Int J Adv Manuf Technol (2011) 53:711–732
DOI 10.1007/s00170-010-2871-4

ORIGINAL ARTICLE

Concurrent multi-objective tolerance allocation of mechanical assemblies considering alternative manufacturing process selection

K. Sivakumar · C. Balamurugan · S. Ramabalan

Received: 10 March 2010 / Accepted: 28 July 2010 / Published online: 18 August 2010

Abstract Concurrent design of tolerances by considering both the manufacturing cost and quality loss of each component by alternate processes of the assemblies may ensure the manufacturability, reduce the manufacturing costs, decrease the number of fraction nonconforming (or defective rate), and shorten the production lead time. Most of the current tolerance design research does not consider the quality loss. In this paper, a novel multi-objective optimization method is proposed to enhance the operations of the non-traditional algorithms (Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) and Multi-Objective Particle Swarm Optimization (MOPSO)) and systematically distribute the tolerances among various the components of mechanical assemblies. The problem has a multi-criterion character in which three objective functions, one constraint, and three variables are considered. The average fitness factor method and normalized weighted objective function method are used to select the best optimal solution from Pareto-optimal fronts. Two multi-objective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the strength of Pareto-optimal fronts. Two more multi-objective performance measures namely optimizer overhead and algorithm effort are used to find the computational effort of NSGA-II and MOPSO algorithms. The Pareto-optimal fronts and results obtained from various techniques are compared and analysed. Both NSGA-II and MOPSO algorithms are best for this problem.

Keywords Tolerance design · Alternative manufacturing process selection · Evolutionary algorithms · Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) · Multi-objective Particle Swarm Optimization (MOPSO)

K. Sivakumar
Department of Mechanical Engineering,
Bannari Amman Institute of Technology,
Sathyamangalam-638 401, Erode District, Tamil Nadu, India
e-mail: ksk71@rediffmail.com

C. Balamurugan (✉)
Department of Mechanical Engineering,
M.A.M. College of Engineering,
Tiruchirappalli-621105, Tamil Nadu, India
e-mail: rlc_bal@yahoo.co.in

S. Ramabalan
Department of Mechanical Engineering,
EGS Pillay Engineering College,
Nagapattinam-611002, Tamil Nadu, India
e-mail: cadsrb@gmail.com

1 Introduction

Tolerance design bridges the space between design, manufacturing, and quality engineers, and as such it plays a key role in concurrent engineering. Ideally, a smart technique for tolerance allocation should take into account coupling between design, manufacturing, and quality for achieving total cost and lead-time reduction. However, the same has not been done in existing work. Traditional practice to tolerance design has been based on a sequential approach to the design and manufacturing considerations. Integrated tolerance design involving simultaneous selection of the design and manufacturing tolerances was introduced in the last decade. Choice of manufacturing processes (or machines) from among the alternatives, frequently encountered in different stages of realization of individual dimensions is an important issue in product development. The problem becomes more complicated if the available alternative manufacturing processes (or machines) have non-overlapping precision range. This topic has been the focus of attention of a large number of researchers for decades.

In earlier years, researchers focused their attention to obtain the best tolerance allocation in product design that not only meets with the functional requirements but also corresponds to a minimum manufacturing cost. In order to solve the same, various numerical methods were employed to deal with complicated computations associated with tolerance design models. Ye and Salustri [1] introduced a new concurrent engineering method for tolerance allocation and constructed a non-linear optimization model to implement it. The model minimized quality loss and manufacturing cost simultaneously in a single-objective function by setting both process tolerances and design tolerances. Singh et al. [2, 3] explored the application of genetic algorithms to obtain optimal solution to a set of tolerance design problems with simple-dimension chain involving sets of alternative processes. Prabhaharan et al. [4] used GA for optimal tolerance allocation to overcome the shortcomings in the conventional tolerance stack analysis and allocation system. Prabhaharan et al. [5] introduced a continuous ant colony algorithm, a kind of metaheuristic approach as an optimization tool for minimizing the critical dimension deviation and allocating the cost-based optimal tolerances. Krishna and Mallikarjuna Rao [6] used scatter search method to simultaneously allocate both design and manufacturing tolerances based on minimum total manufacturing cost. Huang and Shiau [7] obtained the optimized tolerance allocation of a sliding vane rotary compressor's components for the required reliability with the minimum cost and quality loss. Huang and Zhong [8] established the sequential linear optimization models based on the process capabilities. This approach can release the working tolerances, reduce manufacturing costs, and enhance the acceptance rate of machined parts. Singh et al. [9] introduced GA to obtain an optimal solution to the advanced tolerance synthesis problem by considering continuous cost function. This method works for both single and multiple tolerance stack-ups that share one or more individual tolerances. Siva Kumar et al. [10] used a hybrid algorithm (Tabu search+Heuristic algorithm) for optimum tolerance allocation in complex assemblies with alternative process selection. Isabel González et al. [11] developed a methodology to allow an automatic tolerance allocation capable of minimizing manufacturing costs based on statistical approach. Huang and Zhong [12] used linear programming methods to concurrently obtain process tolerances for an assembly by using the information of process planning. Non-linear optimal models have been established to minimize the total weight manufacturing cost. Forouraghi [13] developed a methodology to allow an automatic tolerance allocation capable of minimizing manufacturing costs based on particle swarm optimizer. Siva Kumar and Stalin [14] used Lagrange multiplier method to simultaneously allocate both design and manufacturing tolerances based on minimum total manufacturing cost. Wu et al. [15] developed a methodology to allow an automatic tolerance allocation capable of both minimizing manufacturing costs and quality loss based on Monte Carlo simulation. Muthu et al. [16] used metaheuristic method to balance the manufacturing cost and quality loss to achieve near optimal design and process tolerances simultaneously for minimum combined manufacturing cost and quality loss over the life of the product.

From the above literature, the previous work on tolerance allocation can be summarized as follows:

(1) Scatter search method, sequential linear method, Lagrange multiplier method, Monte Carlo simulation, Linear programming methods, etc. were used to solve the tolerance allocation problem, (2) a lot of research has been carried out on minimization of manufacturing cost without considering quality loss of the product, (3) in some research work, total manufacturing cost and quality loss analysis were performed using only traditional methods, however, these approaches suffer from the following drawbacks: (1) closed form solution is limited to only a small portion of the spectrum of the real-world tolerancing problems. It is applicable to the objective functions involving only simple cost functions, such as 'reciprocal power function' (reciprocal, and reciprocal square functions are the special cases of this function) and 'exponential function', subjected to only traditional tolerance stack-up constraints (based on the worst case and the root sum square criteria), (2) it is not applicable to discontinuous and/or non-differentiable cost functions as it requires a continuous first derivative, (3) cost functions with preferred process limits cannot be handled by these method, (4) assemblies involving interrelated dimension chains (more than one assembly response functions with some common dimensions) are difficult to handle by these methods, (5) these methods can be attracted by local minima. In order to overcome this problem, the process should be started from various initial guesses. Also, the conventional techniques involve the computation of the gradient and the Hessian of objective function and constraints, which imply the continuity of second order must be ensured. (6) Multi-objective optimization deals with generating the Pareto front which is the set of non-dominated solutions for problems having more than one objective. A solution is said to be non-dominated if it is impossible to improve one component of the solution without worsening the value of at least one other component of the solution.

To overcome the drawbacks of the conventional optimization approaches, non-traditional optimization techniques such as GA, simulated annealing [4, 17–20], NSGA-II and MOPSO can be used. Non-traditional algorithms have proven successful in handling many real-world multi-objective concurrent engineering problems [21–24]. The advantages of non-traditional techniques are (1) they are a population-based search techniques, so global optimal solution is possible, (2) they do not need any auxiliary information like gradients, derivatives, etc. (3) They are easier to program and implement than the methods reported in the literature. (4)

They can solve complex and multimodal problems for global optimality. (5) They are problem-independent, i.e., suitable for solving all types of problems. (6) They are computationally superior and faster than the methods reported in the literature. (7) They offer Pareto-optimal fronts that offer more number of optimal solutions for user's choice. (8) They use average fitness function method to find best optimal solution from the Pareto-optimal fronts. (9) They guide the search toward the true Pareto front (non-dominated solutions) or approximate the Pareto-optimal set. (10) They generate a well-distributed Pareto front and (11) they are population-based algorithms which allow them to explore the different parts of the Pareto front simultaneously.

The major limitations of the tolerance design works of Singh et al. [25] is the important objective functions such as quality loss and stack-up tolerance are not considered. To overcome the limitations of Singh et al. [25] work, two evolutionary optimisation techniques (NSGA-II and MOPSO) are proposed in this paper to do optimal tolerance design for mechanical assemblies by considering all objective functions (minimization of manufacturing cost, quality loss and tolerance stack-up), yield design constraint and machining tolerance constraints. The average fitness factor method and weighted objective functions method used to select the best optimal solution from Pareto-optimal fronts (optimal solution trade-offs). Two multi-objective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the strength of Pareto-optimal fronts. Two more multi-objective performance measures namely optimizer overhead and algorithm effort are used to find the computational effort of NSGA-II and MOPSO algorithms.

The significant contributions of our paper to tolerance area are: (1) it proposes a general purpose approach to do alternative manufacturing process selection and tolerance allocation. (2) This paper simultaneously considers the minimization of tolerance stack-up, manufacturing cost and the quality loss as objective functions. (3) It improves the recent optimization model proposed by Singh et al. [25] by adding two new objective functions (quality loss and tolerance stack-up), and (4) the proposed approach using NSGA-II and MOPSO solves all drawbacks of the methods reported in literature [1–29].

The rest of this study is organized as follows: Section 2 deal the problem definition. Section 3 presents the problem formulation. Section 4 presents the proposed NSGA-II and MOPSO techniques to obtain the optimal solutions. Section 5 deals two methods and four multi-objective performance metrics used for evaluating the proposed algorithms. Section 6 illustrates numerical examples. Running procedure for NSGA-II and MOPSO algorithms are given in Section 6.1. In Section 7, the results obtained in various methods are presented and compared. The conclusions are presented in the last section.

2 Problem definition

2.1 Stock removal allowances

Singh et al. [25] addressed the stock removal allowances, Manufacturing tolerance selection is greatly affected by the amount of stock removal allowance. The stock removal allowance is the layer of material to be removed from the surface of a work-piece to obtain the required accuracy and surface quality through machining. It greatly influences the quality and the production efficiency of the machined features. Excessive stock removal allowance will increase the consumption of material, machining time, tool and power, while machining the work-piece by a more expensive downstream process, resulting in an increase in the manufacturing cost. On the other hand, with an insufficient stock removal allowance, the defective surface layer caused by the preceding operation cannot be rectified. The concept of stock removal allowance has been depicted in Fig. 1.

The amount of stock removal allowance is the difference between the dimensions obtained in the preceding operation and the current operation. Since the manufacturing dimensions are not fixed, and each of them is associated with some tolerance, the actual stock removals from work-piece surfaces vary in a certain range. The stock removal based on the nominal working dimensions in successive operations is considered as its nominal value. Variation in the stock removal is the sum of manufacturing tolerances in the preceding and the current operation. An appropriate stock removal allowance is required for each successful manufacturing operation. This yields a set of necessary constraints during simultaneous selection of design and manufacturing tolerances.

2.2 Selection of machining process

The selection of machining process including equipment accuracy, set-up mode, machining sequence and cutting parameters is strongly affected by the tolerance of the part to

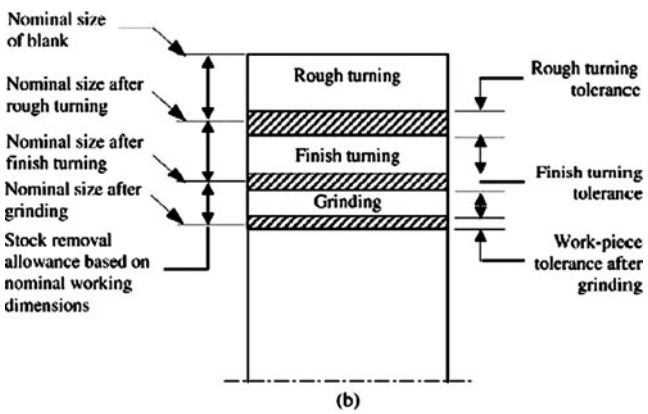

Fig. 1 Machined layers and manufacturing tolerances [25]

be machined. So it is important to do the simultaneous selection of the best machining process while allocating the tolerance.

2.3 Manufacturing cost

Manufacturing cost usually increases with the tight tolerance of quality characteristics, as more refined and precise operations are needed and acceptable ranges of output are reduced. Conversely, large tolerances are less costly to achieve as they require less precise manufacturing processes; but they usually result in poor performance, premature wear and part rejection.

The objective function of the tolerance synthesis problem with the exponential cost function. The following tolerance-cost function is used to find manufacturing cost of a process tolerance.

$$C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{1}$$

2.4 Quality loss function

Variability in the production process is unavoidable due to inconsistency in tool work-piece, material and process parameters. Noorul Haq et al. [17] suggests that given an ideal target value, an evaluation function associated with deviations from the target value can be developed. In this study, it is referred to as the quality loss function. This loss function is a quadratic expression for measuring the cost of the average value versus the target value and the variability of product characteristics in terms of monetary loss due to product failure in the eyes of the consumers. The quality loss function (Q_{L}) is

$$Q_{\text{L}} = \frac{A}{T^2} \sum_{i=1}^{I} {\sigma_i}^2 \tag{2}$$

For a three sigma tolerance design,

$$\sigma_i = \frac{t_i}{3}$$

Then, Eq. 2 is rewritten as

$$Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} {t_i}^2 \tag{3}$$

Where, T is the tolerance stack-up limit of the dimensional chain, A is the quality loss cost.

3 Problem statements

The improved optimization model opens up possibilities for tolerance control in order to achieve selection of best manufacturing processes, economical assembly manufacturing cost and the minimum quality loss of the product. Formulation of the simple optimal tolerance synthesis involves framing of the objective functions and constraints. The objective functions considered are: minimum tolerance stack-up (Z_1), minimum total manufacturing cost of the assembly (Z_2; summation of the manufacturing cost involved in the manufacturing of all the toleranced dimensions) and minimum quality loss function (Z_3). The functional requirements of the assembly (stack-up conditions), process precision limits and process selection conditions form the constraints of the problem. The variables are tolerances of all part dimensions. Multi-criterion optimization methods used in this paper are as follows:

3.1 Objective functions

3.1.1 Example A (Assembly A)

$$\text{Minimize} : Z_1 = \Delta Y = t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} \tag{4}$$

$$Z_2 = C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{5}$$

$$Z_3 = Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} {t_{ij}}^2 \tag{6}$$

3.1.2 Example B (Assembly B)

$$\text{Minimize} : Z_1 = \Delta Y = t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} + t_{6\text{j}} \tag{7}$$

$$Z_2 = C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{8}$$

$$Z_3 = Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} {t_{ij}}^2 \tag{9}$$

Assembly function:

$$Y = X_1 + X_2 + X_3 + X_4 + X_5 + X_6 \tag{10}$$

Stack-up condition:

$$t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} + t_{6\text{j}} \leq 0.01 \tag{11}$$

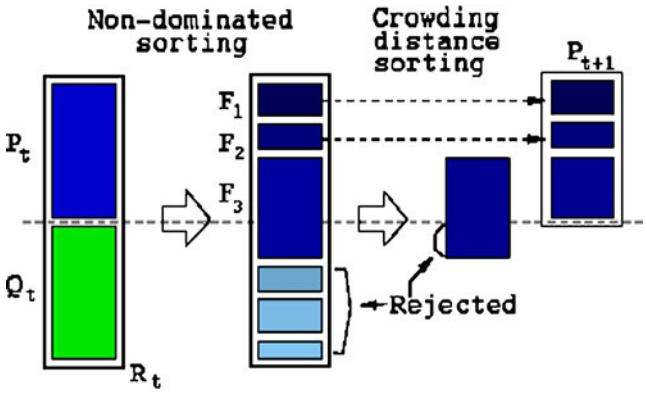

Fig. 2 An iteration procedure of the NSGA-II algorithm

3.1.3 Example C (gearbox assembly)

$$\text{Minimize}: Z_1 = \Delta Y = t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} \tag{12}$$

$$Z_2 = C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{13}$$

$$Z_3 = Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} t_{1j}^2 + t_{2j}^2 + t_{3j}^2 + 2\left(t_{4j}^2\right) \tag{14}$$

Assembly function:

$$Y = X_1 + X_2 - X_3 - X_4 - X_5 \tag{15}$$

Stack-up condition:

$$t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} \leq 0.26 \tag{16}$$

Set-up reduction condition(s):

$t_{1\text{j}} = t_{2\text{j}}$ and $t_{4\text{j}} = t_{5\text{j}}$

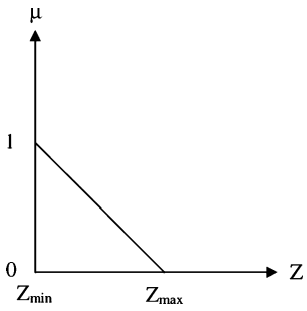

Fig. 3 Representation of fitness factor for minimization of an objective function

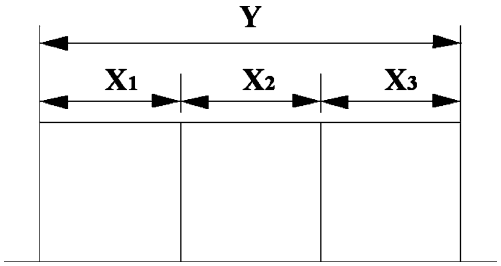

Fig. 4 Assembly A

3.1.4 Example D (Shaft and housing assembly)

$$\text{Minimize}: Z_1 = \Delta Y = t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} + t_{6\text{j}} + t_{7\text{j}} \tag{17}$$

$$Z_2 = C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{18}$$

$$Z_3 = Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} t_{ij}^2 \tag{19}$$

Assembly function:

$$Y = -X_1 + X_2 - X_3 + X_4 - X_5 + X_6 - X_7 \tag{20}$$

Stack-up condition:

$$t_{1\text{j}} + t_{2\text{j}} + t_{3\text{j}} + t_{4\text{j}} + t_{5\text{j}} + t_{6\text{j}} + t_{7\text{j}} \leq 0.3831 \tag{21}$$

Set-up reduction condition:

$t_{4\text{j}} = t_{6\text{j}}$

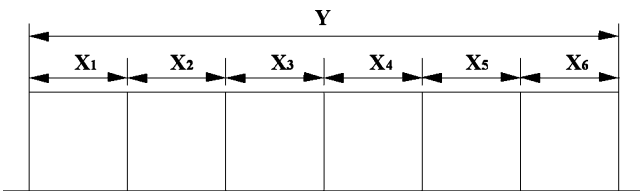

Fig. 5 Assembly B

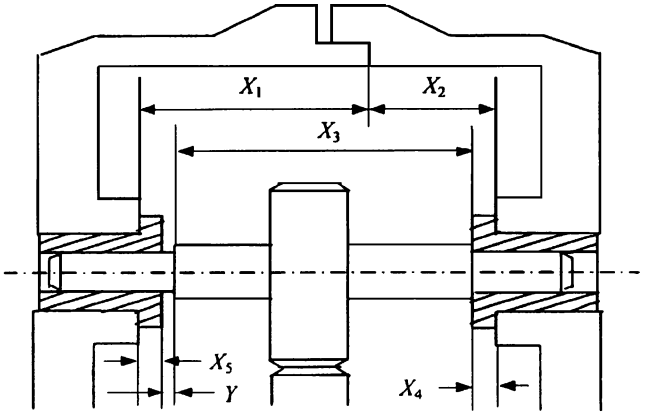

Fig. 6 Gearbox assembly [25]

3.1.5 Example E (one-way clutch assembly)

$$\text{Minimize}: Z_1 = \Delta Y = \left|\frac{\partial Y}{\partial X_1}\right| t_{1j} + \left|\frac{\partial Y}{\partial X_2}\right| t_{2j} + \left|\frac{\partial Y}{\partial X_3}\right| t_{3j} \tag{22}$$

$$Z_2 = C_{\text{asm}} = \sum_{i=0}^{n} \left(a_{ix_i} + b_{ix_i} e^{-c_{ix_i} t_{ix_i}}\right) \tag{23}$$

$$Z_3 = Q_{\text{L}} = \frac{A}{9T^2} \sum_{i=1}^{I} {t_{ij}}^2 \tag{24}$$

Assembly function:

$$Y = a\cos[(X_1 + X_2)/(X_3 - X_2)] \tag{25}$$

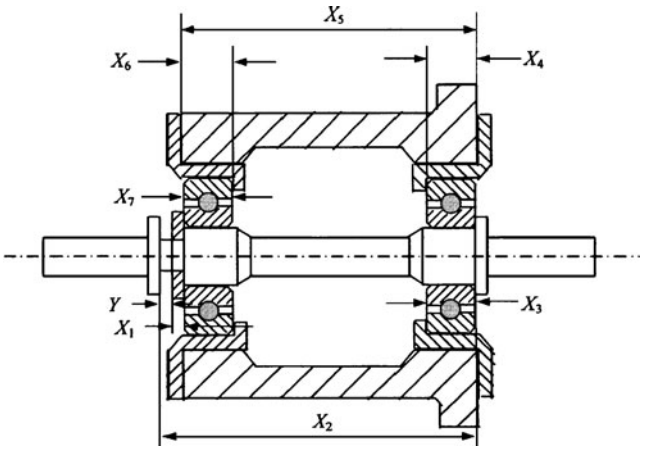

Fig. 7 Shaft and housing assembly [25]

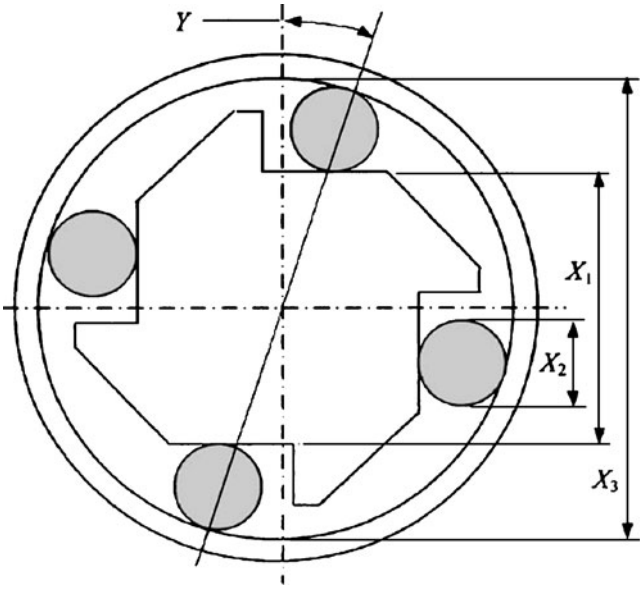

Fig. 8 One-way clutch assembly [25]

Stack-up condition:

$$\left|\frac{\partial Y}{\partial X_1}\right| t_{1\text{j}} + \left|\frac{\partial Y}{\partial X_2}\right| t_{2\text{j}} + \left|\frac{\partial Y}{\partial X_3}\right| t_{3\text{j}} \le 0.0349 \tag{26}$$

which further reduces to

$$\sum_{i=1}^{n} t_{ij} \le T_{k_{\text{asm}}} \tag{27}$$

For a linear assembly

Table 1 Assembly A details

Dimensions	Process	Parameters of cost function		
		a	*b*	*c*
X_1	1	5.0	34.2245	765
	2	4.7	39.9819	782
	3	4.36	45.0974	790
	4*	500	41.5370	777
X_2	1	6.05	53.1921	975
	2	5.62	60.0065	995
	3	5.29	149.5845	986
	4*	500	63.6541	981
X_3	1	5.38	72.6260	1386
	2	5.31	96.5270	1412
	3	5.22	82.8130	1400
	4*	500	87.4149	1408

Table 2 Assembly B details

Dimensions	Process	Parameters of cost function		
		a	*b*	*c*
X_1,X_4	1	5.0	34.2245	765
	2	4.7	39.9819	782
	3	4.36	45.0974	790
	4*	500	41.5370	777
X_2,X_5	1	6.05	53.1921	975
	2	5.62	60.0065	995
	3	5.29	149.5845	986
	4*	500	63.6541	981
X_3,X_6	1	5.38	72.6260	1386
	2	5.31	96.5270	1412
	3	5.22	82.8130	1400
	4*	500	87.4149	1408

3.2 Variables

Process precision limits : $t_{ij}^{\min} \le t_{ij} \le t_{ij}^{\max}$

$i = 1$ to n, $j = 1$ to m_i

Where,

Y	Assembly response function
C_{asm}	Total assembly manufacturing cost
Q_{L}	Quality loss function
C_{ij} (t_{ij})	Cost of producing dimension x_{i} by the process j maintaining tolerance t_{ij}
m_i	Number of the available alternative processes for producing dimension x_{i}
n	Number of the component dimensions
$T_{k_{\text{asm}}}$	Permissible variation in the kth assembly dimension, known as assembly tolerance,
t_{ij}	Tolerance on the dimension x_i produced by the jth process.

4 Proposed methods

In this section, the proposed non-traditional optimization techniques such as NSGA-II and MOPSO are described.

4.1 Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

Kalyanmoy Deb proposed the NSGA-II algorithm [26]. Essentially, NSGA-II differs from non-dominated sorting Genetic Algorithm (NSGA) implementation in a number of ways. Firstly, NSGA-II uses an elite-preserving mechanism, thereby assuring preservation of previously found good solutions. Secondly, NSGA-II uses a fast non-dominated sorting procedure. Thirdly, NSGA-II does not require any tunable parameter, thereby making the algorithm independent of the user.

Initially, a random parent population P_{o} is created. The population is sorted based on the non-domination. A special book-keeping procedure is used in order to reduce the computational complexity to $O(MN^2)$. Each solution is assigned a fitness equal to its non-dominated level (1 is the best level). Thus, minimization of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_{o} of size N. thereafter; the algorithm below is used in every generation.

$R_{\text{t}}=P_tUQ_t$
F=fast non-dominated-sort (R_t)
$P_{t+1}=\phi$ and $i=1$
Until $|P_{t+1}|+|F_i|\le N$
$P_{t+1}=P_{t+1}\text{U}\ F_i$
crowding-distance-assignment (F_i)
$i=i+1$
Sort($F_i \propto_n$)
$P_{t+1} = P_{t+1}UP_{t+1}[1 : (N - |P_{t+1}|)]$
$Q_{t+1} = \text{make}-\text{new}-\text{pop}(\text{P}_{\text{t+1}})$
$t = t + 1$

First, a combined population $R_t=P_tUQ_t$ is formed. This allows parent solutions to be compared with the child population, thereby ensuring elitism. The popula-

Table 3 Assembly C: gear box assembly details

Dimensions	Process	Parameters of cost function		
		a	*b*	*c*
X_1, X_2	1	18.50	71.25	214.56
	2	20.82	68.44	208.68
	3	19.05	69.32	211.05
	4	18.32	73.56	220.73
X_3	1	42.50	30.254	82.566
	2	39.20	33.443	86.688
	3	38.05	34.322	79.005
	4*	539.32	37.061	78.732
X_4, X_5	1	32.50	28.25	82.45
	2	29.20	30.43	86.70
	3	28.05	31.42	80.05
	4	29.32	34.16	78.82

Table 4 Assembly D: shaft and housing assembly details

Dimensions	Process	Parameters of cost function		
		a	*b*	*c*
X_1	Vendor supplied (fixed tolerance t_1=0.0381; C_1=5.00)			
X_2	1	5.34	66.43	2.738
	2	5.12	62.22	2.340
X_3,X_7	Vendor supplied (fixed tolerance t_3=0.0635; C_3=50.00)			
X_4,X_6	1	15.34	69.43	2.728
	2	15.12	65.22	2.340
	3	14.85	66.87	2.112
	4*	500	70.62	2.985
X_5	1	11.34	72.43	2.738
	2	11.12	68.22	2.340
	3	10.85	69.87	2.112
	4*	500	73.62	2.985

tion R_t is of size $2N$. Then, the population R_t is sorted according to non-domination and non-dominated fronts F_1, F_2, and so on are found. The algorithm is illustrated in the following:

The new parent population P_{t+1} is formed by adding solutions from the first front F_1 and continuing to other fronts successively till the size exceeds N. Individuals of each front are used to calculate the crowding distance—the distance between the neighboring solutions. Thereafter, the solutions of the last accepted front are sorted according to a crowded comparison criterion and a total of N points are picked. Since the diversity among the solutions is important, the crowded comparison criterion uses a relation α_n as follows: solution i is better than solution j in relation α_n if ($i_{\text{rank}}<j_{\text{rank}}$) or (($i_{\text{rank}}=j_{\text{rank}}$) and ($i_{\text{distance}}>j_{\text{distance}}$)). That is, between two solutions with differing non-domination ranks the preference is the point with the lower rank. Otherwise, if both the points belong to the same front, then the preference is the point which is located in a region with smaller number of points (or with larger crowded distance). This way, solutions from less dense regions in the search space are given importance in deciding which solutions to choose from R_t. This constructs the population P_{t+1}. This population of size N is now used for selection, crossover and mutation to create a new population Q_{t+1} of size N. A binary tournament selection operator is used but the selection criterion is now based on the crowded comparison operator α_n. The above procedure is continued for a specified number of generations. It is clear from the above description that NSGA-II uses (1) a faster non-dominated sorting approach, (2) an elitist strategy, and no niching parameter. It has been proved that the above procedure has $O(MN^2)$ computational complexity. Figure 2 shows an iteration of the proposed NSGA-II procedure.

4.2 MOPSO

The proposed algorithm which we shall call MOPSO extends the single-objective particle swarm optimization (PSO) algorithm to handle the multi-objective optimization problems. It incorporates the mechanism of the

Table 5 Assembly E: one-way clutch assembly details

Dimensions	Process	Parameters of cost function		
		a	*b*	*c*
X_1(hub)=55.291	1	15.05	71.24	4.348
	2	15.12	70.10	4.015
	3	16.54	68.32	4.478
	4	18.92	66.98	5.552
X_2(roll)=22.86	Vendor supplied (fixed tolerance t_2=0.0635; C_2=30.00)			
X_3(cage)=101.6	1	21.55	98.87	18.39
	2	21.02	98.12	17.81
	3	23.14	99.78	18.94
	4	500	101.10	19.24

Table 6 Optimization results for Assembly A

Techniques	Dimensions	Process	Tolerance	Objective function		
				Z_1	Z_2	Z_3
NSGA-II	X_1	1	0.001500	0.005	40.856503	0.042082
	X_2	1	0.001900			
	X_3	1	0.001900			
MOPSO	X_1	1	0.001424	0.005	43.3628	0.0374
	X_2	1	0.001828			
	X_3	1	0.001745			

crowding distance computation into the algorithm of PSO specifically on global best selection and in the deletion method of an external archive of non-dominated solutions. The crowding distance mechanism together with a mutation operator maintains the diversity of non-dominated solutions in the external archive. MOPSO also has a constraint handling mechanism for solving constrained optimization problems.

The most striking difference between MOPSO and the other evolutionary algorithms is that MOPSO chooses the path of cooperation over competition. The other optimization algorithms commonly use some form of decimation, survival of the fittest. In contrast, the MOPSO population is stable and individuals are not destroyed or recreated. Individuals are influenced by the best performance of their neighbors. Individuals eventually converge on optimal points in the problem domain. In addition, the MOPSO traditionally does not have genetic operators like crossover between individuals and mutation, and other individuals never substitute particles during the run. So, in MOPSO, all the particles tend to converge to the best solution quickly, as compared to the other optimization algorithms. Pseudo Code for MOPSO is explained in the Appendix section A.

The main advantages of MOPSO method are:

1. It works simultaneously with a set of possible solutions, the so-called population, and several non-dominated solutions maybe found in a single run of the algorithm.
2. It gives optimal solution trade-offs with more number of non-dominated solutions for user's choice than other algorithms.
3. It does not require prior knowledge of the relative importance of the objectives.
4. There is a set of acceptable trade-off near optimal solutions. This set is called Pareto front or optimality trade-off surfaces.
5. The algorithm handles constraints in a very simple and efficient way, as comparing to different solutions.
6. It is less sensitive to the shape or continuity of the Pareto surface.

4.3 NSGA-II operators

The following are the values of the parameters of NSGA-II technique used in this study:

Variable type=real variable, population size=100, crossover probability=0.6, real-parameter mutation probability=

Table 7 Optimization results for Assembly B

Techniques	Dimensions	Process	Tolerance	Objective function		
				Z_1	Z_2	Z_3
NSGA-II	X_1	1	0.0015	0.01	81.707703	0.021044
	X_2	1	0.0019			
	X_3	1	0.0019			
	X_4	1	0.0015			
	X_5	1	0.0019			
	X_6	1	0.0019			
MOPSO	X_1	1	0.001483	0.01	86.8368	0.0186
	X_2	1	0.001776			
	X_3	1	0.001734			
	X_4	1	0.001483			
	X_5	1	0.001776			

Table 8 Optimization results for Assembly C (gear box assembly)

Techniques	Dimensions	Process	Tolerance	Objective function		
				Z_1	Z_2	Z_3
NSGA-II	X_1, X_2	4	0.015138	0.085751	159.019974	0.002518
	X_3	3	0.024135			
	X_4, X_5	3	0.01567			
MOPSO	X_1, X_2	4	0.014416	0.0863	158.5301	0.0025
	X_3	3	0.024018			
	X_4, X_5	3	0.016713			

0.01, real-parameter SBX parameter=10, real-parameter mutation parameter=100, total number of generations=100.

4.4 MOPSO operators

The following are the values of the parameters of MOPSO technique that have been used to obtain the best optimal results: population size=100, mutation probability=0.5, total number of generations=100, inertia weight=0.4

5 Performance measures and methods for multi-objective optimisation

In this section, two methods and four performance metrics are recommended and applied to examine the strength and weaknesses of the proposed multi-objective evolutionary algorithms. Two methods (normalized weighting objective functions and average fitness factor) are used to select best optimal solution. Two multi-objective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the Pareto-optimal fronts. Two more multi-objective performance measures namely optimizer overhead and algorithm effort are used to find computational effort of an optimisation algorithm. These methods and metrics are chosen since they have been widely used for performance comparisons in multi-objective optimization [28].

5.1 Normalized weighting objective functions

Multiple objectives are combined into a scalar objective via weight vector. Weights may be assigned through direct assignment, eigenvector method, empty method, minimal information method, randomly determined or adaptively determined. If the objective functions are simply weighted and added to produce a single fitness, the function with largest range would dominate evolution. A poor input value for the objective with larger range makes the overall value much worse than a poor value for the objective with smaller range. To avoid this, all objective functions are normalized to have same range. Also normalizing parameters make all objective functions as unitless functions. For our problem, the combined objective function (f_c) is defined as follows:

Example A (Assembly A)

$$\text{Minimize } :f_c = W_1 \times Z_1/N_1 + W_2 \times Z_2/N_2 + W_3 \times Z_3/N_3 \quad (28)$$

Table 9 Optimization results for Assembly D (shaft and housing assembly)

Techniques	Dimensions	Process	Tolerance	Objective function		
				Z_1	Z_2	Z_3
NSGA-II	X_1	Vendor supplied (fixed tolerance)	0.0381	0.291699	393.666779	0.010314
	X_2	2	0.026594			
	X_3,X_7	Vendor supplied (fixed tolerance)	0.0635			
	X_4,X_6	2	0.03			
	X_5	2	0.040005			
MOPSO	X_1	Vendor supplied (fixed tolerance)	0.0381	0.2894	393.9548	0.0103
	X_2	2	0.020013			
	X_3,X_7	Vendor supplied (fixed tolerance)	0.0635			
	X_4,X_6	2	0.030171			
	X_5	2	0.043949			

Table 10 Optimization results for Assembly E (one-way clutch assembly)

Techniques	Dimensions	Process	Tolerance	Objective function		
				Z_1	Z_2	Z_3
NSGA-II	X_1	3	0.00254	0.003748	228.885376	0.010669
	X_2	Vendor supplied (fixed tolerance)	0.01020			
	X_3	2	0.002541			
MOPSO	X_1	3	0.002567	0.0038	228.1188	0.0109
	X_2	Vendor supplied (fixed tolerance)	0.010200			
	X_3	2	0.002646			

Example B (Assembly B)

$$\text{Minimize } :f_c = W_1 \times Z_1/N_1 + W_2 \times Z_2/N_2 + W_3 \times Z_3/N_3 \quad (29)$$

Example C (gearbox assembly)

$$\text{Minimize } :f_c = W_1 \times Z_1/N_1 + W_2 \times Z_2/N_2 + W_3 \times Z_3/N_3 \quad (30)$$

Table 11 Results obtained from NSGA-II and MOPSO algorithms

Techniques	Z_1	Z_2	Z_3	μ_{avg}
Assembly A				
Zmax	0.005	46.111019	0.042082	
Zmin	0.0047	40.856503	0.033202	
NSGA-II	0.0053	40.8565	0.04208	0
MOPSO	0.005	43.6737	0.0369	0.349137
Assembly B				
Zmax	0.01	92.225082	0.021044	
Zmin	0.0094	81.707703	0.0166	
NSGA-II	0.0106	81.7077	0.02104	0
MOPSO	0.01	87.0292	0.0186	0.347994
Assembly C				
Zmax	0.299976	170.50705	0.031356	
Zmin	0.072003	131.117508	0.001920	
NSGA-II	0.29998	131.118	0.03136	0.333333
MOPSO	0.2507	131.594	0.0227	0.499371
Assembly D				
Zmax	0.383566	395.982391	0.016858	
Zmin	0.2751	380.940033	0.009854	
NSGA-II	0.38357	380.94	0.01686	0.333333
MOPSO	0.3772	381.797	0.0165	0.350938
Assembly E				
Zmax	0.0039	228.887527	0.011133	
Zmin	0.003748	227.9932	0.010668	
NSGA-II	0.00388	227.983	0.01113	0.374279
MOPSO	0.0039	227.994	0.0111	0.356542

Example D (shaft and housing assembly)

$$\text{Minimize } :f_c = W_1 \times Z_1/N_1 + W_2 \times Z_2/N_2 + W_3 \times Z_3/N_3 \quad (31)$$

Example E (one-way clutch assembly)

$$\text{Minimize } :f_c = W_1 \times Z_1/N_1 + W_2 \times Z_2/N_2 + W_3 \times Z_3/N_3 \quad (32)$$

W_1, W_2, and W_3 are weightages given to objective functions 1, 2, and 3, respectively. Here, normalized weighting objective functions method is used only to select the best optimal solution from Pareto-optimal fronts obtained from NSGA-II and MOPSO. So we can give any weightage to each objective function. But the condition is $W_1 + W_2 + W_3 = 1$. It means the total weightage should be 100%. The value of W_1=W_2=W_3=0.333 (for Assemblies A, B, C, D, E). It means we are giving equal weightage to all objective functions.

N_1, N_2, and N_3 are normalizing parameters of objective functions (Average value of individual objective functions). The values of N_1=0.01, N_2=10, and N_3=0.10 (for Assembly A), the values of N_1=0.10, N_2=10, and N_3=0.10 (for Assembly B), the values of N_1=1.0, N_2=100, and N_3=0.01 (for Assembly C), the values of N_1=1.0, N_2=100, and N_3=0.10 (for Assembly D), the values of N_1=0.01, N_2=100, and N_3=0.10 (for Assembly E).

For example A, the original values of first, second and third objective functions from NSGA-II are 0.005, 40.856503, and 0.042082, respectively. So to bring all objective functions to have the same range, the first second and third objective functions are divided by their individual average values 0.01, 10, and 0.10, respectively. Now the normalized values of first, second, and third objective functions are 0.50, 4.0856503, and 0.42082.

Table 12 Results obtained from NSGA-II and MOPSO algorithms

Techniques	Z_1	Z_2	Z_3	Combined objective function (Fc)
Assembly A				
NSGA-II	0.005	40.856503	0.042082	1.677145
MOPSO	0.005	43.3628	0.0374	1.735023
Assembly B				
NSGA-II	0.01	81.707703	0.021044	2.826241
MOPSO	0.01	86.8368	0.0186	2.986903
Assembly C				
NSGA-II	0.085751	159.019974	0.002518	0.641941
MOPSO	0.0863	158.5301	0.0025	0.639893
Assembly D				
NSGA-II	0.291699	393.666779	0.010314	1.442392
MOPSO	0.2894	393.9548	0.0103	1.442539
Assembly E				
NSGA-II	0.003748	228.885376	0.010669	0.922524
MOPSO	0.0038	228.1188	0.0109	0.922473

5.2 Average fitness factor (F_{avg})

The deterministic models proposed in the literature suffer from real-world optimal tolerance allocation limitation because that a decision maker does not have sufficient information related to the different criteria. So he may not know what weightage is to be given to each objection. In that situation, he may use the average fitness factor method proposed in this paper.

The average fitness factor given in Fig. 3 is a graphical representation of the magnitude of each input. The F value is 1 at Z_{min} and 0 at Z_{max} for minimizing an objective function and vice versa for maximizing an objective function.

The proposed fitness factor is as follows:

$F_i = (Z_{i\max} - Z_i)/(Z_{i\max} - Z_{i\min})$ for minimization of objective function

Z_i objective function (i=objective function number, 1… 2 for this problem)

Z_{max} maximum objective function value

Z_{min} minimum objective function value

The solution that has the highest average membership function value (μ_{avg}) is the best optimal solution that gives a non-dominated solution. For our problem, the average fitness factor is defined as follows:

Table 13 Algorithm effort obtained from NSGA-II and MOPSO algorithms

Proposed algorithm	Simulation time T_{run} (sec)	No. of function evolution (N_{eval})	Algorithm effort
Assembly A			
NSGA-II	2	45	0.0444
MOPSO	2	77	0.0259
Assembly B			
NSGA-II	2	48	0.0444
MOPSO	2	79	0.0253
Assembly C			
NSGA-II	2	47	0.0444
MOPSO	2	81	0.0247
Assembly D			
NSGA-II	2	39	0.0513
MOPSO	2	71	0.0282
Assembly E			
NSGA-II	2	33	0.0606
MOPSO	2	65	0.0308

Table 14 SSM, RNI, and OO obtained from NSGA-II and MODE algorithms

Proposed Algorithm	SSM	RNI	OO	Computation time (sec)
Assembly A				
NSGA-II	0.553736	1	0.2727	2.5
MOPSO	0.533614	1	0.0889	1.2
Assembly B				
NSGA-II	0.674728	1	0.1500	2.5
MOPSO	0.640237	1	0.1042	1.2
Assembly C				
NSGA-II	0.742259	1	0.1842	2.5
MOPSO	0.92361	1	0.0588	1.2
Assembly D				
NSGA-II	0.560891	1	0.1935	2.5
MOPSO	0.544037	1	0.04545	1.2
Assembly E				
NSGA-II	0.535718	1	0.1707	2.5
MOPSO	0.636052	1	0.0980	1.2

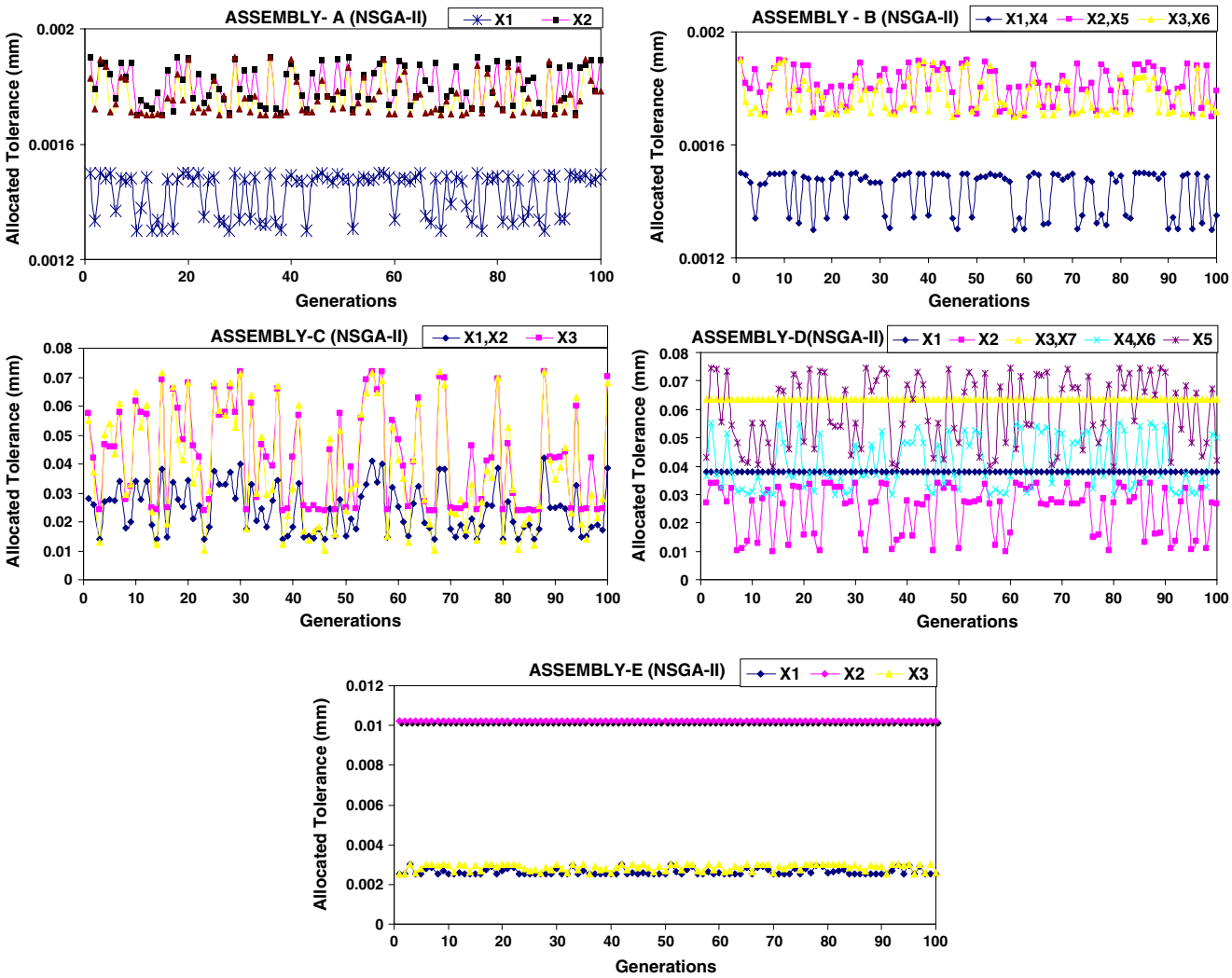

Fig. 9 Tolerance allocation for optimal processes combination of Assemblies A to E from NSGA-II

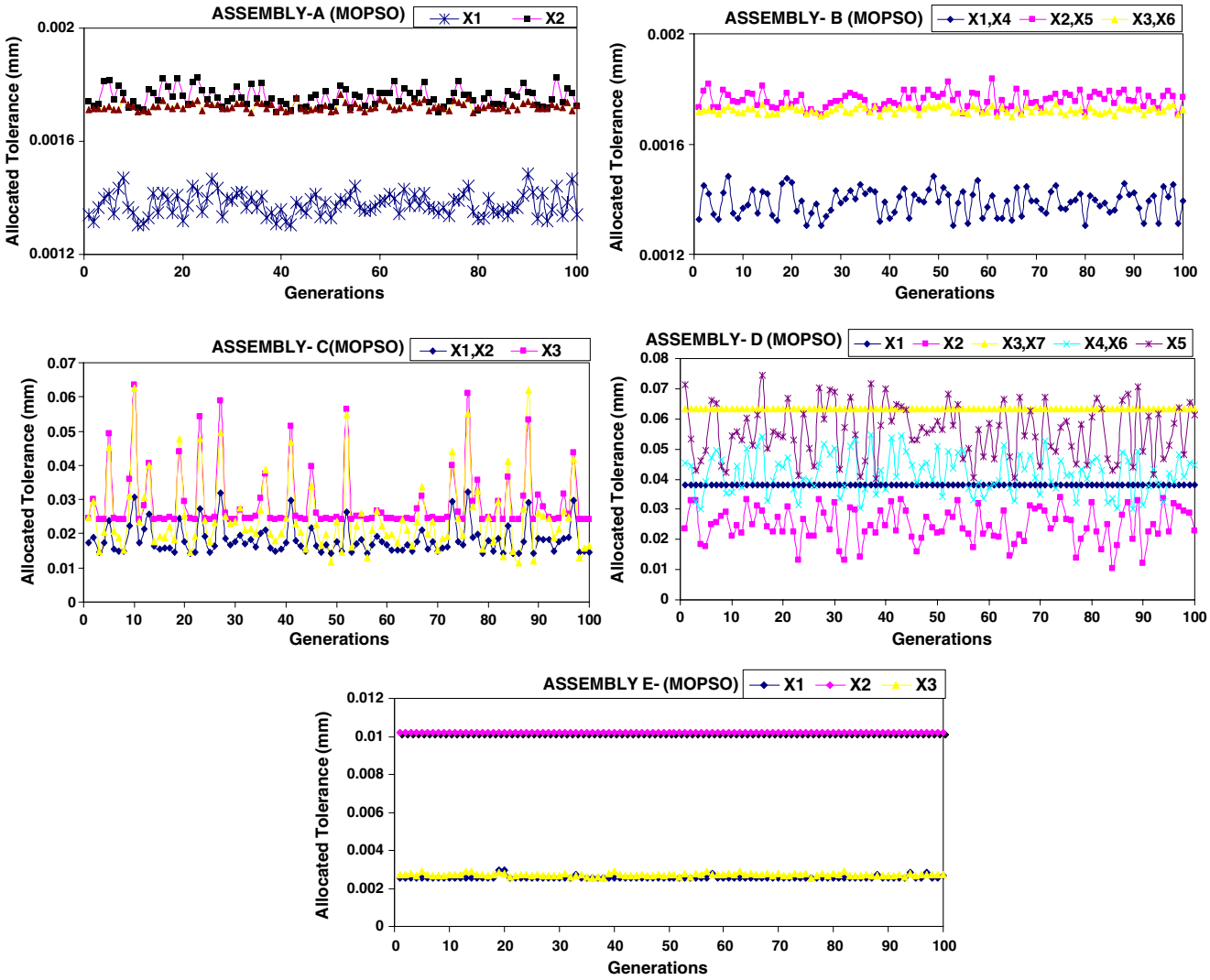

Fig. 10 Tolerance allocation for optimal processes combination of Assemblies A to E from MOPSO

Maximize average fitness factor

$$F_{\text{avg}} = (F_1 + F_2 + F_3)/3.0 \tag{33}$$

Where,

$$F_1 = (z_1 \max - z_1)/(z_1 \max - z_1 \min),$$
$$F_2 = (z_2 \max - z_2)/(z_2 \max - z_2 \min),$$
$$F_3 = (z_3 \max - z_3)/(z_3 \max - z_3 \min).$$

5.3 Solution spread measure

While it is desirable to find more Pareto-optimal solutions, it is also very much desirable to find the ones scattered uniformly over the Pareto frontier in order to provide a variety of compromise solutions to the decision maker. Solution spread measure (SSM) represents the distribution of the solutions along the Pareto front.

$$\text{SSM} = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - d \right|}{d_f + d_l + (N-1)d} \tag{34}$$

Where N is the number of solutions along the Pareto front, (so there are (N−1) consecutive distances), d_i is the distance (in objective space) between each solution, d is the arithmetic mean of all d_i and d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained non-dominated set. Thus, a low performance measure characterizes an algorithm with good distribution capacity.

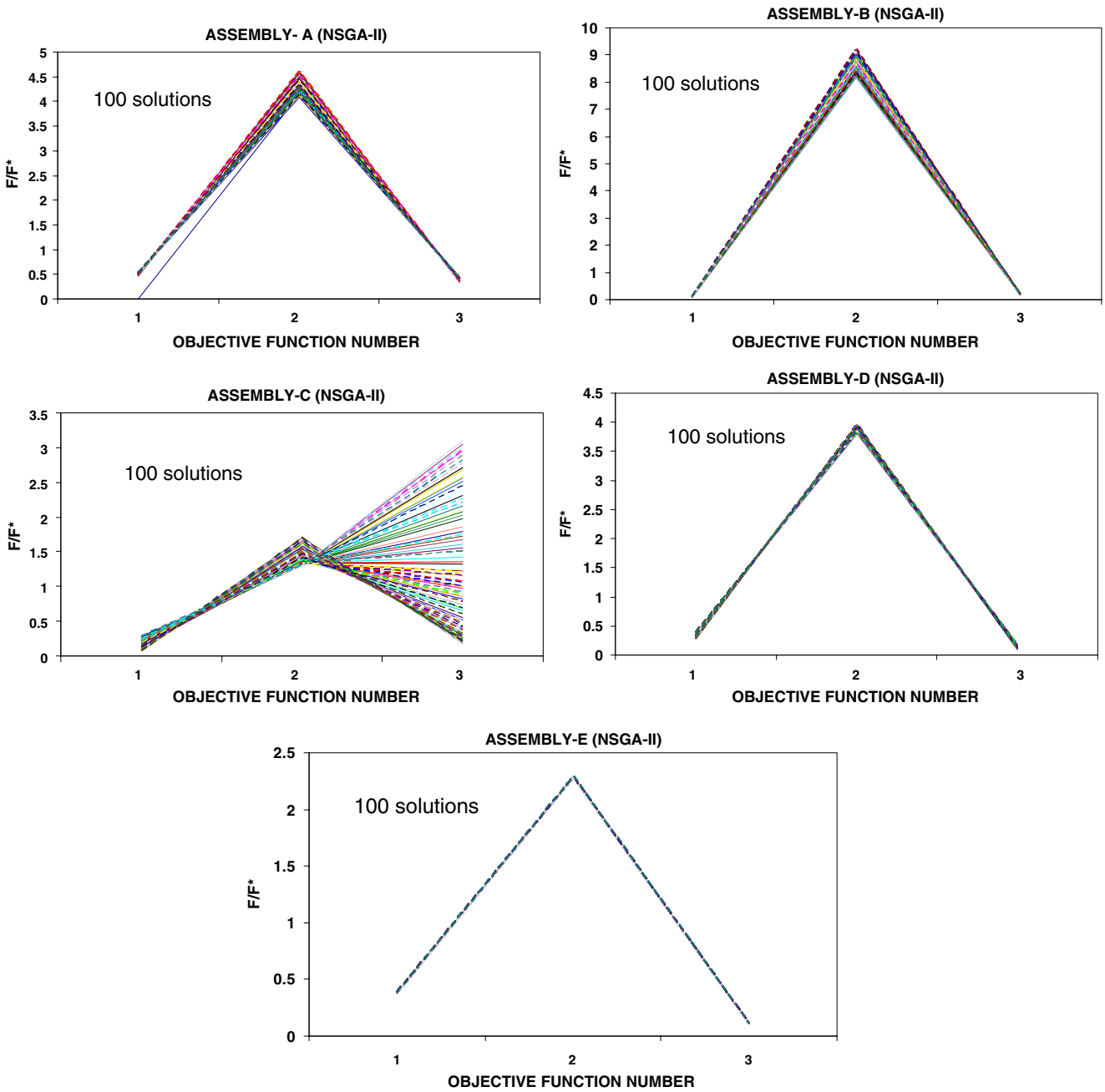

Fig. 11 Optimal solution trade-offs obtained from NSGA-II for Assembly A to E

5.4 Ratio of non-dominated individuals

This performance metric is defined as the ratio of non-dominated individuals (RNI) for a given population X,

$$\mathrm{RNI}(X) = \mathrm{nondom\_indiv}/P \tag{35}$$

Where nondom_indiv is the number of non-dominated individuals in population X and P is the size of population X. therefore the value RNI=1 means all the individuals in the population are non-dominated, and RNI=0 represents the situation where none of the individuals in the population is non-dominated. Since a population size of more than zero is often desired, there is always at least one non-dominated individual in the population within the range of $0<\mathrm{RNI}<1$.

5.5 Optimizer overhead

Total number of evaluations and total CPU time may be used for testing the algorithm. This would be useful in indicating how long and optimization or simulated evolution process would take in real world and to indicate the

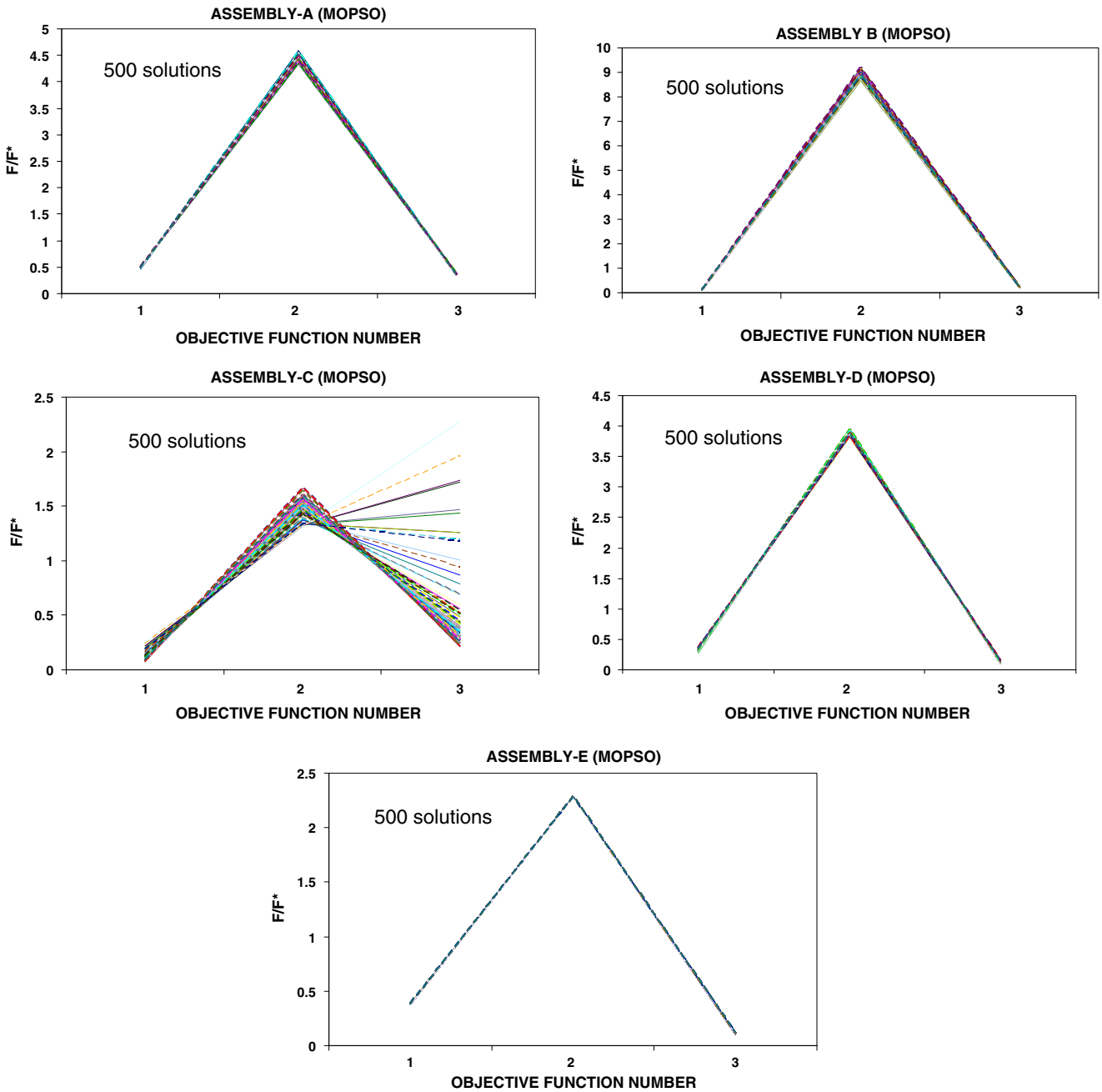

Fig. 12 Optimal solution trade-offs obtained from MOPSO for Assembly A to E

amount of program overhead as a result of the optimization manipulations such as those by evolutionary algorithm operators. More quantitatively, the optimizer overhead (OO) may be calculated by

$$\text{Optimiser overhead} = (T_{\text{Total}} - T_{\text{PFP}})/T_{\text{PFP}} \tag{36}$$

Where T_{Total} is the total time taken and T_{PFP} is the time taken for pure function evaluations. Thus, a value of zero indicates that an algorithm is efficient and does not have any overhead. However, this is an ideal case and is not practically reachable.

5.6 Algorithm effort

The performance in multi-objective optimization is often evaluated not only in terms of how the final Pareto front is, but also in terms of the computational effort required in obtaining the optimal solutions. For this purpose, the algorithm effort is defined as the ratio of the total number

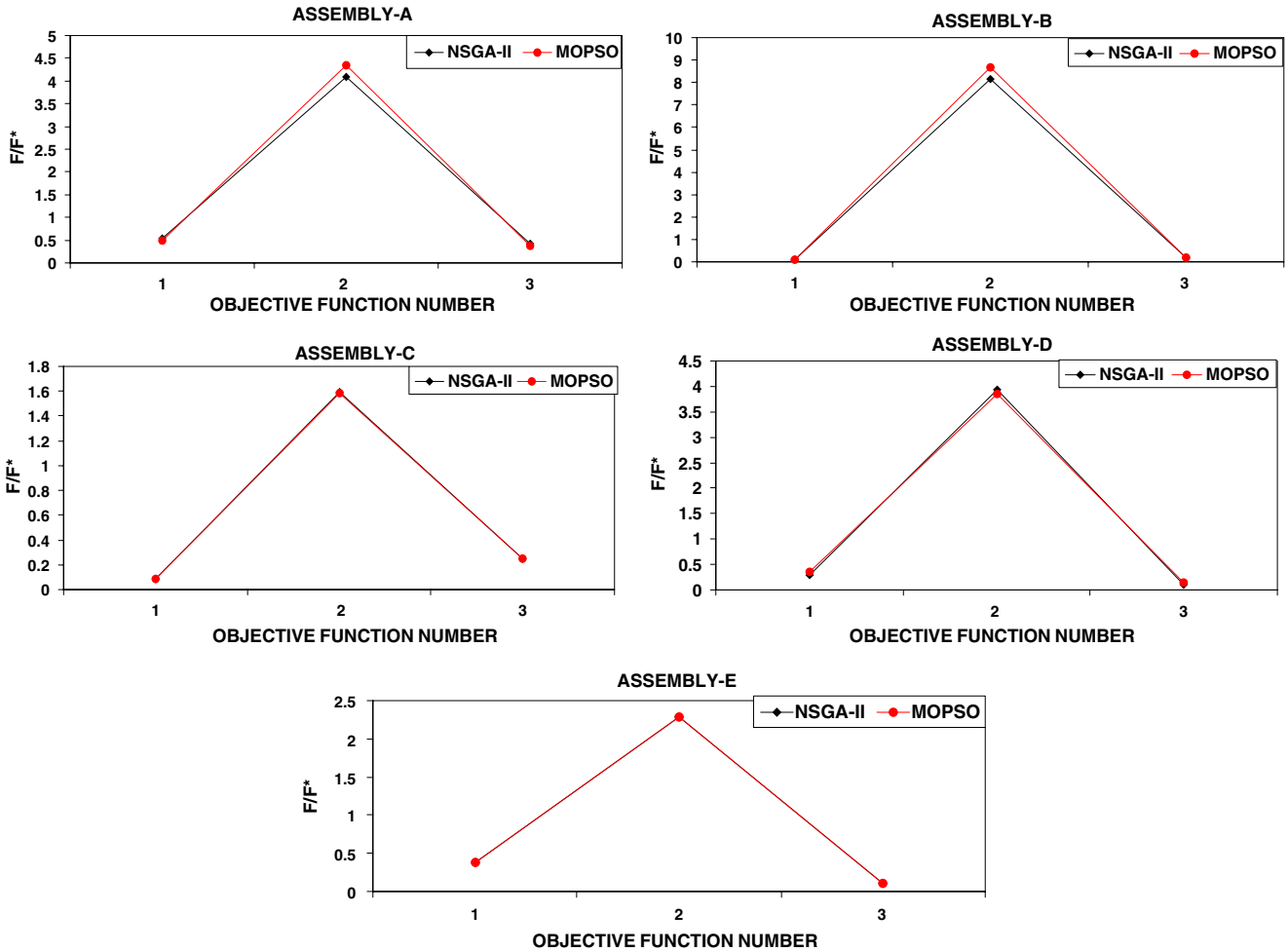

Fig. 13 Best solution trade-offs obtained from NSGA-II and MOPSO for Assemblies A to E

of function evolutions N_{eval} over a fixed period of simulation time T_{run},

$$\text{Algorithm effort} = T_{\text{run}}/N_{\text{eval}}, \left(T_{\text{run}} > T_{1\text{stgen}}\right) \cap \left(T_{\text{eval}} \propto N_{\text{eval}}\right) \tag{37}$$

As shown in above equation, for a fixed period of T_{run}, a greater number of function evolutions being performed indirectly indicates that less computational effort is required by the optimization algorithm and hence resulting in a smaller algorithm effort. The condition of $T_{\text{run}} > T_{1\text{stgen}}$, where $T_{1\text{stgen}}$ is the computation time for the first generation, should be held that T_{run} and N_{eval} are >0. This results algorithm effort is bounded in the range of (0,∞).

6 Numerical examples

In this paper, the assemblies Figs. 4, 5, 6, 7 and 8 considered by Singh et al. [25] are considered as numerical examples. The assemblies involving only a simple-dimension chain have been considered to make a comparison of the results obtained by the two methods possible. Details of the mechanical assemblies (number of dimensions, number of processes available for manufacture of each dimension, revised number of processes for manufacture of each dimension as applicable to the algorithm, parameters of cost functions, stack-up conditions, etc.) have been given in Tables 1, 2, 3, 4 and 5. Assembly B, which is simply Assembly A doubled, has been considered to study the effect of the problem size. Assemblies A, B, C, and D are linear assemblies, while Assembly E is a non-linear assembly. Processes marked with an asterisk are fictitious processes, added to fulfil the requirements of the algorithm. Assemblies C and D involve a few dimensions that can be produced on the same machine and hence it is desirable to have the same value of associated tolerances for reducing the number of set-ups. The impact of all the toleranced dimensions must be considered in the formulation of the assembly manufacturing cost and the stack-up condition.

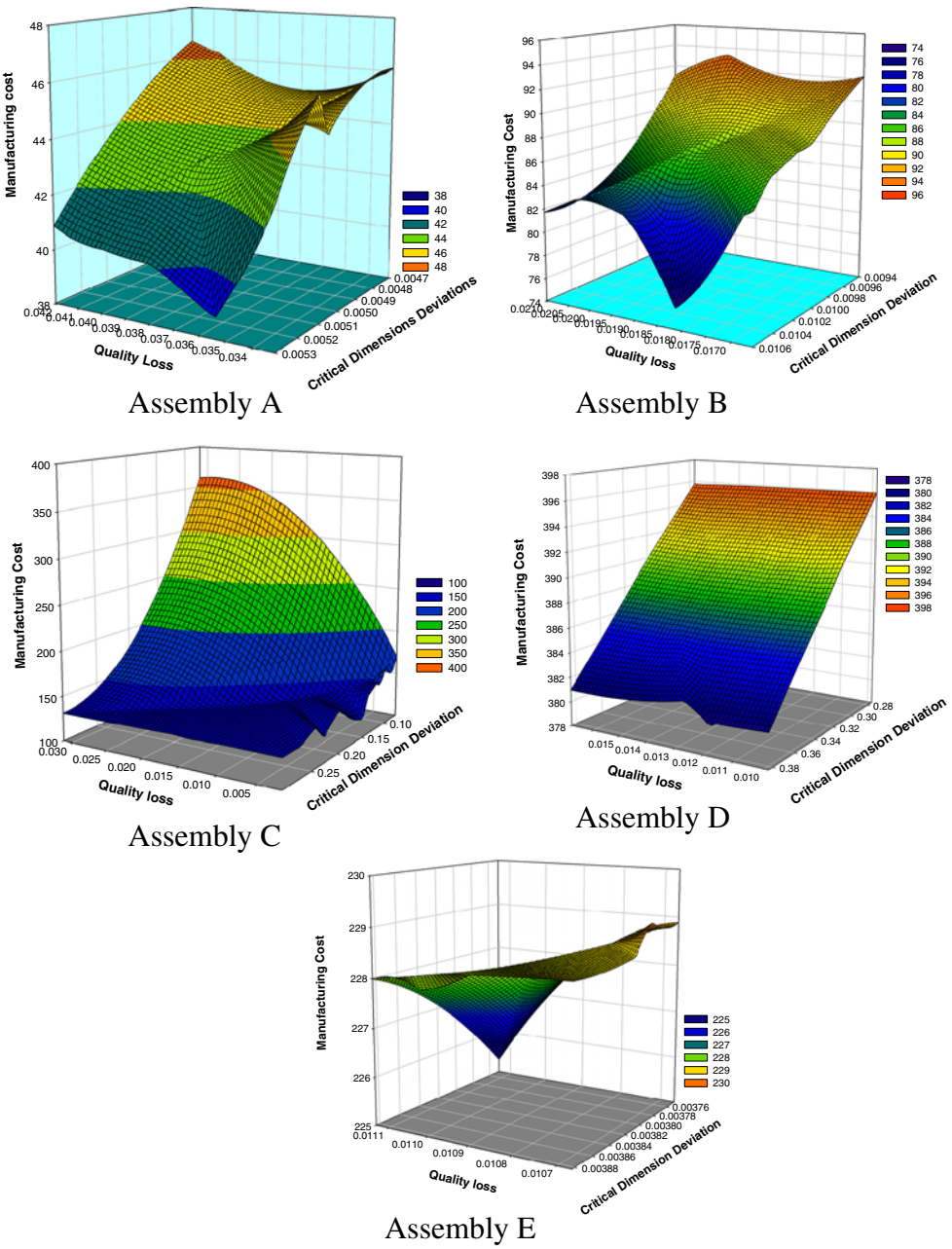

Fig. 14 Pareto-optimal fronts obtained from NSGA-II

Assemblies D and E involve a few vendor supplied components, the tolerances of which are considered fixed and do not count as decision variables.

6.1 Running NSGA-II and MOPSO algorithms

The objective functions such as tolerance stack-up, assembly manufacturing cost and quality loss function are minimized by considering the design constraints based on the assumed stack-up criteria (Eq. 11 for example B, Eq. 16 for Example C, Eq. 21 for example D and Eq. 26 for example E).

The steps for running NSGA-II and MOPSO algorithms are summarized below:

Step 1: The following are inputs to software program of NSGA-II and MOPSO:

1. Details of mechanical assemblies:

- Parameters cost functions and tolerance limits for Assembly A (given in Table 1)
- Parameters cost functions and tolerance limits for Assembly B (given in Table 2)

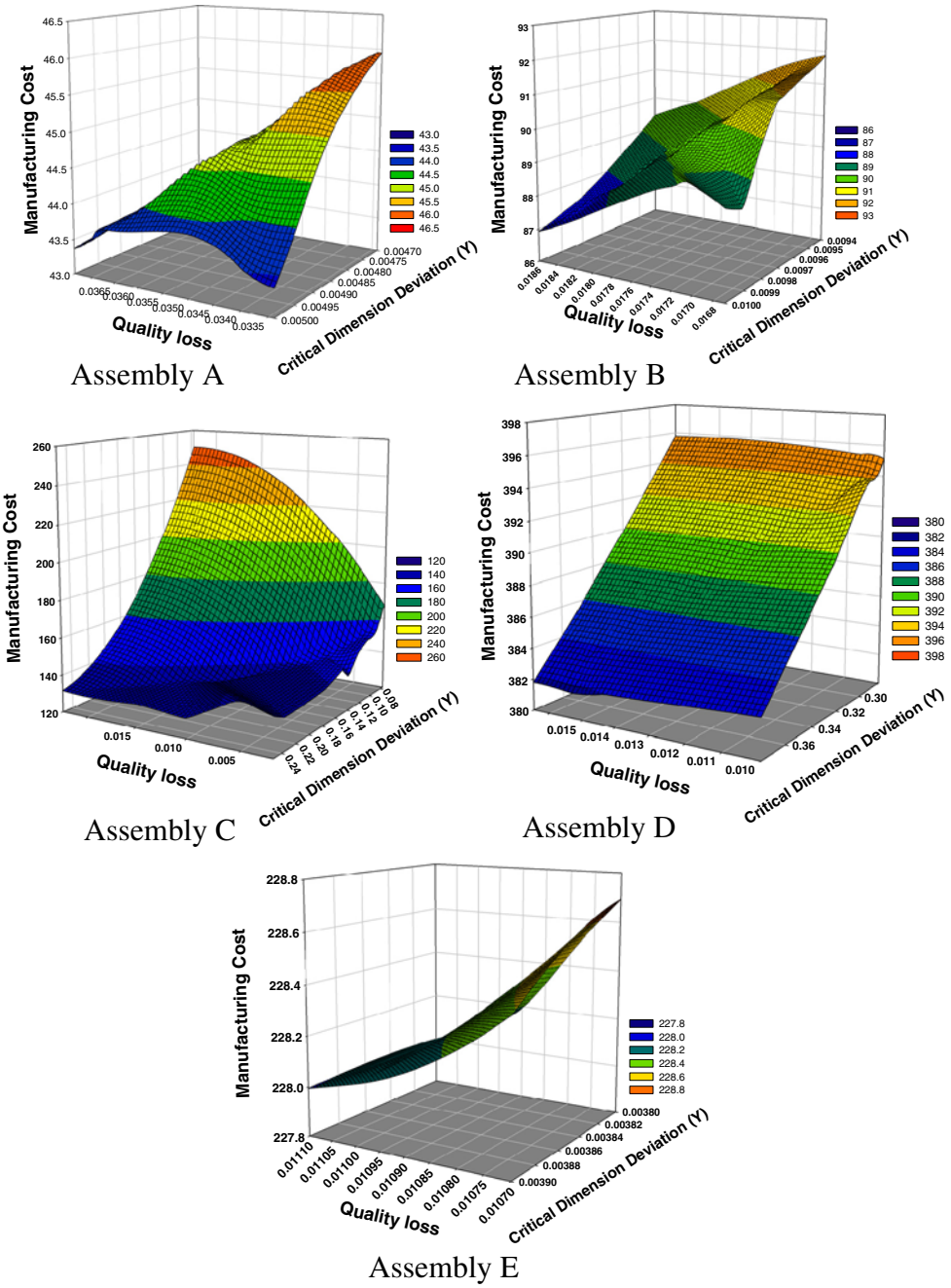

Fig. 15 Pareto-optimal fronts obtained from MOPSO

- Parameters cost functions and tolerance limits for Assembly C (given in Table 3)
- Parameters cost functions and tolerance limits for Assembly D (given in Table 4)
- Parameters cost functions and tolerance limits for Assembly E (given in Table 5)

2. Details of total manufacturing cost and quality loss formulae:(Eqs. 5 and 6 for Assembly A, Eqs. 8 and 9 for Assembly B, Eqs. 13 and 14 for Assembly C, Eqs. 18 and 19 for Assembly D, and Eqs. 23 and 24 for Assembly E)
3. The parameters NSGA-II and MOPSO algorithms (given in Sections 4.3 and 4.4)
4. Formulae to check all the constraints. (Eq. 11 for example B, Eq. 16 for Example C, Eq. 21 for example D and Eq. 26 for example E)

Step 2: The software programs of NSGA-II and MOPSO will find the optimal tolerances values (t_{ij}) in such a way that

1. All objective function (F_c) are minimum.
2. All constraints (tolerance stack-up) are satisfied.

Step 3: Step 2 will be repeated up to the maximum number of iterations.

Step 4: The following are the outputs from NSGA-II and MOPSO algorithms:

1. Optimal solutions obtained from NSGA-II and MOPSO algorithms. Each solution shall have the optimal objective functions value, optimal value of variables and the constraints value.
2. The best optimal solution shall give the minimum stack-up tolerance, minimum total manufacturing cost, minimum quality loss and optimal tolerances values, etc.

7 Results and discussion

Tables 6, 7, 8, 9, and 10 compare the optimum results obtained from various techniques for all the assemblies (A–E). From Tables 6, 7, 8, 9, and 10, it is observed that MOPSO gives better results than NSGA-II in majority of the cases.

The results of average fitness factor value (F_{avg}), SSM, RNI, OO and algorithm effort obtained from NSGA-II and MOPSO are listed in Tables 11, 12, 13 and 14 for all the assemblies A–E. From Tables 11, 12, 13, and 14, it is observed that MOPSO technique gives the maximum average fitness factor (F_{avg}), minimum algorithm effort and minimum OO than those of NSGA-II for all the assemblies. From Table 14, it is observed that MOPSO technique gives the minimum SSM than that of NSGA-II for Assemblies A, B and D. But NSGA-II gives, minimum SSM than those of MOPSO for Assemblies C and E. Both NSGA-II and MOPSO gives the same RNI and minimum combined objective function (f_c) value for all the Assemblies A–E.

The values of the allocated tolerance of part dimensions of the product with respect to the number of iterations are shown in Figs. 9 and 10 obtained from NSGA-II and MOPSO for all the assemblies. From Figs. 9 and 10, the optimum combination of process that gives a better result (minimum manufacturing cost) than the other process combinations for each assemblies is selected.

The best solution is selected by the average fitness factor method from the optimal solution trade-offs obtained from NSGA-II and MOPSO. The optimal solution trade-offs (Pareto-optimal fronts) obtained from NSGA-II and MOPSO are given in Figs. 11 and 12 respectively for Assemblies A–E. From Figs. 11 and 12, it is observed that MOPSO gives the optimal solution trade-offs with more number of non-dominated solutions for user's choice than NSGA-II.

The best solution trade-offs selected by the average fitness factor method from the optimal solution trade-offs obtained from NSGA-II and MOPSO are shown in Fig. 13. From Fig. 13, it is noted that NSGA-II gives the best results for the objective functions (minimum tolerance stack-up (Z_1) for Assemblies C and E and minimum manufacturing cost (Z_2) for Assemblies A, B and D and minimum quality loss (Z_3 and Z_5) for Assembly E). But MOPSO gives the best result for the objective functions (minimum tolerance stack-up (Z_1) for Assembly D and minimum manufacturing cost (Z_2, Z_4) for Assembly C, E and F and minimum quality loss (Z_3) for assemblies A, B, C and D). So both NSGA-II and MOPSO algorithms are best for this problem.

The Pareto-optimal fronts obtained from NSGA-II and MOPSO are given in Figs. 14 and 15 for all the assemblies. From Figs. 14 and 15, it is observed that both algorithms simultaneously minimize stack-up tolerance, manufacturing cost and quality loss of the product with respect to the number of iterations.

8 Conclusions

This study is based on a new general methodology using NSGA-II and MOPSO for the optimal tolerance allocation and alternative process selection for the mechanical assemblies. The average fitness factor method and normalized weighted objective function method are used to select the best optimal solution from Pareto-optimal fronts. Two multi-objective performance measures viz., solution spread measure and ratios of non-dominated individuals are used to evaluate the strength of the Pareto-optimal fronts. Two more multi-objective performance measures namely optimizer overhead and algorithm effort are used to find the computational effort of NSGA-II and MOPSO algorithms. The Pareto-optimal fronts (optimal solution trade-offs) and results obtained from various techniques are compared and analysed. The results indicate that MOPSO technique gives the maximum average fitness function (μ_{avg}), minimum SSM, minimum OO, and minimum algorithm effort than those of NSGA-II in majority of the cases i.e., it is faster than NSGA-II technique. Also the computational time to find the optimum solutions in MOPSO is one-third of that in NSGA-II. MOPSO is faster than NSGA-II. So MOPSO is superior to NSGA-II for this problem, if the user wants a best optimal solution quickly. But both MOPSO and NSGA-II technique gives the same RNI and minimum combined objective function (f_c). Also MOPSO gives the best Pareto-optimal front with more number of non-dominated solutions for user's choice than NSGA-II. So MOPSO and NSGA-II are the best for this multi-criterion optimisation problem. This work opens the door for further investigations on how the evolutionary optimisation techniques can be used to solve complex problems.

Appendix A

The pseudo code of MOPSO algorithm is given below:

1. For i=1 to M (M is the population size)
   (a) Initialize P[i] randomly (P is the population of particles)
   (b) Initialize V[i]=0 (V is the speed of each particle)
   (c) Evaluate P[i]
   (d) Initialize the personal best of each particle
   PBESTS[i]=P[i]
   (e) GBEST=best particle found in P[i]
2. End for
3. Initialize the iteration counter t=0
4. Store the non-dominated vectors found in P into A
   (A is the external archive that stores non-dominated solutions found in P)
5. Repeat
   (a) Compute the crowding distance values of each non-dominated solution in the archive A
   (b) Sort the non-dominated solutions in A in descending crowding distance values
   (c) For i=1 to M
      1. Randomly select the global best guide for P[i] from a specified top Portion (e.g., top 10%) of the sorted archive A and store its position to GBEST.
      2. Compute the new velocity:

$$V[i] = W \times V[i] + \mathrm{R1} \times (\mathrm{PBESTS}[i] - P[i]) + \mathrm{R2} \times (A[\mathrm{GBEST}] - P[i])$$

      (W is the inertia weight equal to 0.4)
      (R1 and R2 are random numbers in the range [0..1])
      (PBESTS[i] is the best position that the particle i have reached)
      (A[GBEST] is the global best guide for each non-dominated solution)
      3. Calculate the new position of P[i]:

$$P[i] = P[i] + V[i]$$

      4. If P[i] goes beyond the boundaries, then it is reintegrated by having the decision variable take the value of its corresponding lower or upper boundary and its velocity is multiplied by −1 so that it searches in the opposite direction.
      5. If (t<(MAXT×PMUT),
      then perform mutation on P[i].
      (MAXT is the maximum number of iterations)
      (PMUT is the probability of mutation)
      6. Evaluate P[i]
   (d) End for
   (e) Insert all new non-dominated solution in P into A if they are not dominated by any of the stored solutions.
      1. Compute the crowding distance values of each non-dominated solution in the archive A
      2. Sort the non-dominated solutions in A in descending crowding distance values
      3. Randomly select a particle from a specified bottom portion (e.g., lower 10%) which comprise the most crowded particles in the archive then replace it with the new solution
   (f) Update the personal best solution of each particle in P. If the current PBESTS dominates the position in memory, the particles position is updated using
   PBESTS[i]=P[i]
   (g) Increment iteration counter t
6. Until maximum number of iterations is reached

References

1. Ye B, Salustri FA (2003) Simultaneous tolerance synthesis for manufacturing and quality. Res Eng Des 14(2):98–106
2. Singh PK, Jain SC, Jain PK (2003) Tolerance allocation with alternative manufacturing processes-suitability of genetic algorithm. Proceeding Inst Mech Eng J Eng Manufacture 2(1–2):22–34
3. Singh PK, Jain SC, Jain PK (2004) A GA based solution to optimum tolerance synthesis of mechanical assemblies with alternate manufacturing processes: focus on complex tolerancing problems. Int J Prod Res 42(24):5185–5215
4. Prabhaharan G, Asokan P, Ramesh P, Rajendran S (2004) Genetic algorithm-based optimal tolerance allocation using least-cost model. Int J Adv Manuf Technol 24:647–660
5. Prabhaharan G, Asokan P, Rajendran S (2005) Sensitivity-based conceptual design and tolerance allocation using the continuous ants colony algorithm (CACO). Int J Adv Manuf Technol 25:516–526
6. Krishna G, Mallikarjuna Rao K (2006) Simultaneous optimal selection of design and manufacturing tolerances with different stack-up conditions using scatter search. Int J Adv Manuf Technol 30(3–4):328–333. doi:10.1007/s00170-005-0059-0
7. Huang YM, Shiau C-S (2006) Optimal tolerance allocation for a sliding vane compressor. J Mech Des 128(1):98–107
8. Huang MF, Zhong YR (2007) Optimized sequential design of two-dimensional tolerances. Int J Adv Manuf Technol 33:579–593
9. Singh PK, Jain PK, Jain SC (2008) Optimal tolerance design of mechanical assemblies for economical manufacturing in the presence of alternative machines—a genetic algorithm-based hybrid methodology. Proceeding Inst Mech Eng J Eng Manufacture 222B:591–604
10. Sivakumar M, Kannan SM, Jayabalan V (2009) A new algorithm for optimum tolerance allocation of complex assemblies with alternative processes selection. Int J Adv Manuf Technol 40(7–8):819–836
11. González I, Sánchez I (2009) Statistical tolerance synthesis with correlated variables. Mech Mach Theory 44(6):1097–1107
12. Huang M, Zhong Y (2008) Dimensional and geometrical tolerance balancing in concurrent design. Int J Adv Manuf Technol 35(7–8):723–735. doi:10.1007/s00170-006-0749-2

13. Forouraghi B (2009) Optimal tolerance allocation using a multi-objective particle swarm optimizer. Int J Adv Manuf Technol 44:710–724
14. Siva Kumar M, Stalin B (2009) Optimum tolerance synthesis for complex assembly with alternative process selection using Lagrange multiplier method. Int J Adv Manuf Technol 44:405–411
15. Wu F, Dantan J-Y, Etienne A, Siadat A, Martin P (2009) Improved algorithm for tolerance allocation based on Monte Carlo simulation and discrete optimization. Comput Ind Eng 56(4):1402–1413
16. Muthu P, Dhanalakshmi V, Sankaranarayanasamy K (2009) Optimal tolerance design of assembly for minimum quality loss and manufacturing cost using metaheuristic algorithms. Int J Adv Manuf Technol 44:1154–1164
17. Noorul Haq A, Sivakumar K, Saravanan R, Muthiah V (2005) Tolerance design optimization of machine elements using genetic algorithm. Int J Adv Manuf Technol 25:385–391
18. Singh PK, Jain PK, Jain SC (2003) Simultaneous optimal selection of design and manufacturing tolerances with different stack-up conditions using genetic algorithms. Int J Prod Res 41 (11):2411–2429
19. Singh PK, Jain SC, Jain PK (2005) Advanced optimal tolerance design of mechanical assemblies considering interrelated dimension chains and process precision limits. Comput Ind 56(2):179–194
20. Singh PK, Jain SC, Jain PK (2005) Comparative study of genetic algorithm and simulated annealing for optimal tolerance design formulated with discrete and continuous variables. Proceeding Inst Mech Eng J Eng Manufacture 219B:735–759
21. Coello Coello C, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, 2nd edn. Springer, New York
22. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
23. Knowles J, Corne D, Deb K (eds) (2008) Multiobjective problem solving from nature: from concepts to applications. Springer, New York
24. Salazar D, Rocoo CM (2007) Solving advanced multi-objective robust designs by means of multiple objective evolutionary algorithms (MOEA): a reliability application. Reliab Eng Syst Saf 92:697–706. doi:10.1016/j.ress.2006.03.003
25. Singh PK, Jain SC, Jain PK (2004) A genetic algorithm based solution to optimum tolerance synthesis of mechanical assemblies with alternate manufacturing processes—benchmarking with the exhaustive search method using the Lagrange multiplier. Proceeding Inst Mech Eng J Eng Manufacture 218 B:765–778
26. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
27. Price K, Storn R (1997) Differential evolution—a simple evolution strategy for fast optimization. DrDobb's Journal 22 (4):18–24 & 78
28. Tan KC, Lee TH, Khor EF (2002) Evolutionary algorithms for multi-objective optimization: performance assessments and computations. Artif Intell Rev 17:253–290
29. Singh PK, Jain SC, Jain PK (2006) Concurrent optimal adjustment of nominal dimensions and selection of tolerances considering alternative machines. Comput Aided Des 38:1074–1087