Int J Adv Manuf Technol (2009) 44:710–724
DOI 10.1007/s00170-008-1892-8

ORIGINAL ARTICLE

Optimal tolerance allocation using a multiobjective particle swarm optimizer

Babak Forouraghi

Received: 24 March 2008 / Accepted: 4 December 2008 / Published online: 9 January 2009

Abstract Particle swarm optimizers are routinely utilized in engineering design problems, but much work remains to take advantage of their full potential in the combined areas of sensitivity analysis and tolerance synthesis. In this paper, a novel Pareto-based multiobjective formulation is proposed to enhance the operations of a particle swarm optimizer and systematically distribute tolerances among various components of a mechanical assembly. The enhanced algorithm relies on nonlinear sensitivity analysis and the statistical root sum squares model to simultaneously optimize product performance criteria, the manufacturing cost, and the stack-up tolerance. It is shown that the proposed algorithm can accomplish its optimization task by successfully shifting nominal values of design parameters instead of the expensive tightening of component tolerances. Several numerical experiments for optimal design of a stepped bar assembly were conducted, which highlight the advantages of the proposed methodology.

Keywords Concurrent engineering · Multiobjective optimization · Particle swarm optimization · Sensitivity analysis · Tolerance allocation

1 Introduction

Optimum tolerance allocation is an effective robust design and concurrent engineering tool to reduce manufacturing costs while increasing the overall robustness of the product [1–3].

Evolutionary algorithms have proven successful in handling many real-world multiobjective concurrent engineering problems [4–7]. For example, an evolutionary radial basis function network was proposed to model a robust design optimizer, and the developed system was used to study a number of computationally expensive multiobjective optimizations problems [8].

Recently, approaches based on particle swarm optimization (PSO) have received a great deal of attention in engineering design because of their simplicity and fast convergence rates [9–11]. A multiobjective particle swarm optimizer was developed to evolve generations of hyper-rectangular particles where dimensional tolerances were treated as intervals. This optimizer did not directly minimize manufacturing costs but rather attempted to widen design tolerances by maximizing hyper-rectangular design volumes in successive generations [12]. Several other improved PSO systems have been proposed and successfully used in challenging engineering design problems [13–17], but none of them included any sensitivity analysis study while performing parameter design.

The major goal of sensitivity analysis is to identify the sensitive dimensions of parameters in a product or process design. Sensitivity analysis is an important component in robust engineering design where the values of design parameters are prone to many sources of uncertainty and variation. It is, therefore, necessary to understand the sensitivity of the product or process responses to the perturbation in the model inputs [18]. An ant colony optimizer was developed to perform sensitivity analysis and tolerance allocation in concept design by optimizing the manufacturing cost and the product yield using an aggregated form of design objectives [19]. It has, however,

B. Forouraghi (✉)
Mathematics and Computer Science Department,
Saint Joseph's University,
Philadelphia, PA 19131, USA
e-mail: bforoura@sju.edu

been shown that the traditional weighted aggregated method suffers from many shortcomings, the most important of which is its assumption of convexity of the Pareto region, which is generally invalid in many real-world engineering design problems. Furthermore, Pareto fronts are typically not uniform, that is, the obtained solutions are not uniformly distributed on the Pareto front given a set of evenly distributed weights [20]. In another approach based on stochastic integer programming, a quality loss function was developed to include cost tolerances, manufacturing costs, and design constraints [21]. The traditional approaches to robust tolerance design, however, generally do not scale up as the complexity of the problem domain increases [3].

A PSO-based approach was proposed for worst-case circuit design of LC high-pass filters [22] but with the main underlying assumption that all the individual worst-case tolerance limits occur at the same time. This shortcoming can potentially result in unnecessarily tight tolerances and high manufacturing costs [3]. A PSO system was reported for achieving the multiple objectives of minimum quality loss function and minimum manufacturing cost for the machining tolerance allocation of an over running clutch assembly [23]. This system utilized the weighted aggregated method, which as explained before, cannot successfully identify all the trade-off solutions along the Pareto front.

In this paper, a novel Pareto-based multiobjective formulation is proposed to enhance the operations of a particle swarm optimizer and optimally distribute tolerances among various components of a mechanical system. The enhanced algorithm relies on nonlinear sensitivity analysis and the statistical root sum squares (RSS) model to simultaneously optimize product performance criteria, the manufacturing cost, and the stack-up tolerance. It is shown that the proposed algorithm can accomplish its optimization task by efficiently exploring the Pareto front and successfully shifting nominal values of design parameters instead of the expensive tightening of component tolerances. Furthermore, the algorithm is able to operate in a mixed mode where some of the design tolerances must be kept fixed due to manufacturing restrictions while others are allowed to widen.

The remainder of the paper is organized as follows. Sections 2 and 3 provide brief descriptions of sensitivity analysis and multiobjective optimization, respectively. Section 4 introduces the particle swarm optimization methodology along with the implemented enhancements for more efficient coverage of the feasible design region. Section 5 is the application of the proposed methodology to multiobjective concurrent engineering design of a stepped bar assembly. And finally, Section 6 is the summary and conclusions.

2 Sensitivity analysis

The major goal of robust engineering design is to minimize functional sensitivity of a product or process to parametric changes caused by uncontrollable operational and manufacturing conditions. Sensitivity analysis is an important component of robust design in that it utilizes various optimization methods to identify the sensitive dimensions of parameters in a design, and more importantly, it aims to ensure that effects on product performance brought by changes in the design parameters are minimal.

There are various approaches to performing sensitivity analysis. For instance, mathematical sensitivity analysis, component manufacturing process output distribution sensitivity analysis, nonlinear sensitivity analysis, and statistical sensitivity analysis, to name a few. The following is a brief description of the use of nonlinear sensitivity analysis in concept design as it directly relates to the focus of this work. More thorough discussions of the wide topic of sensitivity analysis, however, can be found elsewhere [1, 2, 18, 19, 21].

2.1 Nonlinear sensitivity analysis

The functional relationship between the independent design parameters x_1, x_2, ..., x_n and the dependent product or process response Y can be expressed as:

$$Y = f(x_1, x_2, \ldots, x_n) \tag{1}$$

In many robust engineering design applications, however, the above relationship is not linear in nature. Therefore, small changes in the response may be expressed by a Taylor's series expansion as given by [2]:

$$\Delta Y = \sum_i = \frac{\delta f}{\delta x_i} \Delta x_i + \frac{1}{2} \sum_i \sum_j \frac{\delta^2 f}{\delta x_i \delta x_j} \Delta x_i \Delta x_j + \cdots \tag{2}$$

The common practice for performing nonlinear sensitivity analysis is to evaluate the first term from the above Taylor series expansion and to manually calculate a sensitivity S by taking the partial derivative of the functional relationship between the dependent dimension and the independent component dimensions as $S_i = \delta Y / \delta x_i$.

Manufacturing process variations in the real-world situations manifest themselves as distributions around the nominal design variable values and may cause the deviation of model response around its nominal value. This model deviation can be mathematically expressed as:

$$\delta Y = f(S_1 \delta x_1, S_2 \delta x_2, \ldots, S_n \delta x_n) \tag{3}$$

Here, δY is the product response variation, and S_i and δx_i are the sensitivity and deviation for the ith design variable, respectively.

The variance of the product response function can then be expressed as a sum of the design variable variances as attenuated by their respective sensitivity coefficients:

$$dY^2 = \sum_{i=1}^{n} (S_i \times dx_i)^2 \tag{4}$$

A careful examination of the above relation reveals two design opportunities to control the deviation of the product response Y. First, dY can be kept small by tightening manufacturing tolerances dx_i. And second, S_i can be reduced to make dY insensitive to independent design parameter variations. The first approach, which was traditionally used, is generally more expensive as it requires tighter manufacturing tolerances and protection from aging and the environment. The second way to control the product or process deviation is more practical as it can be accomplished without tightening manufacturing tolerances but by simply altering the value of $S_i \times dx_i$.

The above approach is very helpful in avoiding the cost associated with quality improvement based on buying down variance by tolerance tightening. In fact, any number of conceptual designs can be analyzed very early in the concept development stage using various optimization techniques to minimize critical dimension variance [18, 19, 21].

3 Multiobjective optimization

Multiobjective optimization (MO) is a methodology for finding optimal solutions to multivariate problems with multiple, often conflicting, objectives [4, 5]. The main goal of MO is to find the optimum input parameter vector, which results in some desired combination of maximization, minimization, or nominalization of the involved product or process responses. Mathematically, MO attempts to optimize the p-dimensional vector function F of objective responses $F(X) = [f_1(X), \ldots, f_p(X)]^T$ where $X = [x_1, \ldots, x_n]^T$ is an n-dimensional vector of design variables and p is the number design objectives for a product. The problem can be stated formally as follows (l is the inequality and m is the equality constraints).

$$\begin{array}{ll} \text{Minimize/Maximize :} & f_i(X)\ for\ i = 1, \ldots, p \\ \text{Subject to :} & g_j(X) \le 0 \text{ for } j = 1, \ldots, l \\ & h_k(X) = 0 \text{ for } k = 1, \ldots, m \end{array}$$

Since MO problems often have conflicting objectives, it is virtually impossible to find any one ideal solution. Instead, MO produces Pareto-optimal solutions. A design vector is Pareto optimal if there is no other design vector that optimizes one criterion without causing the simultaneous degradation of at least one other criterion [24]. Without loss of generality, it can be stated that the goal is to maximize p objective responses considering the following key definitions [4, 5, 13]:

Definition 1. A vector $u = (u_1, \cdots, u_p)$ is said to be *inferior to* (*dominated by*) vector $v = (v_1, \cdots, v_p)$ if and only if $\forall i \in \{1, \cdots, p\}$, $v_i \ge u_i \wedge \exists i \in \{1, \cdots, p\} |\ v_i > u_i$.

Definition 2. Vectors u and v are said to be *noninferior* to each other if neither u is inferior to v nor v is inferior to u.

The main challenge of MO is to develop efficient algorithms that can quickly converge to well-spread Pareto fronts in the feasible, nondominated design regions. This and other pertinent issues relating to the specifics of MO can be found elsewhere [14–17].

4 Particle swarm optimization

PSO is a stochastic global optimization approach, and its main strength is in its simplicity and fast convergence rates. The following is a brief introduction to PSO [9–11]:

A total of p particles are randomly distributed throughout the feasible design region, where X_i^t is the position of a particle i representing a design scenario at time t. The position of the particle can be updated in the following manner:

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{5}$$

with a velocity V_i^{t+1} which is calculated as:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 \left(P_i^t - X_i^t\right) + c_2 r_2 \left(P_g^t - X_i^t\right) \tag{6}$$

Here, the point P_i^t is the best local solution found up to now (time t) and represents the cognitive contribution to the search vector V_i^{t+1}. The point P_g^t is the best global solution found among all particles so far and forms the social contribution to the velocity vector. Random numbers r_1 and r_2 are uniformly distributed in the interval [0,1]. The cognitive and social scaling factors c_1 and c_2 are typically selected such that $c_1 \times c_1$ and $c_2 \times c_2$ have a mean of 1 so that the particles overshoot the attraction points P_i^t and P_g^t half the time, thereby allowing wider search fronts [25]. The variable ω is the inertia weight and is typically chosen in the range of [0,1]. A larger inertia weight facilitates global exploration, and a smaller inertia tends to facilitate local exploration. Therefore, ω is a critical factor for the convergence behavior of PSO and is used to promote global exploration of the search space [11].

The cognitive learning factor is computed by the term $c_1 r_1 \left(P_i^t - X_i^t\right)$ in Eq. 6, and it is the short-term memory of a particle representing the particle's inclination to repeat past

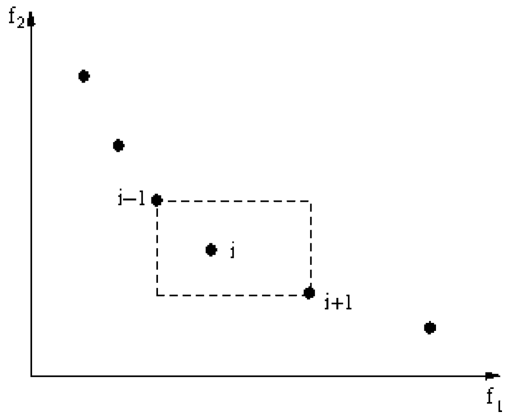

Fig. 1 The nearest neighbor density estimator

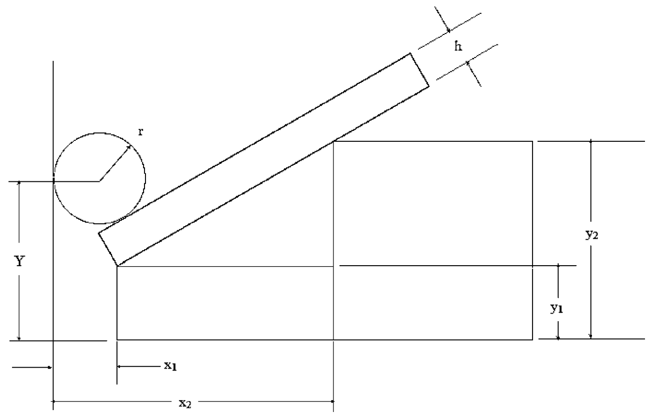

Fig. 3 Stepped bar assembly

behavior that has proven to be successful for that particular particle.

The social learning factor, on the other hand, is computed by the term $c_2\, r_2 \left(P_g^t - X_i^t\right)$ in Eq. 6, and it is the peer pressure of a particle representing the particle's inclination to imitate or emulate the behavior of other particles that are successful; it is the influence of a particle's neighbors.

During the course of the PSO algorithm, swarm particles are evaluated, and based on those evaluations, the new velocities and positions of the particles are updated. The basic equations (Eqs. 5 and 6) for PSO were utilized in this work, but a few key elements were modified in accordance with enhancements described in the following sections.

4.1 Crowding distance computation

The nearest neighbor density estimator quantifies how crowded the closest neighbors of a given particle are in the objective space. As illustrated in Fig. 1, this measure is estimated by the area of the largest cuboid formed by using the two nearest neighbors of particle i as the vertices [11].

Before computing the crowding distance (d_c) for particles on the boundary of the feasible region, the entire nondominated population is first sorted based on the increasing values of their objective functions, one at a time. A d_c^j measure for a particular particle P_i^j is then defined as the average distance of its two nearest neighbors P_{i-1}^j and P_{i+1}^j along the dimension of a specific objective function j. The total d_c value for a particle is then computed as $\sum_{j=1}^{M} d_c^j$, where M is the total number of design objective functions. In the case of the two extreme solutions with the highest and lowest objective function values, the d_c measure is set to infinity so the two boundary points will always be selected [4, 26].

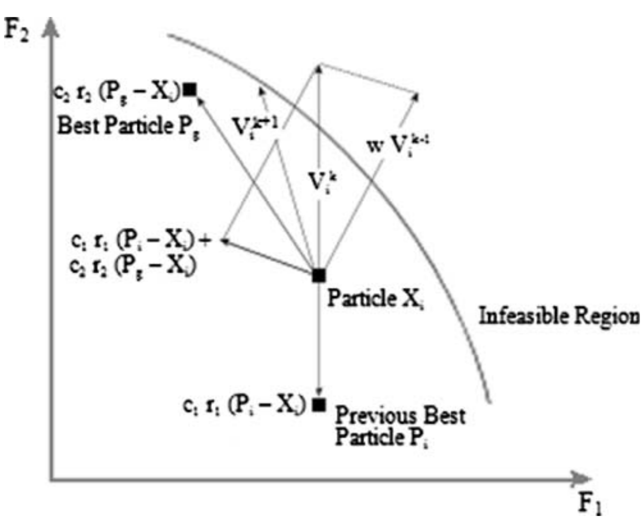

Fig. 2 The fly-back mechanism

4.2 Elitist mutation

Although one of the greatest advantages of PSO-based approaches are their simplicity—both conceptually and from the standpoint of implementation—these stochastic can also experience difficulty controlling population diversity while dealing with multiple-objective optimization problems [4].

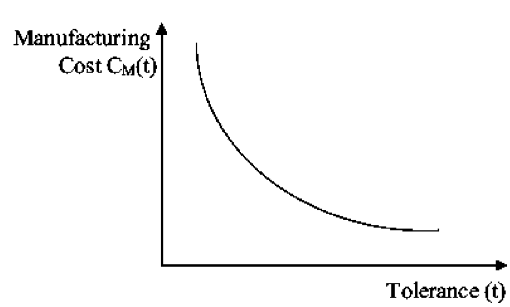

Fig. 4 A cost-tolerance curve

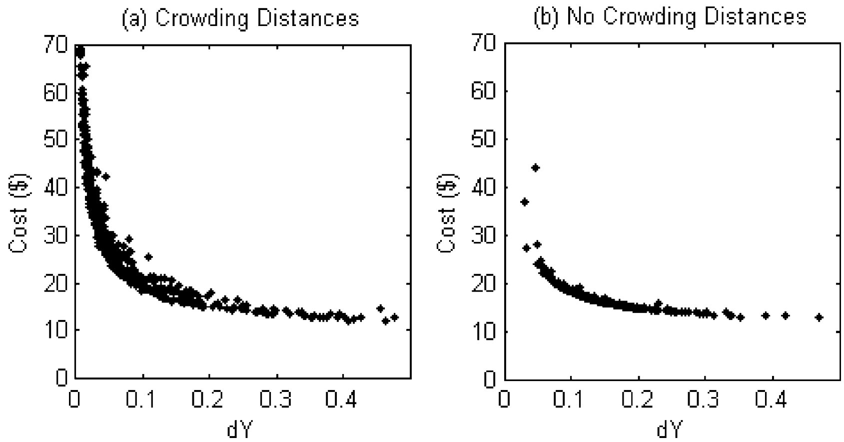

Fig. 5 Discovered Pareto fronts. **a** Crowding distances. **b** No crowding distance

The loss of diversity along the Pareto front can potentially be avoided by utilizing the mutation operation.

The mutation operator implemented in this work is based on the elitist-mutation mechanism that was proposed to improve the performance of the particle swarm algorithm [16]. The main objective here is to best identify the true Pareto-optimal front by promoting diversity among the particles maintained in the repository. In its initial phase, the elitist-mutation operator replaces the infeasible solutions in the repository with the least crowded ones, and in its later phases, it will exploit the sparsely explored regions along the Pareto front. The main steps to mutation are outlined below:

1. Sort and index the particles in the repository in ascending order based on their value of a randomly selected objective.
2. Use the crowding distance metric to sort the current solutions in the repository in descending order and randomly select a solution in the top 10%.
3. Apply mutation to a predefined number of particles in the current population.

4.3 Maintaining feasibility

For the PSO algorithm to maintain feasibility in the population kept in the repository, an intuitive approach is to fly back a particle to its previous position when it is outside the feasible region [27]. This is called the *fly-back* mechanism, and it is a particularly useful strategy when solving engineering optimization problems with multiple geometric constraints. Each time a constraint is violated, the particle in flight is made to revisit its previous position, allowing effective exploration of the search space along the boundaries of the feasible region.

In this study, a fly-back mechanism to the feasible region was implemented as illustrated in Fig. 2.

Figure 2 illustrates how a particle X_i might have flown into the infeasible design region $\left(V_i^k\right)$ in the kth iteration of

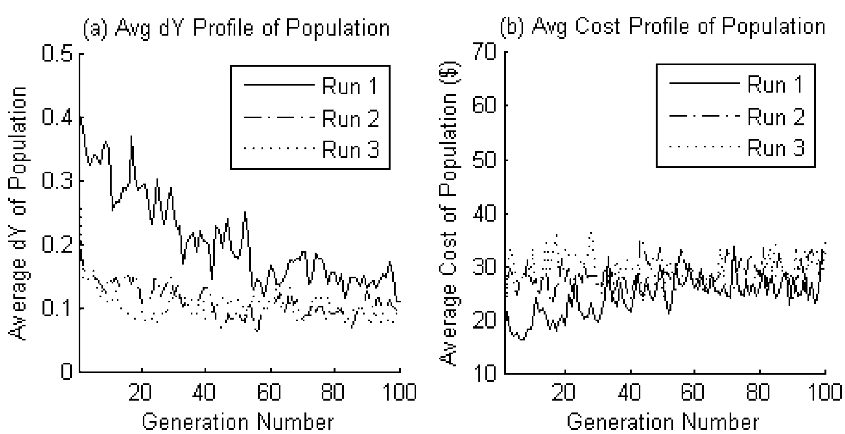

Fig. 6 Average functional behavior of particles. **a** Average dY profile of population. **b** Average cost profile of population

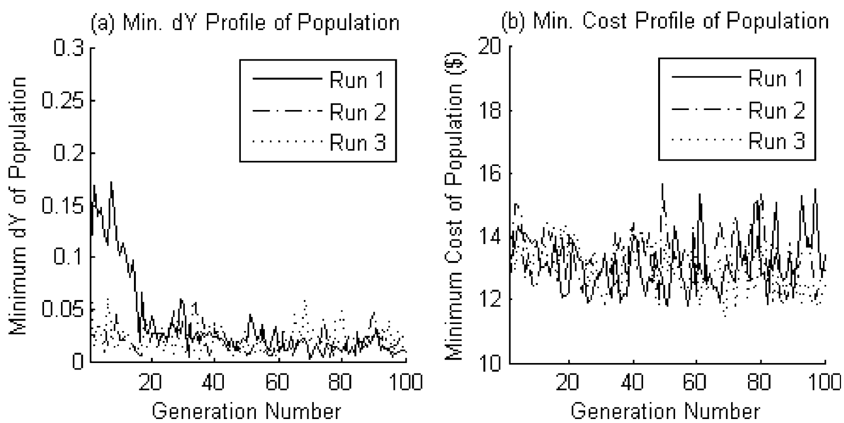

Fig. 7 Best functional behavior of particles. **a** Minimum dY profile of population. **b** Minimum cost profile of population

the algorithm. Assuming that the position of the best particle P_g does not change, a fly back to the particle's best position in the previous iteration (P_i) will ensure that the direction of the new velocity $\left(V_i^{k+1}\right)$ will point to the feasible space boundary but it will be closer to the global best P_g. Experimental results have shown that this mechanism is particularly suitable for mechanical design problems where optimal solutions lie on or near the feasible Pareto front [27]. The main guarantee of employing the fly back is that starting with an initial population of feasible solutions, each population generated in the subsequent iterations of the algorithm is also feasible.

Furthermore, in order to examine feasibility in a given population, there has to be a mechanism to compare two solutions in the repository. A simple yet elegant method based on NSGA-II was implemented in this work as outlined below [24]:

Definition 3. A solution S_i is said to dominate a solution S_j if:

1. S_i is feasible and S_j is infeasible.
2. S_i and S_j are both infeasible but S_i violates fewer design constraints.
3. S_i and S_j are both feasible but S_i dominates S_j (see Definition 1).

4.4 Performance measures

The performance of an optimizer can be judged along two categories: *efficiency*, which is a measure of computational effort to obtain optimal solutions (e.g., number of function evaluations, CPU time, etc.), and *effectiveness*, which measures the accuracy and convergence of the obtained solutions [4]. A crucial consideration regarding effectiveness of a multiobjective optimizer is that not only the optimizer should converge to Pareto-optimal solutions but it should also maintain diversity along the feasible region [4, 5, 16, 24].

Since no single performance measure can handle both requirements for making progress toward the Pareto-optimal front and for evaluating the spread or diversity of the obtained solutions, in this study, it was decided to utilize three performance metrics, which together better evaluate the overall effectiveness of a PSO-based multi-objective optimizer. Each of the utilized effectiveness

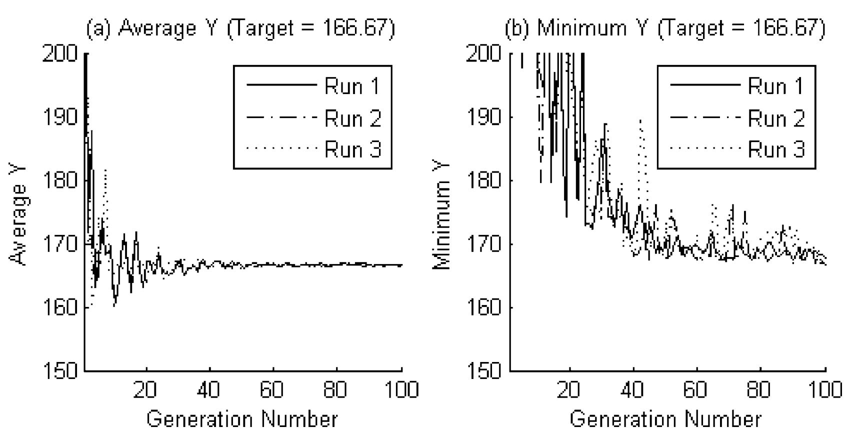

Fig. 8 Constraint profiles of the population. **a** Average Y (target=166.67). **b** Minimum Y (target=166.67)

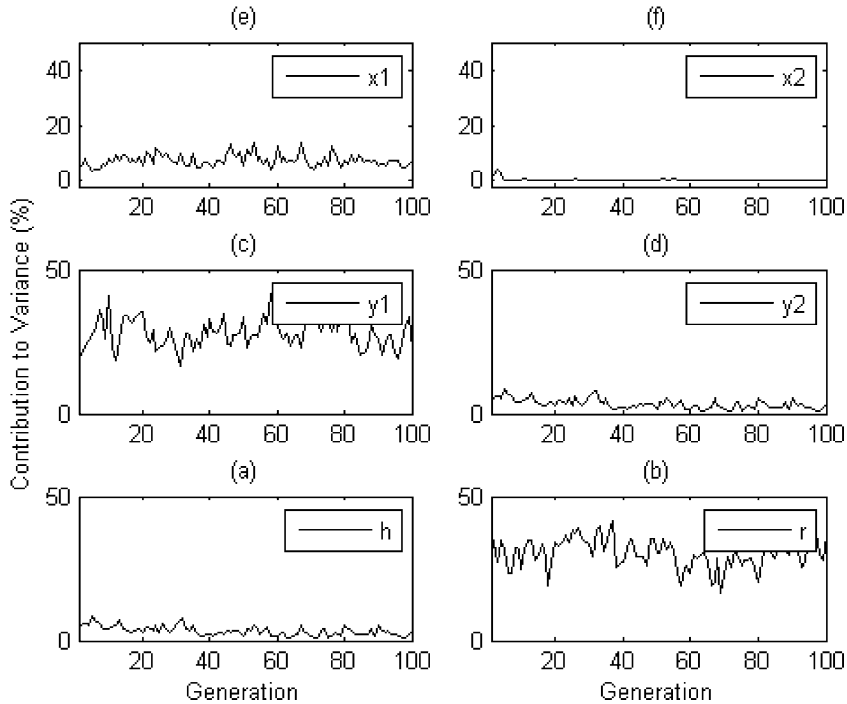

Fig. 9 **a**–**f** Design factor contributions to functional variance

metrics is described below. Please note that in the ensuing discussions, the set of discovered solutions is referred to as Q and P^* is the true Pareto-optimal set.

4.4.1 Generational distance

This performance metric measures the proximity of the obtained Pareto-optimal solutions Q to the true Pareto-optimal solutions P^* [4, 5, 16]:

$$\mathrm{GD} = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{1/p}}{|Q|} \tag{7}$$

The Euclidean distance-based metric d_i (p=2) is the phenotypic distance between each member i of the obtained set Q and the closest member in P^* to that member:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum \left(f_m^{(i)} - f_m^{*(k)} \right)^2} \tag{8}$$

Here, $f_m^{*(k)}$ is the value of the mth objective function of the kth in P^*. Observe that a successful swarm should discover solutions that produce a small value for GD.

Table 1 Two obtained solutions by the PSO (experiment 1)

Variable	Value	Tolerance	Sensitivity	Deviation	Factor contribution
x_1	22	0.2	−0.047	0.009	0.07%
x_2	300	0.4	−0.007	0.003	0.007%
y_1	67	0.3	0.860	0.258	52.8%
y_2	82	0.5	0.139	0.069	3.82%
h	42	0.1	1.001	0.100	7.93%
r	55	0.2	1.056	0.211	35.35%
				Y=166.67, dY=0.355, Cost=$12.71	
x_1	23	0.2	−0.006	0.001	0.001%
x_2	204	0.4	−0.001	0.0006	0.0003%
y_1	109	0.3	0.813	0.243	50.06%
y_2	110	0.5	0.186	0.093	7.32%
h	45	0.1	1.000	0.100	8.41%
r	57	0.2	1.008	0.201	34.19%
				Y=166.67, dY=0.344, *Cost*=$12.71	

Table 2 The best solution obtained by a traditional technique

Variable	Value	Tolerance	Sensitivity	Deviation	Factor contribution
x_1	70	0.2	−0.212	0.042	1.15%
x_2	338	0.4	−0.145	0.058	2.07%
y_1	65	0.3	0.932	0.279	49.24%
y_2	128	0.5	0.108	0.054	1.82%
h	35	0.1	1.025	0.102	6.29%
r	65	0.2	1.251	0.250	39.43%
				Y=166.67, dY=0.396, Cost=not available	

4.4.2 Spread

The spread metric (Δ) is used as an indicator of how well the solutions are distributed along the PSO's discovered Pareto front Q [4]:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} \left| d_i - \overline{d} \right|}{\sum_{m=1}^{M} d_m^e + |Q|\overline{d}} \tag{9}$$

where d_i is the Euclidean distance between a solution i and its nearest member in Q, and $\overline{d}$ defined as:

$$\overline{d} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} d_i \tag{10}$$

is the mean value of these distances. The parameter d_m^e is the distance between the extreme solutions in Q and P^* relating to an objective function m where there are a total of M response functions in the problem. An algorithm finding a smaller value of Δ (i.e., closer to zero) is better able to identify a diverse set of nondominated solutions.

4.4.3 Set coverage

This metric can be used as a measure of the relative spread of solutions between two sets of solutions A and B. The set coverage $C(A,B)$ calculates the proportion of solutions in B, which are weakly dominated by solutions in A [4, 5]:

$$C(A,B) = \frac{|\{b \in B | \exists a \in A : a \leq b\}|}{|B|} \tag{11}$$

If $C(A,B)$=1, all solutions in B are weakly dominated by A, whereas if $C(A,B)$=0, none of the solutions in B are weakly dominated by A. Thus, $C(Q, P^*)$ should always be 0.

5 Design of a stepped bar assembly

The optimum design of a stepped bar assembly, as shown in Fig. 3, has been previously attempted using traditional calculus-based sensitivity analysis [2] and continuous ant colony optimization [19].

5.1 The design formulation

Consider the assembly shown Fig. 3. The height of the center of the cylinder, Y, is the only critical dimension in this problem, and it is dependent on the value of design

Table 3 Two obtained solutions by the PSO (experiment 2)

Variable	Value	Tolerance	Sensitivity	Deviation	Factor contribution
x_1	30	0.09	−0.117	0.0106	0.82%
x_2	243	0.21	−0.025	0.0054	0.21%
y_1	93	0.07	0.820	0.0575	24.23%
y_2	123	0.11	0.179	0.0197	2.85%
h	9	0.07	1.010	0.0720	36.70%
r	58	0.06	1.153	0.0690	35.17%
				Y=165.24, dY=0.12, Cost=\$20.00	
x_1	20	0.09	−0.021	0.002	0.029%
x_2	206	0.21	−0.007	0.002	0.020%
y_1	88	0.07	0.736	0.052	21.71%
y_2	93	0.11	0.264	0.029	6.91%
h	11	0.07	1.000	0.070	40.13%
r	66	0.06	1.029	0.062	31.18%
				Y=166.56, dY=0.11, Cost=\$20.00	

Table 4 The best solution obtained by an ant colony optimizer

Variable	Value	Tolerance	Sensitivity	Deviation	Factor contribution
x_1	73	0.09	−0.185	0.017	N/A
x_2	336	0.21	−0.004	0.009	N/A
y_1	70	0.07	0.976	0.066	N/A
y_2	120	0.11	0.023	0.002	N/A
h	42	0.07	1.017	1.017	N/A
r	60	0.06	1.208	1.208	N/A
					Y=171.35, dY=0.12

variables x_1, x_2, y_1, y_2, h, and r. The smaller the deviation of Y is, the higher the quality of the product [2].

$$Y = y_1 + (r + h)\sec\theta + (r - x_1)\tan\theta \tag{12}$$

Where,

$$\tan\ \theta = (y_2 - y_1)/(x_2 - x_1) \tag{13}$$

$$\sec\ \theta = \sqrt{(1 + \tan^2\theta)} \tag{14}$$

The goal of the problem is to minimize the deviation of the critical dimension (dY) by changing the design variables while keeping the target value of the critical dimension Y at or as close as possible to 166.67.

Based on the discussions in Section 2, the variance of the critical dimension can be expressed as the sum of n component variances as attenuated by their respective sensitivity coefficients:

$$\mathrm{d}Y^2 = \sum_{i=1}^{n} (S_i \times \mathrm{d}x_i)^2 \tag{15}$$

The individual sensitivities can be calculated by taking the partial derivative of the functional relationship between the critical assembly dimension Y and the noncritical component dimensions x_1, x_2, y_1, y_2, h, and r. Hence,

$$S_1 = \frac{\partial Y}{\partial x_1} = \frac{(r+h)(y_2-y_1)^2}{\sec\ \theta\ (x_2-x_1)^3} - \frac{y_2-y_1}{x_2-x_1} + \frac{(r-x_1)(y_2-y_1)}{(x_2-x_1)^2} \tag{16}$$

$$S_2 = \frac{\partial Y}{\partial x_2} = -\frac{(r+h)(y_2-y_1)^2}{\sec\ \theta\ (x_2-x_1)^3} - \frac{(r-x_1)(y_2-y_1)}{(x_2-x_1)^2} \tag{17}$$

$$S_3 = \frac{\partial Y}{\partial y_1} = 1 - \frac{(r+h)(y_2-y_1)}{\sec\ \theta\ (x_2-x_1)^2} - \frac{(r-x_1)}{(x_2-x_1)} \tag{18}$$

$$S_4 = \frac{\partial Y}{\partial y_2} = \frac{(r+h)(y_2-y_1)}{\sec\ \theta\ (x_2-x_1)^2} - \frac{(r-x_1)}{(x_2-x_1)} \tag{19}$$

Table 5 An optimum solution obtained by the PSO (experiment 3)

Variable	Value	Tolerance	Sensitivity	Deviation	Factor contribution
x_1	58	0.09	−0.269	0.0242	3.49%
x_2	379	0.48	−0.001	0.0006	0.002%
y_1	89	0.07	0.995	0.0696	28.78%
y_2	176	0.46	0.004	0.0021	0.028%
h	38	0.07	1.036	0.0725	31.20%
r	40	0.06	1.307	0.0784	36.48%
				Y=164.70, dY=0.13, Cost=$17.80	

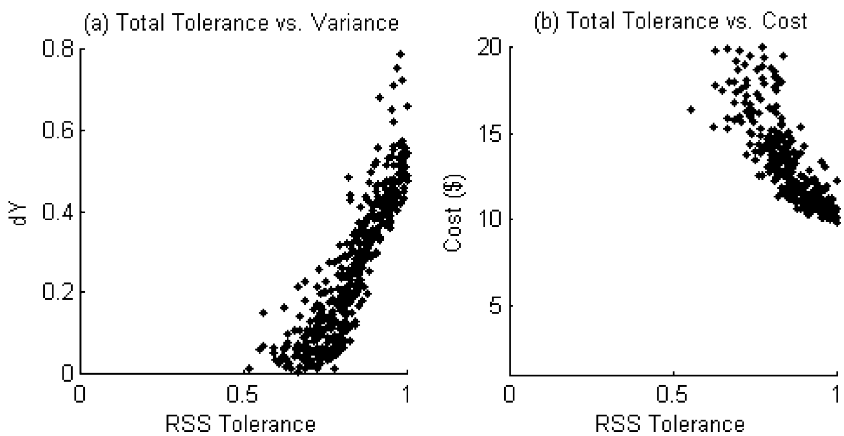

Fig. 10 Total tolerance vs. cost and variation. **a** Total tolerance vs. variance. **b** Total tolerance vs. cost

$$S_5 = \frac{\partial Y}{\partial h} = \sec\theta \tag{20}$$

$$S_6 = \frac{\partial Y}{\partial r} = \sec\theta + \tan\theta \tag{21}$$

5.2 Least cost-tolerance allocation

The final production cost of any mechanical assembly, including the one shown in Fig. 3, is significantly impacted by the specified tolerances on the dimensions of the manufactured parts. Tight tolerances can result in excessive process costs, while loose tolerances can lead to waste and assembly problems [1]. Figure 4 shows a typical manufacturing cost-tolerance curve.

Generally, there are two models to estimate how individual tolerances in a mechanical assembly stack up during the manufacturing process: the worst-case model (WCM) and the statistical RSS model [3].

The WCM computes the product's final tolerance (T_{tot}) by adding the individual tolerances (T_i) as $\sum T_i$. The main assumption here is that all the individual worst-case tolerance limits occur at the same time, and that can result in unnecessarily tight tolerances and a higher manufacturing cost.

In the RSS model, the individual tolerances are assumed to follow a normal distribution, and the final tolerance is calculated as:

$$T_{\mathrm{tot}} = \left(\sum T_i^2\right)^{0.5} \tag{22}$$

Clearly, the RSS approach allows for looser tolerances than the WCM method and, therefore, results in lower manufacturing costs.

A substantial amount of research has been carried out regarding optimal tolerance allocation using cost-tolerance functions, and various functions have been proposed to describe the cost-tolerance relationship [1, 2]. Assuming that for a component i the constant coefficient A_i ($) represents the fixed costs, such as tooling, setup, prior operations, etc., and that the B_{i} ($) term represents the cost of producing the

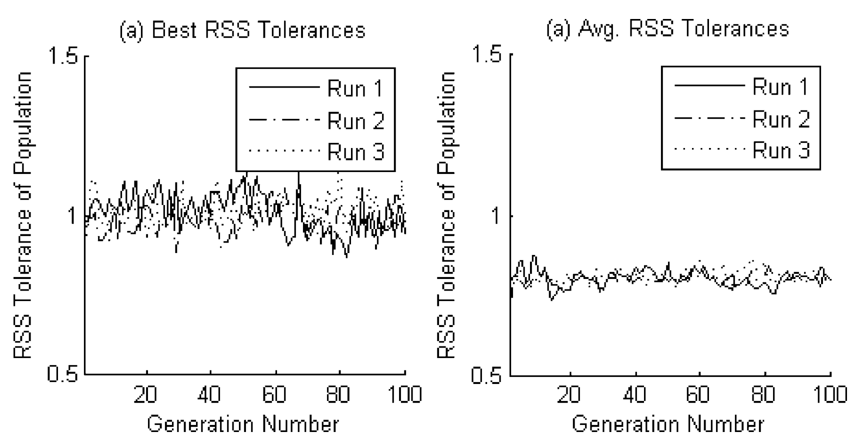

Fig. 11 RSS tolerance profiles of the population. **a** Best RSS tolerances. **b** Average RSS tolerance

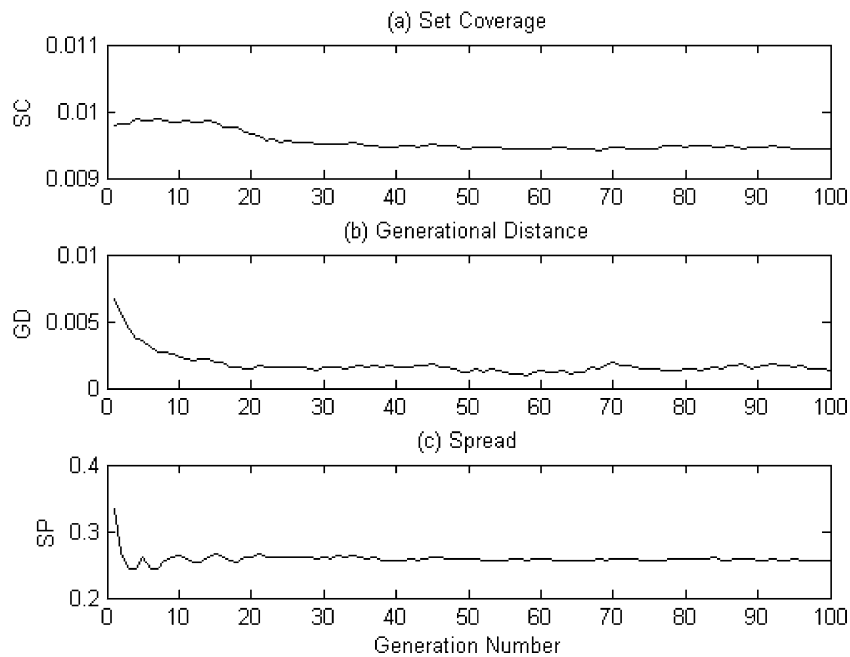

Fig. 12 Performance measures of **a** set coverage (*SC*), **b** generational distance (*GD*), and **c** spread (*SP*) for one run of the optimizer over 100 generations

component dimension to a specified tolerance T_i, the reciprocal power cost-tolerance function for the each component of the assembly can be calculated as [3]:

$$C_i = A_i + B_i / T_i^{K_i} \tag{23}$$

For some integer power, K_i is to be selected from standard data sets depending upon the required manufacturing process.

5.3 Multiobjective formulation of the problem

Previously, this problem was attempted by means of an ant colony optimizer, which aggregated the product variation dY and the minimum manufacturing cost C into a single term using the weighted aggregation method as follows [19]:

$$F(X) = \sum\nolimits_{i=1}^{m} w_i f_i(X) \tag{24}$$

Where f_{i} is the ith objective function, m is the number of objectives to be optimized, $w_i \geq 0$ with $\sum\nolimits_{i=1}^{m} w_i = 1$ is the weight assigned by designers to objective f_i representing its degree of importance, and X is the n-dimensional vector of design parameters. It has, however, been shown that the traditional weighted aggregated method suffers from many shortcomings, the most important of which is its assumption of convexity of the Pareto region, which is generally an invalid assumption in many real-world engineering design problems [5]. Furthermore, Pareto fronts are typically not uniform, that is, the obtained solutions are not uniformly distributed on the Pareto front given a set of evenly distributed weights [20].

The particle swarm optimization method utilized in this study was enhanced to perform a multimodal Pareto search for the combination of tolerances that minimize the total cost function for the assembly within the feasible region while also minimizing the assembly response variance dY and maximizing the total RSS tolerance. The multiobjective problem can formally be stated as shown below.

Optimize $F = [f_1, f_2, f_3]^T$, where the goal of optimization is to:

$$\text{Minimize} f_1 = \mathrm{d}Y = \sqrt{\left(\sum\nolimits_{i=1}^{n} (S_i \times \mathrm{d}x_i)2\right)} \tag{25}$$

Table 6 Summary of performance measures for 50 statistically independent trials

Performance statistics	Set coverage (SC)	Generational distance (GD)	Spread (SP)
Best	0.00979	0.00100	0.14327
Worst	0.05816	0.00671	0.34456
Mean	0.04556	0.00175	0.24281
Standard deviation	0.01463	0.00084	0.00857

Table 7 Comparison of optimization methods

Technique	Optimal design (x_1, x_2, y_1, y_2, r, h)	Y	dY	Cost ($)
Traditional calculus-based	(70±0.2, 338±0.4, 65±0.3, 128±0.5, 35±0.1, 65±0.2)	166.67	0.39	–
PSO[1]	(22±0.2, 300±0.4, 67±0.3, 82±0.5, 42±0.1, 55±0.2)	166.67	0.35	12.71
	(23±0.2, 204±0.4, 109±0.3, 110±0.5, 45±0.1, 57±0.2)	166.67	0.34	12.71
Ant colony optimizer	(73±0.09, 336±0.21, 70±0.07, 120±0.11, 42±0.07, 60±0.06)	171.35	0.12	–
PSO[2]	(30±0.09, 243±0.21, 93±0.07, 123±0.11, 9±0.07, 58±0.06)	165.24	0.12	20.00
	(20±0.09, 206±0.21, 88±0.07, 93±0.11, 11±0.07, 66±0.06)	166.57	0.11	20.00
PSO[3]	(58±0.09, 379±0.48, 89±0.07, 176±0.46, 38±0.07, 40±0.06)	164.70	0.13	17.80
PSO[4]	(53±0.42, 287±0.46, 53±0.29, 82±0.43, 26±0.23, 83±0.21)	166.70	0.42	11.00
	(51±0.36, 315±0.49, 42±0.13, 88±0.40, 64±0.10, 58±0.10)	166.61	0.21	13.83

$$\text{Minimize} f_2 = C_M(T) = \sum_{i=1}^{n} \left(A_i \times B_i / T_i^{K_i} \right) \tag{26}$$

$$\text{Minimize} f_3 = T_{\text{RSS}} = \left(\sum_{i=1}^{n} T_i^2 \right)^{0.5} \tag{27}$$

Subject to:

Y is to be on or as close as possible to the target value Y_{Target}=166.67
$x_2 > x_1$ and $y_2 > y_1$
$0.001 \le T_i \le 0.5$ for i=1,...,n, where n is the number of toleranced dimensions

Clearly, the three objectives f_1, f_2, and f_3 are conflicting in nature, and their optimization will require a thorough examination of the trade-off solutions along the Pareto front.

5.4 The experiments and results

In the experiments conducted, the enhanced PSO used 30 particles over 100 generations, maintained a repository of 500 particles (maximum size), and performed mutation at the rate of 0.5. The global best particle was selected from the top 10% sorted repository and replaced one of the nondominated solutions in the bottom 10% of the repository. To account for statistical fluctuations, the reported results were averaged over three statistically independent runs of the algorithm (the overall computation was very fast, and each experiment consisting of 100 generations typically took anywhere from 1 to 2 s).

Figure 5 depicts two Pareto fronts of the functional variance (dY) vs. the manufacturing cost (C_{M}) discovered by the optimizer after 100 generations with and without the crowding distance computations, respectively. In terms of solution diversity, it is clearly demonstrated in Fig. 5a that the utilization of the crowding distance computation allows a more evenly distributed exploration of the Pareto front.

A true indication of an optimizer's accuracy is the average behavior of it solutions as the swarm of particles explores the boundaries of the Pareto front across the feasible region. As shown in Fig. 6, as generations of solutions evolve, the average behavior (the statistical mean over three independent runs) of each of the population stays focused, gradually converging toward the trade-off optimum values both for product variance dY and the cost function C_{M}.

The profile of the best discovered (minimum-valued) solutions for the two objective functions are depicted in Fig. 7. A cursory examination of the range of Cost and dY functions shown in Fig. 5 illustrates the efficiency of convergence of the optimizer.

The optimization constraint required that the height of the center of the cylinder (Y) remains as close as possible to the target value of 166.67. The method of handling constraint violations via the fly-back mechanism to maintain feasibility was discussed in Section 4.3. Figure 8 shows the statistical means of three independent runs for the average and minimum product response Y of an entire swarm of particles.

The main goal of tolerance allocation is to re-distribute the "tolerance budget" within an assembly, systematically tightening tolerances on less expensive processes and loosening tolerances on costly processes, for a net reduction in cost [1]. This goal can very economically be accomplished by designers by simply moving the nominal values of the independent design parameters to less sensitive design regions. Figure 9 depicts the effects of the six design parameters (x_1, x_2, y_1, y_2, h, r) on the overall product variation calculated in one run of the algorithm.

A careful examination of factor contributions shown in Fig. 9 reveals that design variables x_1, y_1, and r have the largest leverage on the overall product variation. Therefore, by changing the dimensions of the independent variables and *not* the deviations on their manufacturing (tolerances), it will be possible to improve the design quality with little or no additional cost. This point is best demonstrated by the two solutions obtained from the first conducted experiment where the algorithm was allowed to modify the design nominal values but not the tolerances (see Table 1).

In the first experiment, the tolerances for the six design parameters were fixed at 0.2, 0.4, 0.3, 0.5, 0.1, and 0.2, respectively, so that the performance of the PSO algorithm can be directly compared to that of a traditional calculus-based technique as shown in Table 2 [2]. Clearly, the PSO's solutions are more desirable as they better represent the optimal portions of the trade-off design region. Further, the solutions reported in Table 1 are but two of dozens of optimal trade-off solutions available in the repository.

To further judge the true mettle of the PSO, its performance was compared to that of an ant colony optimizer that used the method of weighted aggregation to collapse its objective functions into a single global one [19]. In this second experiment, the tolerances were fixed at 0.09, 0.21, 0.07, 0.11, 0.07, and 0.06, respectively, so that the two approaches can be compared on a more equal footing. Table 3 demonstrates two solutions obtained by the PSO algorithm (Y_{Target}=166.67 in both experiments).

Table 4 shows the best nominal solutions obtained by the ant colony optimizer [19]. Note that factor contributions (percent) were not reported, and manufacturing cost comparisons could not be made simply due to the differences between the two approaches' method of cost computation (choice of A and B values in Eq. 26). Cost computations notwithstanding, the solutions obtained by the PSO have better optimized both the objective (minimization of dY) and the constraint (Y=Y_{Target}).

In another attempt to further reduce the incurred cost associated with design scenarios reported in Table 3 while maintaining on or near-target performance levels, a third experiment was conducted as follows. Design factor contributions to overall product variation (see Fig. 9) indicate that variables x_1, y_1, and r have the most leverage on variance. It was, therefore, decided to start the optimization task with the same set of tolerances used in the second experiment (0.09, 0.21, 0.07, 0.11, 0.07, 0.06) but this time have the optimizer widen the tolerances for the remaining parameters (x_2, y_2, and h) to which the assembly appeared less sensitive. Table 5 shows one of many optimal solutions that were obtained in this third experiment.

The trade-off solution shown in Table 5 results in an 11% reduction in the manufacturing cost compared to the solutions discovered in experiment 2. However, this reduction will cause an increase of 18% in the overall product variation dY. The designers can carefully examine the trade-off solutions and determine if they are acceptable for their specific needs.

The optimization tasks performed in the three experiments reported so far only relied on the minimization of cost (C_{M}) and variance (dY) as the tolerances where kept at fixed levels for comparative purposes. In the last experiment reported in this section, the total RSS tolerance (T_{RSS}) was allowed to be maximized (see Section 5.3) simultaneously as product variation and manufacturing cost were each minimized. Figure 10 depicts the Pareto fronts obtained in this manner.

As expected and shown in Fig. 10a and b, allocating wider tolerances to the components of the stepped bar assembly significantly decreased the manufacturing cost while adding to the overall product variation. The advantage of this type of analysis is that the designers can readily re-distribute tolerances by tightening tolerances on less costly processes while widening tolerances on more expensive ones. Figure 11 depicts how the algorithm to allocated wider tolerances as each generation of particles was undergone the various evolutionary operators. It must be noted that the widest-possible T_{RSS}=1.24, where all individual tolerances reach their highest levels.

To further assess the performance of the swarm optimizer, the performance measures of generational distance (GD), set coverage (SC), and spread (SP) were calculated as described in Section 4.4. To calculate the SC and GD metrics, a set of approximately 16,000 (a full-factorial experiment with five design levels per independent variable) uniformly spaced true Pareto-optimal solutions (P*) was calculated a priori and used in all the conducted experiments. Figure 12 depicts the performance of one run of the algorithm over 100 generations.

As noted earlier, it is desired for the optimizer to identify solution regions in which all the performance measures are as close as possible to zero. Figure 12 demonstrates that the algorithm is able to converge very quickly and identify optimal solutions starting around the generation number 20.

To better examine the quality of the nondominated solutions obtained by the optimizer, a total of 50 random experiments were conducted, and statistical information about the collected performance measures for the final (100th) generation of each independent trial is depicted in Table 6.

It can be seen from Table 6 that the optimizer is able to converge to the true Pareto-optimal set with good distribution of nondominated solutions. The low values of GD and SC metrics indicate that the optimizer was able to explore and exploit solution regions (Q) very closed to the true Pareto-optimal front P*.

Finally, Table 7 tabulates the results of four model experiments conducted in this study (PSO^1, PSO^2, PSO^3, and PSO^4) and compares them to the solutions obtained through other optimization methods (see Tables 1, 2, 3, 4, and 5).

The solutions obtained in the last experiment (PSO^4) were able to maintain the performance very close to Y_{Target} while minimizing the cost and allocating widest-possible tolerances. Designers can in turn decide whether or not the operational costs due to the projected product deviations are worth considering. The important point, therefore, is that the multiobjective particle swarm optimizer *does* provide the designers with the ability to have access to myriad Pareto-optimal solutions.

6 Conclusions

Experience has shown that the benefits of reducing operational costs of a product due to its functional variations far outweigh the benefits of selecting more expensive and efficient components. Designers rely on tolerance allocation, which is an effective design tool to reduce the overall cost of production while ensuring the product performance stays on target.

Various types of evolutionary algorithms have recently been applied to the task of tolerance allocation in engineering design problems. Given cost tolerances for individual components of an assembly, the optimizer can systematically search the design space for specific combinations of tolerances that minimize the cost while satisfying the overall design objectives and constraints. In this paper, an enhanced particle swam optimizer was successfully utilized to perform tolerance allocation in multiple-objective design of a stepped bar mechanical assembly. The optimizer was designed to search the Pareto-optimal solution space in one of two possible modes. First, it was allowed to optimize design objectives and constraints by modifying both the tolerances and the nominal values of the independent design variables. And second, in order to avoid expensive manufacturing costs associated with tighter tolerances, the optimizer was only allowed to modify the nominal component dimensions with the assumption that the component tolerances were fixed.

Comparative studies revealed that the swarm optimizer outperformed a traditional calculus-based method and ant colony optimizer in identifying optimal solutions. The optimizer's use of an effective fly-back mechanism along the boundaries of the Pareto front and the computation of crowding distances ensured that the repository of the nondominated particles represented the widest-possible coverage of the Pareto front. The advantages of the proposed methodology in allocating widest-possible tolerances for reducing manufacturing costs while minimizing functional variations and maintaining on or near-target performance levels were demonstrated in several numerical experiments.

Acknowledgments The author gratefully acknowledges the helpful comments and suggestions of the anonymous reviewers.

References

1. Chase KW (1999) Minimum-cost tolerance allocation. ADCATS Report No. 99-5. Department of Mechanical Engineering, Brigham Young University, Provo
2. Creveling CM (1997) Tolerance design, a handbook for developing optimal specifications. Wesley, Massachusetts
3. Altarazi S (2005) Operational tolerance allocation and machine assignment under process capability and product value constraints, Ph.D. Thesis, Wichita State University
4. Coello Coello C, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, 2nd edn. Springer, New York
5. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
6. Knowles J, Corne D, Deb K (eds) (2008) Multiobjective problem solving from nature: from concepts to applications. Springer, New York
7. 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
8. Ray T, Smith W (2006) A surrogate assisted parallel multiobjective evolutionary algorithm for robust engineering design. Eng Optim 38:997–1011 doi:10.1080/03052150600882538
9. Engelbrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, Chichester
10. Clerc M (2006) Particle swarm optimization. ISTE, California
11. Reyes-Sierra M, Coello C (2006) A survey of the state-of-the-art multi-objective particle swarm optimizers. Int J Comput Intell Res 2:287–308
12. Ochlak E, Forouraghi B (2006) A particle swarm algorithm for multiobjective design optimization. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 06) 765–772
13. Ho SL, Yang S, Ni G, Lo EW, Wong HC (2005) A particle swarm optimization-based method for multiobjective design optimizations. IEEE Trans Magn 41:1756–1759 doi:10.1109/TMAG.2005.846033
14. Liu D, Tan K, Goh C, Ho W (2007) A multiobjective memetic algorithm based on particle swarm optimization. IEEE Trans Syst Man Cybern B Cybern 37:585–605
15. Ray T, Liew KM (2002) A swarm metaphor for multiobjective design optimization. Eng Optim 34:141–153 doi:10.1080/03052150210915
16. Reddy MJ, Kumar DN (2007) An efficient multi-objective optimization based on swarm intelligence for engineering design. Eng Optim 39:49–68 doi:10.1080/03052150600930493
17. Shim M, Suh M (2002) Pareto-based continuous evolutionary algorithms for multiobjective optimization. Eng Comput 19:22–48 doi:10.1108/02644400210413649
18. Saltelli A, Tarantola S, Chan K (1999) A quantitative, model independent method for global sensitivity analysis of model output. Technometrics 41:39–56 doi:10.2307/1270993
19. 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 doi:10.1007/s00170-003-1846-0
20. Jin Y (2002) Effectiveness of weighted aggregation of objectives for evolutionary multiobjective optimization: methods, analysis and applications. Technical Report. Honda R&D Europe (D) GmbH, Germany
21. Feng CX, Kusiak A (1997) Robust tolerance design with the integer programming approach. J Manuf Sci Eng 119:603–610 doi:10.1115/1.2831193

22. Steiner G, Watzenig D (2003) Particle swarm optimization for worst case tolerance design. IEEE International Conference on Industrial Technology (ICIT ’03) 1:78–82
23. Haq AN, 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 doi:10.1007/s00170-004-2274-5
24. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. KanGAL Report 200001, Indian Institute of Technology, Kanpur, India
25. Li X (2004) Better spread and convergence: particle swarm multiobjective optimization using the maximum fitness function. Lect Notes Comput Sci 3102:117–128
26. Raquel CR, Naval PC (2005) An effective use of crowding distance in multiobjective particle swarm optimization. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO ‘05) 257–264
27. He S, Prempain E, Wu QH (2004) An improved particle swarm optimizer for mechanical design optimization problems. Eng Optim 36:585–605 doi:10.1080/03052150410001704854