In fact, the type of crossover and mutation operator to be used depends on a particular problem. It is observed in real-coded genetic algorithms, that one type crossover that performs well for a problem may not perform well for a different problem. Even in the same problem, one type of crossover may perform well in earlier stages of the problem and may not perform well in the later stages of the same problem. This fact has been brought out in *no free lunch theorem*, presented in [24]. Because of this, it is suggested in [21] to apply different crossover operators simultaneously on the population for practical situations. The advantage is that the search becomes faster and the computation time is reduced. Our interest is to show the applicability of real-coded genetic algorithm to the machining condition optimization problem, and so we have used only one type of crossover in our study. We have shown the performance of different crossover types for Model.1 in Table 6.

It is possible in some of the crossover operators to obtain more than two new solutions. In our study, we deal with crossover operators that need only two parents and produce two solutions. In our study, in the average crossover, we obtain only one offspring. Hence, we use the better solution from one of the parents as another solution for next generation.

The crossover operators described above are able to produce exploration and exploitation. Exploration generate additional diversity and exploitation generate better solutions. The average and geometric crossovers are exploitative crossovers when $\lambda \in [0,1]$. When $\lambda<0$ or $\lambda>1$ produces exploration. BLX-α crossover is explorative crossover. We will explain further about exploration and exploitation, with a numerical example in the Appendix. The computation time for average, arithmetical and geometric crossover are 1.541, 1.652, and 1.584 s on PentiumIV 3.4 GHz machine.

The number function evaluations or the number of solutions generated in our approach can be obtained as follows: Let N be the size of initial population, S_c is the probability of crossover, S_m is the probability of mutation, and T is the maximum number of generation. Then the number of solutions generated is $N * S_c * T + N * S_m * T$. In our studies, we have used 1000 as the maximum number of generations, but we observe that the optimal solution is reached around the 600 generation. The results presented in Tables 1, 2, 3, 4, and 5 show that this real coded genetic algorithm perform very well for the machining condition optimization models. The number of function evaluation for Models.1 to 5, (with S_c=0.25, S_m=0.1, N=50 and T=1000) in one simulation run are 14036, 11412, 14057, 14090 and 14128, respectively.

From the results shown in Tables 1, 2, 3, 4, and 5, we see that all the methods are able to obtain solution very close to the optimal result given by continuous simulated annealing. Our interest in this paper is to study the performance of real coded genetic algorithm in terms of crossover rate, mutation rate, and number of function evaluations. We can see that the results obtained by real coded genetic algorithm performs well for all the models.

We have also conducted a study with different crossover and mutation rates for Model.1. The results (the value of objective function) are shown in Table 8. From Table 8, we can see that for this problem the crossover rate of 0.3 and mutation rate of 0.1 gives the best result.

Table 3 Optimal machining conditions Model.3

Parameter	Simulated annealing	Continuous simulated annealing	Genetic algorithm	Generalized reduced gradient	Real-coded genetic algorithm
$V*$	174.394	174.2229	174.399	174.38	174.4137
$f*$	0.2321	0.2321	0.2321	0.232	0.232066
Min Cost	12.097	12.096	12.099	12.10	12.09861

Table 4 Optimal machining conditions Model.4

Parameter	Simulated annealing	Continuous simulated annealing	Genetic algorithm	Generalized reduced gradient	Real-coded genetic algorithm
V^*	433.980	441.2849	434.398	433.60	433.5461
f^*	0.003814	0.003908	0.003816	0.0038	0.003808
Min Cost	1.5526	1.5526	1.5533	1.553	1.552639

6 Conclusions

The non-linear constrained optimization problem of obtaining optimal combination of machining conditions (parameters) is considered. A real coded genetic algorithm (RCGA) to find the optimal combination of machining conditions is presented. Five machining condition optimization models presented in an earlier study are solved using the real coded genetic algorithm. It is shown in Tables 1, 2, 3, 4, and 5 that our real coded genetic algorithm is able to obtain the optimal machining parameters. From the results shown in Tables 1, 2, 3, 4, and 5, we conclude that real coded genetic algorithm is reliable and accurate for solving the machining condition optimization models. Various issues related to this approach are discussed.

Acknowledgement This work was supported by the Kangwon National University BK21 project and Kangwon Institute of Telecommunication and Information.

Appendix

In this appendix, we will explain various issues related to real-coded genetic algorithm in detail. We use the machining condition optimization models presented to explain various issues.

Solution representation In real-coded genetic algorithms, the solution represented as a string of real numbers. The number elements in the string is the number of decision variables. Consider Models.1–4. The length of the string is 2. The first element is the cutting speed (V) and the second element is the feed rate (f). For example, C_1 and C_2 are the two solutions.

$$C_1 = \{c_1^1, c_2^1\}$$
$$C_2 = \{c_1^2, c_2^2\}$$

Population initialization One way of generating the initial population is to generate randomly in the intervals for the decision variables. For Model.1, the allowable range for V and f are given as

$$50 \leq V \leq 400\, m/min$$
$$0.30 \leq f \leq 0.75\, mm/rev$$

So the elements in the solutions C_1 and C_2 are

$$50 \leq c_1^1, c_1^2 \leq 400\, m/min$$
$$0.30 \leq c_2^1, c_2^2 \leq 0.75\, mm/rev$$

For example, we may generate randomly these five solutions: C_1={156.5505,0.3435}, C_2={354.7893,0.6572}, C_3={127.8794,0.5584}, C_4={247.4532,0.1583}, and C_5={78.4532,0.4531}. A valid solution to Model.1 should satisfy the constraints presented. Because of the constraints, some or all the above may be invalid solutions. So we need some sort of "repair algorithm". Repair algorithm would "repair" the solution and make it a valid solution [14].

Repair algorithm If a solution violates the constraints, then either the value of V or f or both are increased or decreased within the range and obtain a new solution. Check if the new solution is a valid solution. In this manner, we can obtain the initial population in which all the solutions are valid solution. Otherwise, we can discard the invalid solutions and generate solutions randomly in the intervals for the decision variables.

Another way is to preprocess the constraints and obtain the initial population. By preprocessing we mean obtaining the initial population from feasible regions. For Model.1, we know that the constraint on tool life (TL) is $25 \leq TL \leq$

Table 5 Optimal machining conditions Model.5

Parameter	Simulated annealing	Continuous simulated annealing	Genetic algorithm	Generalized reduced gradient	Real-coded genetic algorithm
V^*	216.013	216.0618	216.108	216.08	216.1428
f^*	0.3886	0.3886	0.3879	0.388	0.388214
Min Cost	108.0332	108.0177	108.093	108.33	108.0049

Table 6 Results of each crossover after 100 runs for Model.1

Objective function	Average crossover	Arithmetical crossover	Geometric crossover
Minimum	79.54832	79.55549	79.549
Maximum	80.02427	80.15685	79.98157
Average	79.6688	79.71637	79.69231
St-Deviation	0.100176	0.134369	0.094498

45 *min*. The tool life constraint is a function of V and f and the functional relationship is:

$$TL = 60\left\{\frac{10^{10}}{V^5 f^{1.75} d^{0.75}}\right\}$$

In the above equation, we use the value of f from 0.30 to 0.75, for the value of TL=25, and TL=45. The values of V obtained are shown in Table 7.

From this Table 7, for example, we can choose f=0.40 and choose V with in the range 127.2260 to 143.0969. Like this, we can generate the initial population of solutions. Note that for this Model.1, this is the only necessary condition for the solution to be feasible because we are using only one constraint. There are three other constraints and so this is not a sufficient condition for the solution to be feasible. We will explain this further by using Model.2.

Consider the Model.2 presented earlier. There are three physical constraints in this model. These constraints are on surface finish (SF), feed rate (f), and the horse power (HP). These constraints are $SF \leq 100\ \mu in$, $f \leq 0.01$ *in/rev*, and $HP \leq 2hp$. These constraints are functions of cutting speed (V) and feed rate (f) and the functional relationships are:

$$SF = 1.36 \times 10^8 V^{-1.52} f^{1.004}$$
$$f \leq 0.01$$
$$HP = 3.58 V^{0.91} f^{0.78}$$

In the above equation, we use the value of f from 0.001 to 0.01, for the value of SF=100. Then, we use the value of f from 0.001 to 0.01, for the value of HP=2.0. Using, these we obtain the feasible region for obtaining the initial population of solutions. The feasible region is shown in Fig. 2. In Fig. 2, the feasible region is below the curve SF=100 and above the curve HP=2.0. In a similar manner, we obtain the feasible region for Model.3 and it is shown in Fig. 3. In this figure, the feasible region is below the curve R_a=2 and above the curve P_c=5.5. The value of f is from 0.001 to 0.4 in the X-axis.

When we generate the initial population of solutions from the feasible regions, all the solutions in the population are valid solution. This is because in Model.2 and Model.3 there are only two constraints and both these constraints are considered, while obtaining the feasible regions.

Crossover operators Now, we will explain the working of crossover operators presented earlier. For this purpose, we consider Model.3. Let C_1={200, 0.15} and C_2={125, 0.10} are the two solutions selected for crossover operations. In this Model.3, there are only two decision variables. Hence, two-point crossover, simple crossover, and uniform crossover produces the following two offsprings H_1={200, 0.10}, and H_2={125, 0.15}. We can see that the offspring H_1 is a valid solution but the offspring H_2 is not a valid solution. We can use the repair algorithm for the second element f and make it a valid solution.

– *Arithmetical crossover:* Two new solutions H_1 and H_2 are obtained as H_1={143.75, 0.1175}, and H_2={181.25, 0.1375}, when the value of λ=0.25. When the value of λ=0.40, the two new solutions H_1 and H_2 are obtained as H_1={155, 0.12}, and H_2={170, 0.13}. It can be easily seen that when λ=0.5 arithmetical crossover is averaging crossover and hence H_1=H_2={162.5, 0.125}.

Table 7 Feasible values of V and f for Model.1

f	TL=25 V_{Max}	TL=45 V_{Min}
0.30	158.2555	140.7033
0.35	149.9434	133.3131
0.40	143.0969	127.2260
0.45	137.3178	122.0878
0.50	132.3463	117.6677
0.55	128.0042	113.8072
0.60	124.1647	110.3936
0.65	120.7346	107.3438
0.70	117.6432	104.5954
0.75	114.8365	102.0909

Table 8 Results with different crossover and mutation rates Model.1

Crossover rate	Mutation rate				
	0.10	0.30	0.50	0.70	0.90
0.10	79.71164	79.68995	79.69732	79.73374	79.82544
0.30	79.67518	79.68099	79.68615	79.70361	79.83778
0.50	79.67692	79.68237	79.68961	79.69081	79.80294
0.70	79.69019	79.69938	79.68739	79.70938	79.81622
0.90	79.70469	79.69050	79.68043	79.71124	79.74657

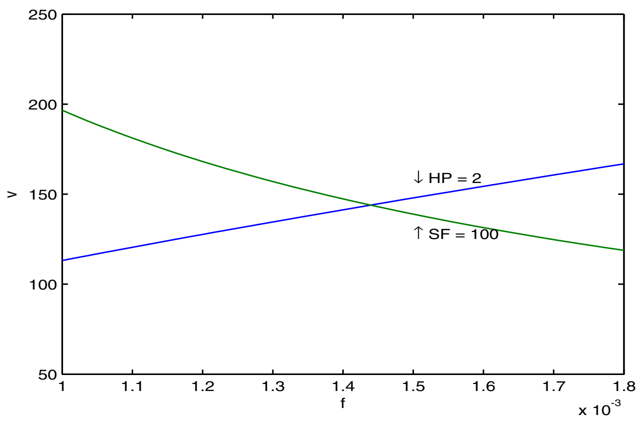

Fig. 2 Feasible region for Model.2

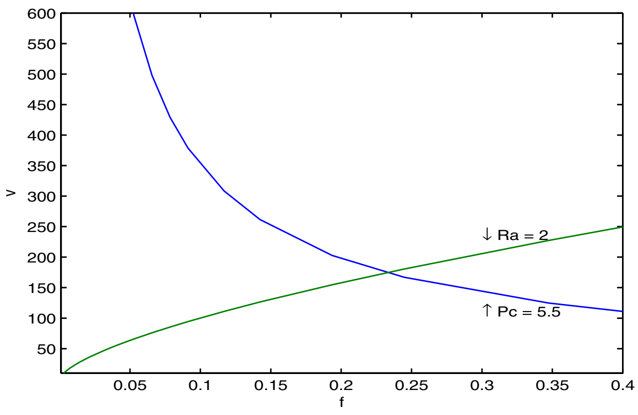

Fig. 3 Feasible region for Model.3

- *Geometrical crossover:* Two new solutions H_1 and H_2 are obtained as $H_1=\{140.5853,\ 0.1107\}$, and $H_2=\{177.8279,\ 0.1355\}$, when the value of $w=0.25$. When the value of $w=0.40$, the two new solutions H_1 and H_2 are obtained as $H_1=\{150.8544,\ 0.1176\}$, and $H_2=\{165.7227,\ 0.1275\}$.
- *BLX-α Crossover:* Two new solutions H_1 and H_2 are obtained as follows: We know that $C_1=\{200,\ 0.15\}$ and $C_2=\{125,\ 0.10\}$ are the two solutions selected for BLX-α crossover operations. Let $\alpha=0.30$.

$$H_1 = \{h_1^1, h_2^1\}$$
$$H_2 = \{h_1^2, h_2^2\}$$

$$C_{max} = Max\{c_1^1, c_1^2\} = Max\{200, 125\} = 200$$
$$C_{min} = Min\{c_1^1, c_1^2\} = Min\{200, 125\} = 125$$
$$I = C_{max} - C_{min} = 200 - 125 = 75$$

where h_1^1 and h_1^2 are randomly chosen from the interval $[C_{min} - I\alpha,\ C_{max} + I\alpha]$. This interval is [102.5, 222.5].

Similarly

$$C_{max} = Max\{c_2^1, c_2^2\} = Max\{0.15,\ 0.10\} = 0.15$$
$$C_{min} = Min\{c_2^1, c_2^2\} = Min\{0.15,\ 10\} = 0.10$$
$$I = C_{max} - C_{min} = 0.15 - 0.10 = 0.05$$

where h_2^1 and h_2^2 are randomly chosen from the interval $[C_{min} - I\alpha,\ C_{max} + I\alpha]$. This interval is [0.085, 0.165].

The arithmetical and geometrical crossovers will produce exploitation when $\lambda \in [0,1]$, and $w \in [0,1]$. Exploitation means the values of the offsprings lie with in the range of the parents; i.e., the first decision variable will be in the range [125, 200], and the second decision variable will be in the range [0.10, 0.15]. When $\lambda<0$ or $\lambda>1$ in arithmetical crossover, and $w<0$ or $w>1$ in geometrical crossover produces exploration.

BLX-α Crossover is explorative crossover. Exploration means the values of the offsprings does not lie with in the range of the parents; i.e., when $\alpha=0.30$, the first decision variable will be in the range [102.5, 222.5], and the second decision variable will be in the range [0.085, 0.165].

References

1. Khan Z, Prasad B, Singh T (1997) Machining condition optimization by genetic algorithms and simulated annealing. Comput Oper Res 24:647–657
2. Wang X, Jawahir IS (2005) Optimization of multi-pass turning operations using genetic algorithms for the selection of cutting conditions and cutting tools with tool wear effect. Int J Prod Res 43:3543–3559
3. Duffuaa SO, Shuaib AN, Alam A (1993) Evaluation of optimization methods for machining economic models. Comput Oper Res 20:227–237
4. Hati SK, Rao SS (1975) Determination of machining conditions probabilistic and deterministic approaches. J Engng Indust Trans ASME Paper No.75-Prod-K
5. Ermer DS (1971) Optimization of the constrained maching economics problem by geometric programming. Trans ASME 93:1067–1072
6. Petropoulos P (1973) Optimal selection of machining variables using geometric programming. Int J Prod Res 11:305–314
7. Ermer DS, Kromodihardjo S (1981) Optimization of multipass turning with constraints. Trans ASME J Eng Ind 103:462–468
8. Iwata K, Murotsu Y, Obe F (1977) Optimization of cutting conditions for multipass operations considering probabilistic nature in machining processes. Trans ASME J Eng Ind Series B 210–217
9. Shanmugham MS, Baskara Reddy SV, Narendran TT (2000) Selection of optimal conditions in multi-pass face milling using a genetic algorithm. Int J Mach Tool Manuf 40:401–414
10. Basker N, Asokan P, Saravanan R, Prabhaharan (2005) Optimization of machining parameters for milling operations using non-conventional methods. Int J Adv Manuf Technol 25:1078–1088

11. Holland HJ (1975) Adaptation in natural and artificial systems. Univ Michigan Press, Ann Arbor, USA
12. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, New York, USA
13. David L (1991) Handbook of genetic algorithms. Van Nostrand, New York, USA
14. Michalewicz Z (1994) Genetic algorithms + data structures = evolution programs. AI Series. Springer, Berlin Heidelberg New York
15. Fogel DB (1995) A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems. Simulation 64:397–404
16. Renders JM, Flasse SP (1996) Hybrid methods using genetic algorithms for global optimization. IEEE Trans Syst Man Cybern Part B 26:243–258
17. Beyer HG, Deb K (2001) On self-adaptive features in real-parameter evolutionary algorithms. IEEE Trans Evol Comput 5:250–270
18. Menzura-Montes E, Coello Coello CA (2005) A simple multi-membered evolution strategy to solve constrained optimization problems. IEEE Trans Evol Comput 9:1–17
19. Venkataraman S, Yen GG (2005) A generic framework for constrained optimization using genetic algorithms. IEEE Trans Evol Comput 9:424–435
20. Takahama T, Sakai S (2005) Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. IEEE Trans Evol Comput 9:437–451
21. Herrera F, Lozano M, Sanchez AM (2003) A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study. Int J Intell Syst 18:309–338
22. Houck CR, Joines JA, Kay MG (1996) A Genetic algorithm for function optimization: a matlab implementation. ACM Trans Math Softw 22:1–14
23. Goldberg DE (1991) Real-coded genetic algorithms virtual alphabets, and blocking. Complex Syst 5:139–167
24. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 11:67–82