Int J Adv Manuf Technol (2009) 40:1118–1124
DOI 10.1007/s00170-008-1418-4

ORIGINAL ARTICLE

A hybridized approach to optimal tolerance synthesis of clutch assembly

Erwie Zahara · Yi-Tung Kao

Received: 28 February 2007 / Accepted: 23 January 2008 / Published online: 19 February 2008

Abstract Specifying proper tolerances for manufactured goods results in greater savings and improved performance, which may ultimately determine whether a product succeeds or fails in the marketplace. In the past, tolerance specification has been more an art than a science, and is largely dependent upon experiences. A more scientific and reliable approach is presented in this paper. A hybrid of Nelder-Mead simplex method and particle swarm optimization (NM-PSO) is introduced for the design of tolerance of the machine elements of an overrunning clutch assembly. The objective is to obtain tolerances of the individual components so that the cost of manufacturing and quality loss is minimized. Experimental results demonstrate that hybrid NM-PSO is extremely effective and efficient in locating best-practice optimal solutions compared to geometric programming (GP), genetic algorithm (GA), and particle swarm optimization (PSO) methods.

Keywords Tolerance optimization · Nelder-Mead simplex search method · Particle swarm optimization

E. Zahara (✉)
Department of Industrial Engineering and Management,
St. John's University,
Tamsui, Taiwan 251, Republic of China
e-mail: erwi@mail.sju.edu.tw

Y.-T. Kao
Department of Computer Science and Engineering,
Tatung University,
Taipei City, Taiwan 104, Republic of China

1 Introduction

Tolerance is born out of the need for the parts of a product to be interchangeable and mass producible. Tolerance synthesis or tolerance allocation is an action that is used to determine the amount of clearance a machined part must have to properly correspond with other machined parts and perform efficiently. Though tight tolerances favor product performance, loose tolerances are often preferred over tight ones because they maximize yield and lower production costs. How these competing requirements are balanced has a direct bearing on the manufacturing cycle time, quality, and cost of a product. Thus, tolerance optimization is of utmost importance in product production, and it should take into account all aspects of the matter, which include limitations of manufacturing processes, functionality, and assembly constraints and costs.

Tolerance synthesis methods can be categorized into three classes: methods based on conventional optimization approaches, methods based on quality engineering, and methods based on heuristic optimization methods and neural networks. The first group accounts for the majority of the publications on tolerance synthesis and mostly uses the cost-tolerance models (Michael and Siddall [1]). This group of methods becomes impractical as the complexity of mechanical assemblies increases. The third group of methods attempts to address the complexity issue by some novel approaches based on genetic algorithm, simulated annealing, particle swarm optimization, neural networks, and fuzzy logic. Kopardekar and Anand [2] applied neural network techniques to tolerance allocation. Dupinet et al. [3] exploited fuzzy logic and simulated annealing, while Ji et al. [4] and Noorul Haq et al. [5] applied the genetic algorithm. Recently, Noorul Haq et al. [6] applied particle swarm optimization to tolerance optimization for clutch

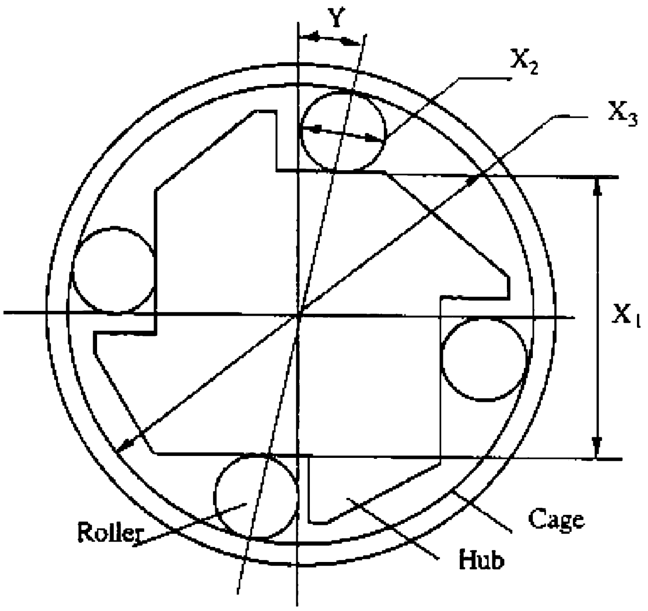

Fig. 1 Overrunning clutch

assembly. In this paper, a hybrid Nelder-Mead simplex search method and particle swarm optimization (NM-PSO) is applied to solve tolerance optimization for clutch assembly.

2 Tolerance optimization for a clutch assembly

The overrunning clutch model for which the tolerance is to be optimized was proposed by Feng and Kusiak [7] and is shown in Fig. 1. The model consists of three components: hub, roller, and cage, denoted by X_1, X_2 and X_3, respectively. The contact angle Y is the functional dimension that must be controlled with the tolerance stack up limit and is expressed as:

$$Y = f(X_1, X_2, X_3) = a\cos\left(\frac{X_1 + X_2}{X_3 - X_2}\right) \tag{1}$$

where a is constant. The cost tolerance data for the clutch (tolerance in 10^{-4} in., cost in dollars) is given in Table 1.

The nominal values of X_1, X_2, X_3 are 2.17706, 0.90000 and 4.00000 in., respectively. The nominal value and the tolerance of angle Y are 0.122±0.035 rad. The derivatives of Y in respect to $X_i (i=1, 2, 3)$ are

$$\left|\frac{\partial f}{\partial X_1}\right| = \frac{1}{X_3 - X_2}\left(1 - \frac{X_1 + X_2}{X_3 - X_2}\right)^{-1/2} \tag{2}$$

$$\left|\frac{\partial f}{\partial X_2}\right| = \frac{X_3 + X_1}{(X_3 - X_2)^2}\left(1 - \frac{X_1 + X_2}{X_3 - X_2}\right)^{-1/2} \tag{3}$$

$$\left|\frac{\partial f}{\partial X_3}\right| = \frac{X_1 + X_2}{(X_3 - X_2)^2}\left(1 - \frac{X_1 + X_2}{X_3 - X_2}\right)^{-1/2} \tag{4}$$

Table 1 Cost-tolerance data for the clutch tolerance (tolerance in 10^{-4} in., cost in dollars)

Hub tolerance	Cost	Roll Tolerance	Cost	Cage tolerance	Cost
2	19.38	1	3.513	1	18.637
4	13.22	2	2.48	2	12.025
8	5.99	4	1.24	4	5.732
16	4.505	8	1.24	8	2.686
30	2.065	16	1.20	16	1.984
60	1.24	30	0.413	30	1.447
120	0.825	60	0.413	60	1.200
–	–	120	0.372	120	1.033

Given X=(2.17706, 0.9000, 4.0000) in., then $\left|\frac{\partial f}{\partial X_1}\right| = 3.750\,\mathrm{rad/inches}$, $\left|\frac{\partial f}{\partial X_2}\right| = 7.472\,\mathrm{rad/inches}$ and $\left|\frac{\partial f}{\partial X_3}\right| = 3.722\,\mathrm{rad/inches}$.

Using regression analysis for the three components, the manufacturing cost functions have been derived as follows [7]:

$$\text{Hub} \quad M(t_1) = -0.731 + \frac{0.0580}{t_1^{0.688}} \tag{5}$$

$$\text{Roll} \quad M(t_2) = -8.3884 + \frac{5.7807}{t_2^{0.0784}} \tag{6}$$

$$\text{Cage} \quad M(t_3) = 0.978 + \frac{0.0018}{t_3} \tag{7}$$

where t_1, t_2, t_3 are the single side tolerance value of the hub, roller, and cage.

Cost associated with quality loss function is

$$Q(t_i) = \sum_{k=1}^{K} \frac{A}{T_k^2}\sigma_k^2 \tag{8}$$

where $\sigma_k^2 = \left(\frac{t_i}{3}\right)^2$ is the standard deviation of dimensional chain k, A is the quality loss coefficient, T_k is the single side functional tolerance stack up limit for dimensional chain $k (T_k=0.035)$, K is the total number of dimensional chains, k is the dimensional chain index.

The models optimizing the manufacturing cost and quality loss cost with the worst-case stack up constraint as:

$$Min\ COF(t_i) = \sum_{i=1}^{3} M(t_i) + Q(t_i) \tag{9}$$

$$\text{s.t.} \sum\left(\left|\frac{\partial f}{\partial X_i}\right| t_i\right) \le T_k \tag{10}$$

Substituting (5)–(7) to (9), the combined objective function (COF) can be formulated:

$$Min\ COF(t_1,t_2,t_3) = -33.3066 + \frac{0.058}{t_1^{0.688}} + 4 \times \frac{5.7807}{t_2^{0.0784}} + \frac{0.0018}{t_3} + \frac{A}{(3)^2 \times (0.035)^2}\left(t_1^2 + 4t_2^2 + t_3^2\right) \tag{11}$$

$$\begin{aligned} \text{s.t.}\quad & 3.7499t_1 + 14.944t_2 + 3.722t_3 \le 0.0350 \\ & 0.0001 \le t_1 \le 0.0120 \\ & 0.0001 \le t_2 \le 0.0005 \\ & 0.0001 \le t_3 \le 0.0120 \end{aligned}$$

Note that $M(t_2)$ has been multiplied by 4 to account for the four rollers present in the assembly.

3 Hybrid optimization method

The goal of integrating Nelder-Mead (NM) simplex method and particle swarm optimization (PSO) is to combine their advantages and avoid disadvantages. For example, NM simplex method is a very efficient local search procedure but its convergence is excessively sensitive to the starting point selected; PSO belongs to the class of global search procedures but requires much computational effort (Fan et al. [8]). This section starts by introducing the procedure of NM and PSO, followed by a description of hybrid Nelder-Mead and particle swarm optimization methods.

3.1 The procedure of NM

The simplex search method proposed by Nelder and Mead [9] is a derivative-free line-search method that was particularly designed for traditional unconstrained minimization scenarios, such as the problems of nonlinear least squares, nonlinear simultaneous equations, and other types of function minimization (see, e.g., Olsson and Nelson [10]). First, function values at the (N+1) vertices of an initial simplex are evaluated, which is a polyhedron in the factor space of N input variables. In the minimization case, the vertex with the highest function value is replaced by a newly reflected, better point, which would be approximately located in the negative gradient direction. Clearly, NM can be deemed as a direct line-search method of steepest descent kind. The ingredients of replacement process consist of four basic operations: reflection, expansion, contraction, and shrinkage. Through these operations, the simplex can improve itself and come closer and closer to a local optimum point sequentially. An example of the function minimization of two variables will illustrate the basic procedure of NM. Starting point B together with initial step sizes will construct an initial simplex design (shown as A, B, and C), as illustrated in Fig. 2. Suppose $f(A)$ is the highest of the three function values and is to be replaced. In this case, a reflection is made through the centroid of $\overline{BC}$ (with the midpoint D) to the point E. Suppose $f(C) < f(B) < f(A)$. At this stage, three situations can arise.

1. If $f(E) < f(C)$, an extension is made to point J. We then keep E or J as a replacement for A, depending on which function value is lower.
2. If $f(E) > f(C)$, a contraction is made to point G or H, depending on whether $f(A)$ or $f(E)$ is lower.
3. If $f(G)$ or $f(H)$ is larger than $f(C)$, the contraction has failed and we then perform a shrinkage operation. The shrinkage operation reduces the size of the simplex by moving all but the best point C halfway towards the best point C.

3.2 The procedure of PSO

Particle swarm optimization (PSO) is one of the latest evolutionary optimization techniques developed by Kennedy and Eberhart [11]. The PSO concept is based on a metaphor of social interaction such as bird flocking and fish schooling. Similar to genetic algorithms, PSO is also population-based and evolutionary in nature, with one major difference from genetic algorithms that it does not implement filtering, i.e., all members in the population survive through the entire search process. PSO simulates a commonly observed social

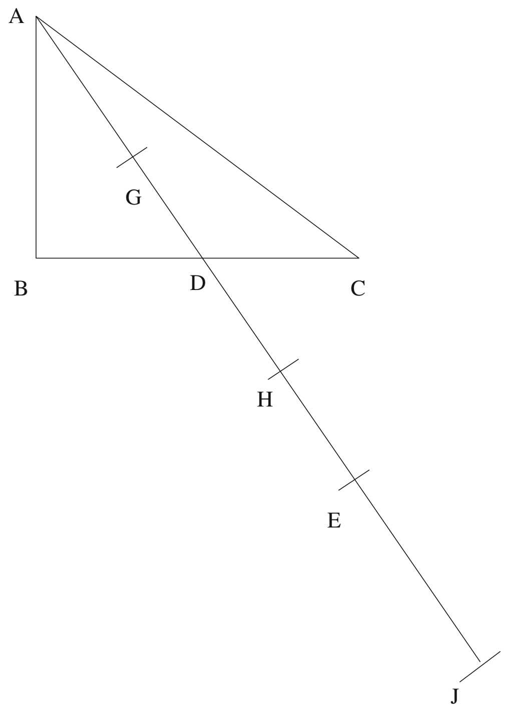

Fig. 2 NM operations of a two-dimensional case

Fig. 3 The hybrid NM-PSO algorithm

```
1. Initialization. Generate a population of size $3N+1$.
Repeat
  2. Evaluation & Ranking. Evaluate the fitness of each particle.
     Rank them based on the fitness.
  3. Simplex Method. Apply the NM operator to the top $N+1$ particles and replace
     the $(N+1)^{\text{th}}$ particle with the update.
  4. PSO Method. Apply PSO operator for updating $2N$ particles with worst fitness.
     Selection. From the population, select the global best particle and the
     neighborhood best particles.
     Velocity Update. Apply velocity update to the 2N particles with worst fitness
     according equations (12) and (13).
Until a termination condition is met.
```

behavior where members of a group tend to follow the lead of the best of the group. The procedure of PSO is illustrated as follows.

1. Initialization. Randomly generate a population of the potential solutions, called "particles", and each particle is assigned a randomized velocity.
2. Velocity Update. The particles then "fly" through 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:

$$V_{id}^{New} = w \times V_{id}^{old} + c_1 \times rand \times \left(p_{id} - x_{id}^{old}\right) + c_2 \times rand \times \left(p_{gd} - x_{id}^{old}\right), \qquad (12)$$

$$x_{id}^{New} = x_{id}^{old} + V_{id}^{New}, \qquad (13)$$

where c_1 and c_2 are two positive constants; w is an inertia weigh,t and rand is a uniformly generated random number from the range [0, 1] that is produced every time for each iteration. Eberhart and Shi [12] and Hu and Eberhart [13] suggested $c_1 = c_2 = 2$ and $w = [0.5 + rand/2.0)]$. Equation (12) shows that in 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 (p_{id}) and the global best location (p_{gd}) all contribute some influence on the outcome of velocity update. The global best location (p_{gd}) is identified, based on its fitness, as the best particle in a 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 discovered 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. The particles' velocities on each dimension are clamped to a maximum velocity V_{max}, which is confined to the range of the search space in each dimension. Equation (13) shows how each particle's position (x_{id}) is updated in the search of solution space.

Table 2 The results of NM-PSO for solving objective function (14)

A	t_1	t_2	t_3	$COF_{NM\text{-}PSO}$	Iteration	N_eval	Constraint
0	0.00494536	0.00050000	0.00241359	11.6375188	100	1004	0.0350000
1	0.00494417	0.00050000	0.00241479	11.6403559	90	907	0.0350000
52	0.00488258	0.00050000	0.00247685	11.7843228	187	1883	0.0350000
100	0.00482796	0.00050000	0.00253187	11.9186119	141	1415	0.0350000
300	0.00463631	0.00050000	0.00272496	12.4680642	110	1080	0.0350000
520	0.00425429	0.00049999	0.00267209	13.0471199	115	1164	0.0337070

Note: N_eval is the number of function evaluations

3.3 Hybrid NM-PSO

The population size of this hybrid NM-PSO approach is set at $3N+1$ when solving an N-dimensional problem. The initial population is created in two steps: using a predetermined starting point, N particles are spawned with a positive step size of 1.0 in each coordinate direction, and the other $2N$ particles are randomly generated. A total of

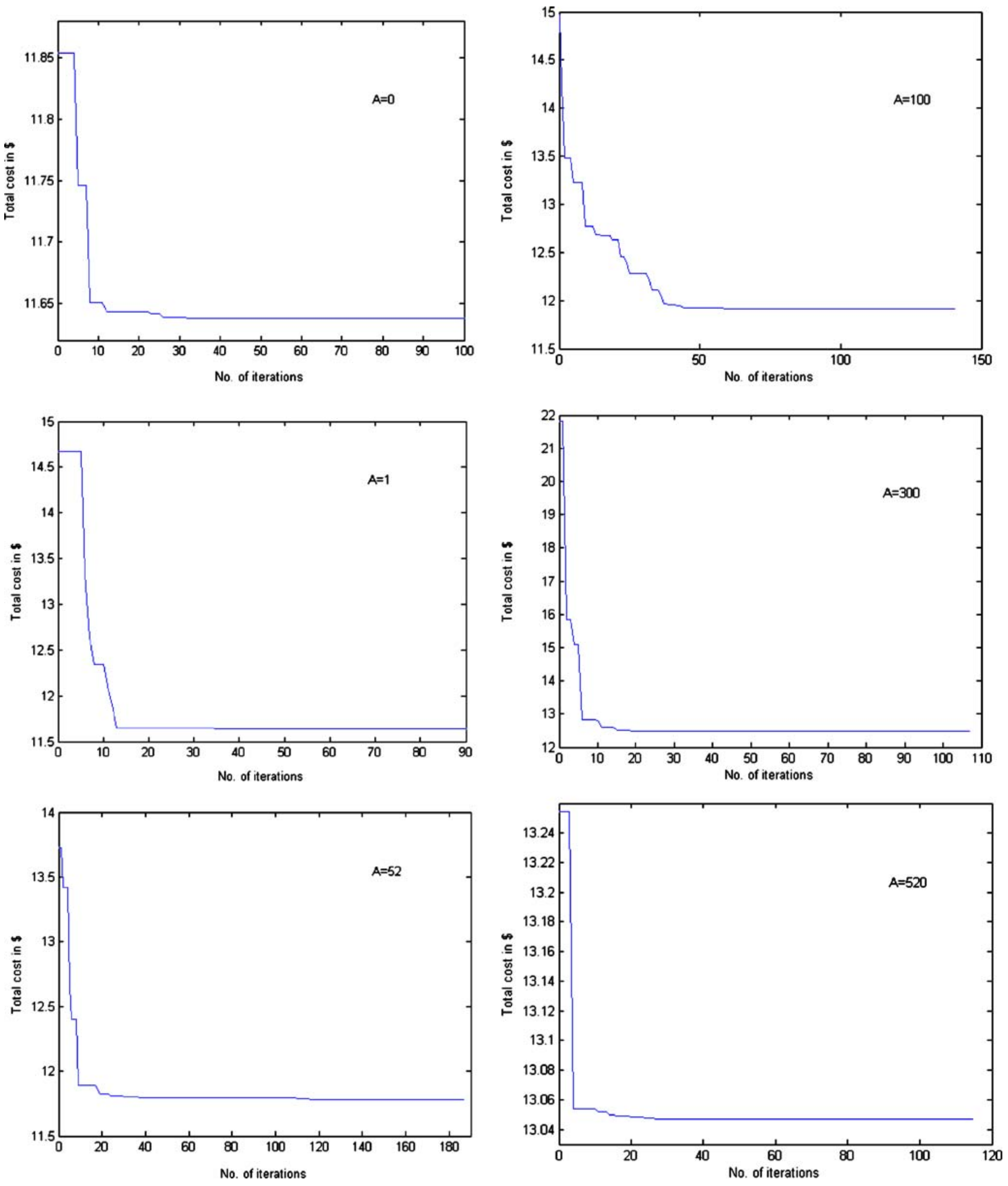

Fig. 4 Solution history of NM-PSO

$3N+1$ particles are sorted by the fitness, and the top $N+1$ particles are then fed into the simplex search method to improve the $(N+1)^{\text{th}}$ particle. The other $2N$ particles are adjusted by the PSO method by taking into account the positions of the $N+1$ best particles. This procedure for adjusting the $2N$ particles involves selection of the global best particle, selection of the neighborhood best particles, and finally velocity updates. The global best particle of the population is determined according to the sorted fitness values. The neighborhood best particles are selected by first evenly dividing $2N$ particles into N neighborhoods and denoting the particle with the better fitness value in each neighborhood as the neighborhood best particle. With Eqs. (12) and (13), a velocity update for each of the $2N$ particles is then carried out. The $3N+1$ particles are sorted in preparation for repeating the entire run. The process terminates when a certain convergence criterion is satisfied. In this case, the hybrid NM-PSO method is not an exhaustive search and we cannot determine its computational complexity. Figure 3 summarizes the algorithm of NM-PSO. For further details of the hybrid NM-PSO, see Fan et al. [8].

In Sect. 2, we have described the overrunning clutch model and the objective function (11) with one constraint, which is optimized by NM-PSO by solving for the best combination of the values of three variables hub, roll and cage t_1, t_2, t_3. As NM-PSO is an algorithm intended for solving unconstrained optimization problems, it is necessary to recast the objective function (11) into an unconstrained form by adding penalty term, as shown in (14).

$$Min\, f(t) = COF(t) + u(t) \times const(t)^2 \times 10^{20} \tag{14}$$

where $COF(t_1, t_2, t_3) = -33.3066 + \frac{0.058}{t_1^{0.688}} + 4 \times \frac{5.7807}{t_2^{0.0784}} + \frac{0.0018}{t_3} + A\left(90.7029t_1^2 + 362.811t_2^2 + 90.7029t_3^2\right)$

$$const(t_1, t_2, t_3) = 3.7499t_1 + 14.944t_2 + 3.722t_3 - 0.0350$$

$$u(t_1, t_2, t_3) = \begin{cases} 0 & if\ const(t_1, t_2, t_3) \le 0 \\ 1 & if\ const(t_1, t_2, t_3) > 0 \end{cases}$$

$$0.0001 \le t_1 \le 0.0120$$
$$0.0001 \le t_2 \le 0.0005$$
$$0.0001 \le t_3 \le 0.0120$$

4 Experimental results

Matlab 7.0 was used to code the NM-PSO method. The initial population was randomly chosen from a uniform distribution of the tolerance values for each component of the over running clutch. The stopping criterion used here is based on the standard deviation of the objective function values over $N+1$ best solutions of the current population, as expressed by

$$S_f = \left[\sum_{i=1}^{N+1} \left(f(x_i) - \overline{f}\right)^2 \Big/ (N+1)\right]^{1/2} < \varepsilon \text{ and } \varepsilon = 1 \times 10^{-7} \tag{15}$$

where $\overline{f} = \sum_{i=1}^{N+1} f(x_i)/(N+1)$ denotes the mean value of the objective function values over $N+1$ best solutions in the current population.

The results of hybrid NM-PSO for solving objective function (14) are shown in Table 2, and Fig. 4 shows a solution history of the results. Table 2 shows us that hybrid NM-PSO locates the global optimum effectively without violating the constraint. NM-PSO is not only effective but also efficient. Figure 4 illustrates NM-PSO converges to the global optimum efficiently in about 50 iterations, and for the algorithm to reach the stopping criterion it takes about 200 iterations or 1,900 number of function evaluations for the various values of A, as shown in Table 2.

Table 3 summarizes the results of applying particle swarm optimization (PSO) [6], genetic algorithm (GA) [5], and geometric programming (GP) [7] to find the optimum

Table 3 The results of PSO, GA, and GP

GP results [7]					
A	t_1	t_2	t_3	COF_{GP}	Constraint
0	0.00507	0.00050	0.00228	11.640	0.0350000
1	0.00477	0.00050	0.00259	11.646	0.0349990
52	0.00509	0.00050	0.00256	11.709*	0.0360873*
100	0.00480	0.00050	0.00256	11.919	0.0349984
300	0.00457	0.00050	0.00279	12.469	0.0349934
520	0.00427	0.00050	0.00261	13.047	0.0331985
GA results [5]					
A	t_1	t_2	t_3	COF_{GA}	Constraint
0	0.00495	0.00050	0.00239	11.640	0.0349296
1	0.00486	0.00050	0.00248	11.647	0.0349271
52	0.00444	0.00050	0.00225	11.789	0.0333252
100	0.00481	0.00050	0.00210	11.923	0.0333252
300	0.00467	0.00050	0.00267	12.470	0.0349218
520	0.00425	0.00050	0.00267	13.047	0.0333468
PSO results [6]					
A	t_1	t_2	t_3	COF_{PSO}	Constraint
0	0.00499	0.00050	0.00237	11.638	0.0349973
1	0.00500	0.00050	0.00245	11.613*	0.0353327*
52	0.00484	0.00050	0.00252	11.785	0.0349976
100	0.00476	0.00050	0.00259	11.920	0.0349831
300	0.00466	0.00050	0.00270	12.468	0.0349974
520	0.00424	0.00050	0.00268	13.047	0.0333496

Note: * the results violate the constraints of the same problem, and it can be seen that some of the results, “marked with an *”, violate the constraints. The comparison of Table 2 and Table 3 clearly indicates that NM-PSO is superior to the other three methods in yielding the optimal machining tolerance allocation of the overrunning clutch assembly. NM-PSO outperforms GP for A=0, 1, 52, 300, surpasses GA for A=0, 1, 52, 100, 300. Finally, the results are compared with PSO and an improvement for A=0, 1, 52, 100 is observed. Note that these results are all within the bounds of the constraints. For the remaining values of A, NM-PSO is as good as the other methods. From the above observations, we conclude that NM-PSO finds the global optimum more effectively and efficiently than the other methods while keeping the constraints.

5 Conclusions

A Matlab program developed by the authors implementing NM-PSO computes the optimal design tolerance for the components of an assembly. Computational efforts and manufacturing costs are both reduced by NM-PSO in comparison to GP, GA, and PSO. These observations lead us to conclude that the proposed hybrid NM-PSO is indeed effective, efficient, and robust at locating best-practice optimum solutions for the clutch assembly problem. In the future, how to apply NM-PSO to solve general optimization problems with multiple constraints would be worth further study.

References

1. Michael W, Siddall JN (1982) The optimal tolerance assignment with less than full acceptance. ASME J Mech Des 104:855–860
2. Kopardekar P, Anand S (1995) Tolerance allocation using neural networks. Int J Adv Manuf Technol 10:269–276
3. Dupinet E, Balazinski M, Czogala E (1996) Tolerance allocation based on fuzzy logic and simulated annealing. J Intell Manuf 7:487–497
4. Ji S, Li X, Ma Y, Cai H (2000) Optimal tolerance allocation based on fuzzy comprehensive evaluation and genetic algorithm. Int J Adv Manuf Technol 16(7):461–468
5. Noorul Haq A, Sivakumar K, Saravanan R, Muthiah V (2005) Tolerance design optimization machine elements using genetic algorithm. Int J Adv Manuf Technol 25:385–391
6. Noorul Haq A, Sivakumar K, Saravanan R, Karthikeyan K (2006) Particle swarm optimization (PSO) algorithm for optimal machining allocation of clutch assembly. Int J Adv Manuf Technol 27:865–869
7. Feng CX, Kusiak A (1997) Robust tolerance design with the integer programming approach. J Manuf Sci Eng: Trans ASME 119:385–391
8. Fan SK, Liang YC, Zahara E (2002) Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. Eng Optim 36:401–418
9. Nelder JA, Mead R (1965) A simplex method for function minimization. Comp J 7:308–313
10. Olsson DM, Nelson LS (1975) The Nelder-Mead simplex procedure for function minimization. Technometrics 17:45–51
11. Kennedy J, Eberhart RC (1995) Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, USA, pp 1942–1948
12. Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, pp 94–97
13. Hu X, Eberhart RC (2001) Tracking dynamic systems with PSO: where’s the cheese? Proceedings of The Workshop on Particle Swarm Optimization, Indianapolis, IN, USA