Struct Multidisc Optim (2015) 51:173–182
DOI 10.1007/s00158-014-1125-8

RESEARCH PAPER

Multiobjective optimization using an aggregative gradient-based method

Kazuhiro Izui · Takayuki Yamada · Shinji Nishiwaki · Kazuto Tanaka

Received: 29 March 2013 / Revised: 18 April 2014 / Accepted: 3 June 2014 / Published online: 2 July 2014

Abstract A process of compromise that addresses conflicting objective functions such as performance and cost is often involved in real-world engineering design activities. If such conflicting relationships among objective functions exist in a multiobjective design optimization problem, no single solution can simultaneously minimize all objective functions, and the solutions of the optimization problem are obtained as a set of design alternatives called a Pareto optimal solution set. This paper proposes a new gradient-based multiobjective optimization method that incorporates a population-based aggregative strategy for obtaining a Pareto optimal solution set. In this method, the objective functions and constraints are evaluated at multiple points in the objective function space, and design variables at each point are updated using information aggregatively obtained from all other points. In the proposed method, a multiobjective optimization problem is converted to a single objective optimization problem using a weighting method, with weighting coefficients adaptively determined by solving a linear programming problem. A sequential approximate optimization-based technique is used to update the design variables, since it allows effective use of design sensitivities that can be easily obtained in many engineering optimization problems. Several numerical examples, including a structural optimization problem, are provided to illustrate the effectiveness and utility of the proposed method.

K. Izui (✉) · T. Yamada · S. Nishiwaki
Department of Mechanical Engineering and Science,
Kyoto University, Kyotodaigaku-Katsura C3, Nishikyo-ku,
Kyoto 615-8540, Japan
e-mail: izui@prec.kyoto-u.ac.jp

K. Tanaka
Department of Biomedical Engineering, Doshisha University,
1-3 Tatara-miyakodani, Kyotanabe 610-0394, Japan

Keywords Design optimization · Multiobjective optimization · Gradient-based optimization · Adaptive weighting coefficient

1 Introduction

The problem of finding suitable compromises among conflicting objective functions such as performance and cost often arises in real-world engineering design processes. When such conflicting relationships among objective functions are present in a multiobjective design optimization problem, no single solution can simultaneously minimize all objective functions, so the solutions of the optimization problem are obtained as a Pareto-optimal solution.

One of the very commonly used methods to obtain Pareto-optimal solutions is scalarization approaches in which a multiobjective optimization problem is converted to a single objective optimization problem. A weighting method (Zadeh 1963; Geoffrion 1968) is the most popular of such approaches, but the obtained solutions are greatly affected by the values of predetermined weighting coefficients, and the need to subsequently adjust these coefficients is a crucial problem. Goal Programming (Charnes and Cooper 1977), which uses target values of objective functions, and Physical Programming (Messac 1996), in which design engineers determine their preferences for objective functions, can be also classified as scalarization approaches. Such scalarization approaches provide only a single solution from a Pareto-optimal solution set, and the obtained solution is highly dependent on given parameter values.

Obtaining an entire set of Pareto-optimal solutions is another approach used to tackle the above difficulties in multiobjective optimization. This approach does not require predetermined preference parameter settings, and moreover,

the obtained Pareto-optimal solution set can aid the design process because it offers designers a clear picture of the trade-off relationships among the conflicting objective functions. Various multiobjective optimization methods have been proposed to obtain Pareto-optimal solution sets. In particular, multiobjective optimization methods based on metaheuristic techniques such as genetic algorithms (Goldberg 1989; Tamaki et al. 1996; Deb et al. 2002), and particle swarm optimization (Coello et al. 2004; Reyes-Sierra and Coello 2006; Kotinis 2011; Zhao and Suganthan 2011; Mahmoodabadi et al. 2012), have been extensively studied and used (Nourbakhsh et al. 2011; Montazeri-Gh et al. 2012; Sharma et al. 2014) in many applications .

Metaheuristic-based multiobjective optimization techniques use an aggregative searching strategy in which many points in the design space are simultaneously evaluated to obtain points that are used in the next iteration. An aggregative strategy is advantageous for global searches and can provide a Pareto-optimal solution set in a single optimization calculation. However, constraints cannot be explicitly handled in such methods, and the algorithms or objective functions must be modified to implement constraint handling (Cai and Wang 2006; Qu and Suganthan 2011; Cagnina et al. 2011). Furthermore, metaheuristic-based methods are inefficient when searching for fine-tuned solutions once a nearly global optimum is found, since the algorithms do not include design sensitivities. Metaheuristic techniques are also not well-suited for handling large-scale problems, that have many design variables, such as topology and shape optimization problems even in single objective optimization problems, although some extended schemes have been proposed (Wang and Tai 2005; Wang et al. 2006).

The use of gradient-based methods in multiobjective optimization problems has been discussed (Fliege and Svaiter 2000; Fliege et al. 2009; Qu et al. 2011) and the convergence properties of multiobjective optimization algorithms for unconstrained problems have been investigated. The normal boundary intersection method (Das and Dennis 1998), the normal constraint method (Messac et al. 2003), and the adaptive weighted-sum method (Kim and De Weck 2005) are methods where additional constraints are included in original optimization problems, to convert a multiobjective optimization problem to several single objective optimization problems and obtain Pareto optimal solutions. These methods can employ gradient-based optimization techniques that can be applied to constrained problems, and they can utilize design sensitivities in the optimization process.

The design sensitivity is the gradient of objective functions, or constraints, with respect to the design variables. Using design sensitivities in the optimization process is advantageous in large-scale constrained problems, since accurate design sensitivities can often be easily obtained in many engineering design optimization problems (Choi and Kim 2004) via semi-analytical computations, by conducting sensitivity analysis prior to the optimization process. This is routinely implemented when dealing with topology and shape optimization problems. In particular, the adjoint method is an effective technique for obtaining equations used to calculate design sensitivities. Once these equations for calculating the sensitivities are obtained through the sensitivity analysis, the computationally costly finite-difference method does not have to be used to carry out gradient calculations. However, these gradient-based methods may obtain local optima in multi-modal problems, which is typical of gradient-based methods because the obtained solutions are highly dependent on the selection of starting points. Therefore, a number of widely distributed starting points are required to obtain global Pareto-optimal solutions.

Therefore, this paper proposes a new gradient-based multiobjective optimization method that employs an aggregative strategy. In this method, the objective functions and constraints are evaluated at multiple points and the design variables at each point are updated using information aggregatively obtained from all other points in the objective function space.

2 Fundamental concept of the aggregative gradient-based multiobjective optimization method

Multiobjective optimization involves the simultaneous optimization of two or more conflicting objectives; a typical nonlinear multiobjective optimization problem can be written as

$$\text{minimize} \quad \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^{\mathrm{T}} \tag{1}$$

subject to:

$$\mathbf{g}(\mathbf{x}) \leq 0 \tag{2}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \tag{3}$$

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}, \tag{4}$$

where objective function vector $\mathbf{f}$ is a function of design variable vector $\mathbf{x}$, and $\mathbf{g}$ is an inequality constraint vector. $\mathbf{x}_L$ and $\mathbf{x}_U$ respectively denote the lower and upper bound of the design variables. Note that any equality constraint can be represented using a combination of inequality constraints. Therefore, equality constraints are simply ignored in the above formulation.

Figure 1 shows the conceptual scheme of the proposed method for a bi-objective optimization problem. In this scheme, objective functions and constraints at multiple points are simultaneously evaluated during an iteration, and every point is updated in an appropriate direction using a gradient-based search. For example, the black points

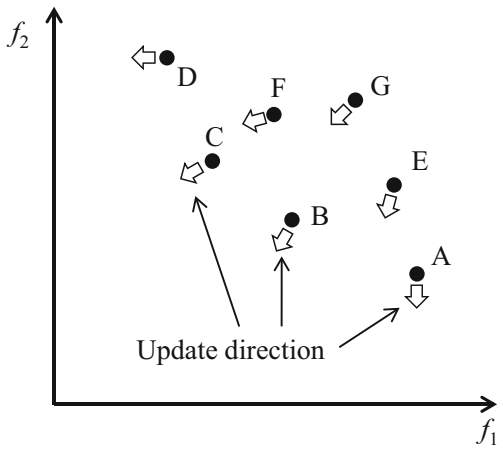

Fig. 1 Aggregative multiobjective optimization scheme of the proposed method

denoted A, B, ..., G in Fig. 1 indicate the positions of evaluated points in the objective function space, and the arrows indicate their update directions. To efficiently obtain a Pareto-optimal solution set, each point is moved toward the Pareto frontier point that is closest to its current position in the objective function space. That is, since point A has the smallest value of f_2 among all evaluated points, the gradient-based search result should move this point in the direction indicated by the arrow, to further minimize f_2. On the other hand, f_1 should be minimized for point D. Furthermore, for points B and C, both f_1 and f_2 should be minimized, to obtain Pareto solutions that represent compromises. Similarly, points E, F, and G should each move toward the corresponding closest Pareto frontier point, in the directions indicated by the white arrows in Fig. 1.

However, the true Pareto frontier is unknown prior to optimization calculations. Therefore, in this paper, the design variables are updated using a weighting method, and the search direction for each point is modified at every iteration by adaptively adjusting the weighting coefficients, based on each point's position relative to the other points in the objective function space.

3 Adaptive weighting coefficients

In the proposed method, weighing coefficients are determined using a Data Envelopment Analysis (DEA) technique. DEA is a tool originally used in the field of economics to evaluate the relative performance of decision-making units (DMUs) in multi-input and multi-output environments (Charnes et al. 1978). In this implementation, the performance values of a DMU for multiple criteria are converted, by solving a linear programming problem, to a single value, termed the efficiency value, which is then used to evaluate relative performance among multiple DMUs. An important feature of the DEA technique is that it can provide optimal weighting coefficients when the efficiency is calculated. In the proposed method, DEA is conducted for each point to obtain appropriate weighting coefficients.

In the case that all objective functions are to be minimized, the efficiency of the M-th point, θ^M, is calculated by solving the following linear programming problem.

$$\text{minimize} \quad \theta^M = \sum_{i=1}^{m} w_i^M f_i^M \quad \text{w.r.t.} \quad w_i^M \tag{5}$$

subject to:

$$\sum_{i=1}^{m} w_i^M f_i^k \geq 1 \quad (\text{for} \quad k = 1, 2, ..., K) \tag{6}$$

$$w_i^M \geq 0 \quad (\text{for} \quad i = 1, 2, ..., m), \tag{7}$$

where K is the total number of points, f_i^k is the k-th point's i-th objective function value, and w_i^M represents the weighting coefficients. If the M-th point is a non-dominated point among K points, θ^M becomes 1, and for dominated points, θ^M becomes larger than 1.

The weighting coefficients calculated by the DEA act to minimize θ^M, which is converted to a single objective function by the weighting method. Therefore, if the M-th point has a smaller value of f_1^M and a larger value of f_2^M than the other points in a bi-objective problem, the above linear programming problem returns a larger w_1^M value and a smaller w_2^M value, which increases the importance of the first objective function. The use of such calculated weighting coefficients when the design variables are subsequently updated is the main idea of the proposed method. The detailed procedure of the proposed multiobjective optimization method is explained in the following section.

4 Procedures

The aggregative gradient-based multiobjective optimization procedure is now described in detail. Figure 2 shows the flowchart of the procedure.

- **Step 1: Initialization**
  Generate initial design variables with random values for K points.
- **Step 2: Evaluate objective functions**
  Evaluate objective functions for all K points.
- **Step 3: Calculate weighting coefficients**
  Solve the linear programming problem shown in (5) through (7) for all points and obtain adaptive weighting coefficients w_i^M. M is then set to 1.

– **Step 4: Calculate design sensitivities**
Evaluate sensitivities of the objective functions and constraint functions for the M-th point.
– **Step 5: Update design variables**
Update the design variables of the M-th point, minimizing the weighted sum of the objective functions using weighting coefficients w_i^M. A Sequential Linear Programming (SLP)-based updating scheme is used in Step 5 since it can stably handle a large number of design variables using well-established linear programing solvers. That is, the single objective optimization problem, converted from the multiobjective optimization problem through the use of adaptive weighting coefficients, is linearly approximated. In this step, the following approximated linear programming problem is solved to update the design variables of the point.

$$\text{minimize} \quad f^M = \sum_{i=1}^{m} w_i^M \sum_{j=1}^{n} \frac{\partial f_i\left(\mathbf{x^M}\right)}{\partial x_j} x_j \quad \text{w.r.t.} \quad x_j \tag{8}$$

subject to:

$$g_s\left(\mathbf{x^M}\right) + \sum_{j=1}^{n} \frac{\partial g_s\left(\mathbf{x^M}\right)}{\partial x_j}\left(x_j - x_j^M\right) \le 0 \quad (\text{for} \quad s = 1, 2, ..., t) \tag{9}$$

$$\tilde{\mathbf{x}}_L \le \mathbf{x} \le \tilde{\mathbf{x}}_U, \tag{10}$$

where f^M is the weighted sum of the objective functions obtained in Step 3, and $\mathbf{x^M}$ is the design variable vector of the M-th point before updating. $\tilde{\mathbf{x}}_L$ and $\tilde{\mathbf{x}}_U$ are respectively the lower and upper bound for this linear programming problem considering the moving limit of the design variables. If $M = K$, the procedure then advances to Step 6 (Check termination condition), otherwise $M = M + 1$ and the procedure returns to Step 4.
– **Step 6: Check termination condition**
If the termination condition is satisfied, the procedure ends, otherwise it returns to Step 2.

Note that in the above procedure, two linear programming problems are solved for each point during a single iteration: 1) to determine weighting coefficient values, and 2) to update the design variables. The weighting coefficients for each point are therefore adaptively updated at every iteration. We note that although our method utilizes the concept of a weighting method, it does not require setting the weighting coefficients to predetermined values. The proposed method employs multiple starting points and some points are generated near Pareto frontier while some other points generated near local optima. Furthermore, appropriate weighting coefficient values are given to points which are close to the Pareto frontier so that global optima can be reached. This method, therefore, has a high likelihood to avoid local optima and obtain the true Pareto solutions.

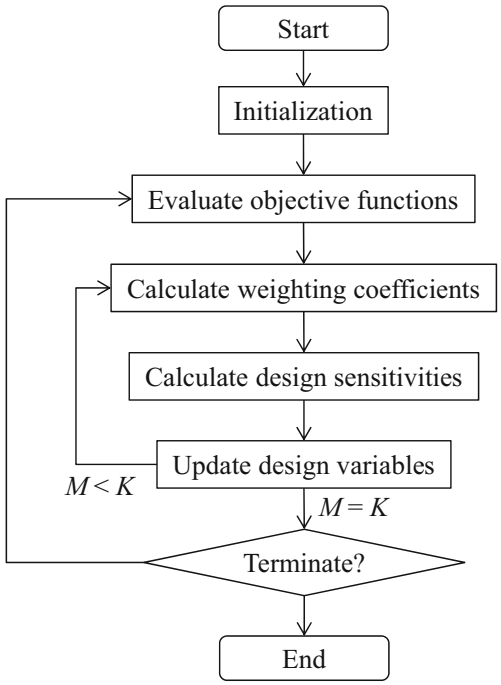

Fig. 2 Flowchart

5 Numerical examples

The proposed method is now applied to five numerical examples to demonstrate its performance. In the following examples in this section, values 5 % larger or smaller than those of the design variables were used as SLP move limits, if not mentioned. Computation time is used as a termination condition. That is to say, when a preassigned computation time is reached, the optimization process is halted. Equations to calculate design sensitivities for these examples can be analytically obtained before starting optimization processes. Sensitivities at each point were calculated using such equations, in the same manner employed in many conventional structural optimization methods.

In some of the the following examples, results of the proposed method are compared with those of a multiobjective genetic algorithm (MOGA). In the following calculation, a genetic algorithm code implemented based on the principles of Non-dominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al. 2002) was used for the MOGA calculation. In this calculation, each design variable is encoded using a real-coding scheme, and simulated binary crossover (SBX) and polynomial mutation were employed. The crossover and mutation rates were set to 1.0 and $1/l$, respectively, where l is the number of design variables. The distribution

indexes for crossover and mutation were set to 10 and 20, respectively. MOGA was conducted for the same computation time as that used for the proposed method. The various solutions are compared in figures, and the quantitative metrics for the hypervolume (Zitzler and Thiele 1999) and diversity (Deb et al. 2002) are also given.

Note that the use of computation time for the comparison of the optimization efficiency has an implicit limitation, since computation time significantly depends on how the particular method is implemented, as well as the complexity of the calculations carried out. In prior research, the number of function calls has been frequently been used as a basis for comparison. However, since the proposed method evaluates not only objective function values but also design sensitivities, the number of function calls cannot be used for the comparison. This is why we use the computation time, and we note that the following examples only provide simple assessment results from the standpoint of this single aspect.

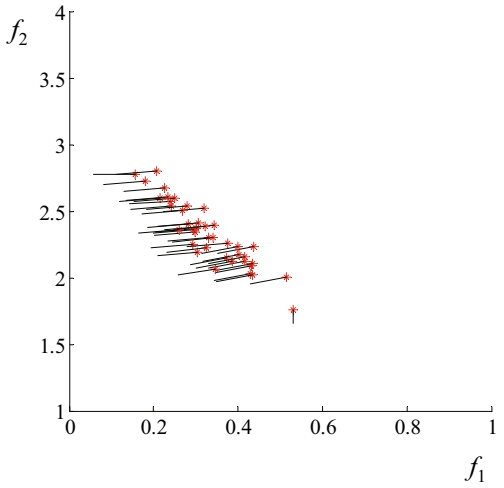

Fig. 3 Distribution of initial points and their weighting vectors in Example 1

5.1 Example 1

The first example is a two-objective optimization problem (Jin et al. 2001), formulated as below.

$$f_1 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 \tag{11}$$

$$f_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - 2)^2 \tag{12}$$

$$0 \le x_i \le 1 \quad (\text{for} \quad i = 1, 2, ..., n), \tag{13}$$

where n is the number of design variables, set here to 10. K denotes the total number of points and was set to 40 in this numerical example. The computation time was set to 60 second.

In Fig. 3, the eight-point stars indicate the initial point distribution in the objective function space. The design variables of the initial points were generated using a random uniform distribution function in the range of the upper and lower side constraint, [0, 1]. The black line segments attached to the initial point indicators represent weighting vectors at the first iteration, calculated by solving the linear programming problem in (5) through (7).

In Fig. 4, the "+" symbols indicate the solution points obtained after 66 iterations, and are connected by black lines to the corresponding initial points. This figure illustrates that the proposed method obtains well-distributed Pareto-optimal solutions, although our method does not guarantee this. Note that, in order to conduct optimization for 66 iterations, $66 \times 40 = 2,640$ function calls, and the same number of design sensitivity calculations, are required.

Figure 5 shows a comparison of the non-dominated solutions obtained for this problem when using two different methods: the proposed method, and MOGA, with the same termination criterion of computational time used in both cases. In the MOGA calculation, the population size was set to 40 and computation was conducted for 465 generations. This figure shows all non-dominated solutions obtained during the optimization process, whereas Fig. 4 only shows the position of all points at the very last iteration. Table 1 provides the hypervolume and diversity metric values for both methods. Figure 5 and Table 1 show that the solution distribution obtained by the proposed method is almost equivalent to that of the MOGA. However, the proposed methodfs solutions are slightly better than those of the MOGA, since the proposed method can conduct local searches near the Pareto frontier.

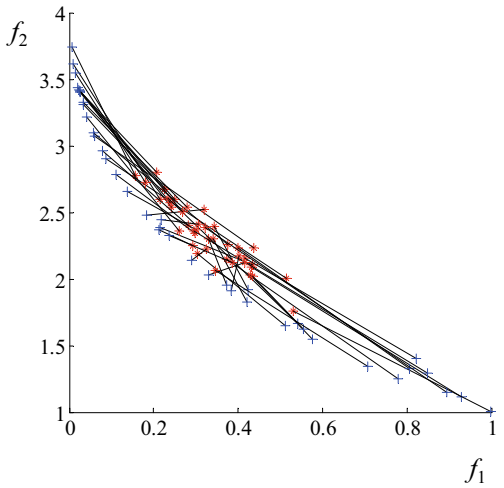

Fig. 4 Obtained solutions in Example 1

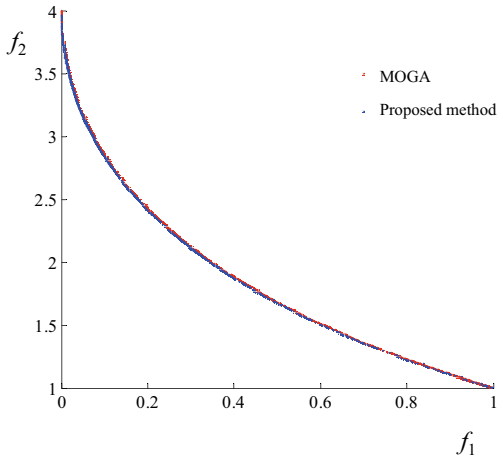

Fig. 5 Comparison with MOGA

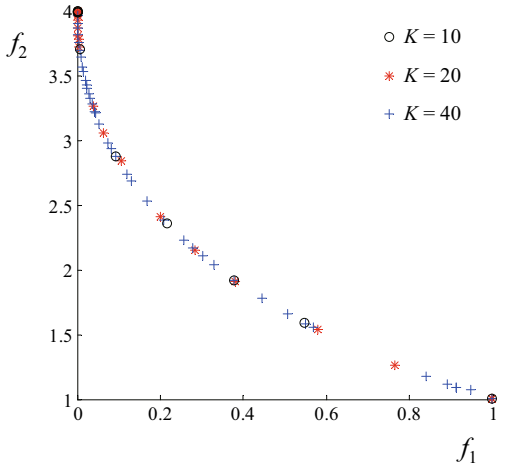

Fig. 6 Effect of the number of starting points

Figure 6 shows the position of points at the final iteration of the optimization process when different numbers of K but the same termination criterion are used. This figure illustrates that the proposed method works well for different numbers of starting points, but a larger number of points is preferable for obtaining closely distributed solutions.

5.2 Example 2

Next, the proposed method was applied to a test function (Preuss et al. 2006). The problem is stated as follows.

$$f_1 = x_1^4 + x_2^4 - x_1^2 + x_2^2 - 10x_1x_2 + 0.25x_1 + 20 \quad (14)$$

$$f_2 = (x_1 - 1)^2 + x_2^2 \quad (15)$$

$$-2 \le x_1, x_2 \le 2 \quad (16)$$

In this calculation, the computation time was set to 15 seconds, and the optimization process was conducted until the calculation reached the 26th iteration. Therefore, $40 \times 26 = 1,040$ function calls and the same number of design sensitivity calculations were conducted.

Figure 7 shows the results for this problem, where K was set to 40. The eight-point stars again indicate the initial points and the "+" symbols represent the obtained solutions. The circled "+" marks indicate dominated solutions that represent local optima for this problem. This result illustrates that the proposed method provided true Pareto-optimal solutions, although several points are stacked at local optima. The proposed method succeeded in avoiding local optima in this example because it utilizes multiple starting points, and appropriate weighting coefficient values are given to points close to the Pareto frontier, so that global optima can be reached.

Figure 8 and Table 2 show a comparison of the non-dominated solutions obtained using the proposed method and MOGA, in which a population size of 40 was used and computation was conducted for 182 generations. MOGA is suitable for this kind of problem since it has only two variables and includes a large discontinuity in the Pareto frontier line of the objective function space. However, the proposed method also provided good solutions.

Figure 9 shows the effect of selecting the number of starting points, and this figure also illustrates that good solutions are successfully obtained regardless of the number of starting points.

Table 1 Comparison of two optimization methods in Example 1

	Proposed method	MOGA
Hypervolume	0.7209	0.7142
Diversity	0.7745	0.8484

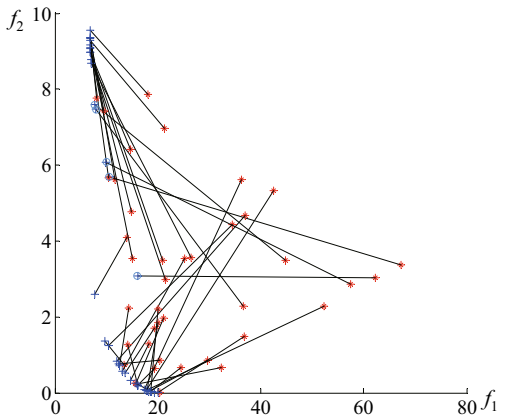

Fig. 7 Obtained solutions in Example 2

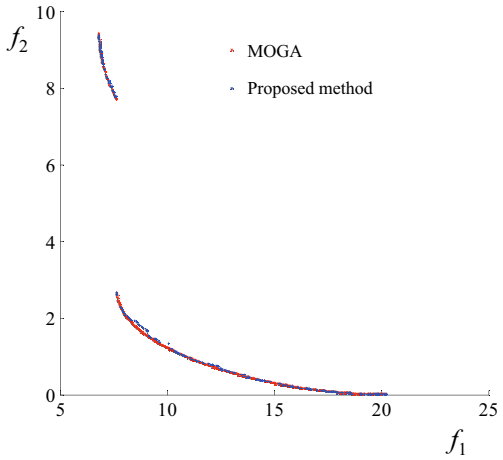

Fig. 8 Comparison with MOGA in Example 2

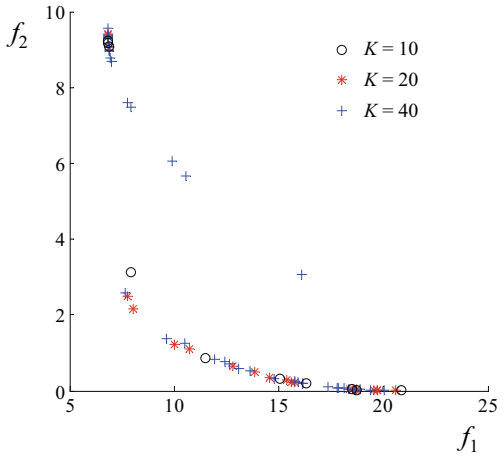

Fig. 9 Effect of the number of starting points in Example 2

5.3 Example 3

Next, the proposed method is applied to a three-objective optimization problem (Laumanns et al. 2002). The problem formulation is as follows.

$$f_1 = 3 - (1 + x_3)\cos(x_1\pi/2)\cos(x_2\pi/2) \tag{17}$$

$$f_2 = 3 - (1 + x_3)\cos(x_1\pi/2)\sin(x_2\pi/2) \tag{18}$$

$$f_3 = 3 - (1 + x_3)\cos(x_1\pi/2)\sin(x_1\pi/2) \tag{19}$$

$$0 \le x_1, x_2, x_3 \le 1 \tag{20}$$

In this problem, the number of starting points K was set to 60, the moving limit for the SLP search was set to 0.1, and the computation time was set to 15 second.

Figure 10 shows all the non-dominated solutions obtained during the optimization process for 28 iterations. This figure illustrates that the proposed method can effectively provide well-distributed solutions.

5.4 Example 4

For the fourth example, the 105-bar truss design optimization problem shown in Fig. 11 was solved. In this problem, K was set to 20, and the computation time was set to 1,500 seconds. In this example, a 1.0×10^5 N force in the $-Y$ direction was applied at node A, at the lower right-hand edge of the structure. The design variables were the cross-sectional area of each truss element, with an upper and lower constraint of $[1.0 \times 10^{-3}, 1.0 \times 10^{-6}]$. The objective functions in this problem were the displacement at node A, denoted d_A, and the total volume of the structure, V, and both were to be minimized under compressive and tensile stress constraints. The Young's modulus for each truss element and the allowable stress, σ_a, was set to 210 GPa and 270 MPa, respectively. This problem was formulated as follows.

$$f_1 = V \tag{21}$$

$$f_2 = d_A \tag{22}$$

subject to:

$$1.0 \times 10^{-6} \le x_j \le 1.0 \times 10^{-3} \tag{23}$$

$$-\sigma_a \le \sigma_j \le \sigma_a \quad (j = 1, 2, \ldots, 105), \tag{24}$$

where σ_j is the stress at the j-th truss member.

Figure 12 shows a comparison of the non-dominated solutions obtained for this problem when using two different methods, i.e., the proposed method and the MOGA. In this MOGA calculation, the population size was set to 100. This figure illustrates that the proposed method obtained

Table 2 Comparison of two optimization methods in Example 2

	Proposed method	MOGA
Hypervolume	0.8770	0.8828
Diversity	0.5771	0.6286

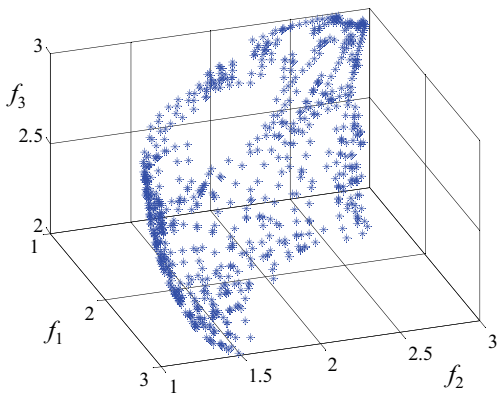

Fig. 10 Non-dominated solutions in Example 3

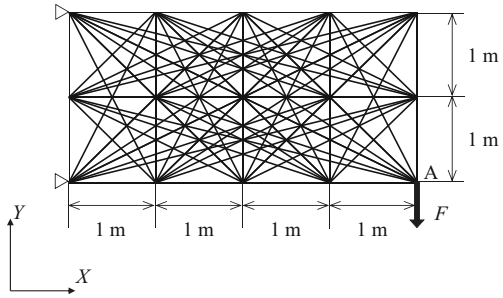

Fig. 11 Configuration of 105-bar truss problem in Example 4

better solutions than those of the MOGA with NSGA-II method, and that the non-dominated solutions were widely distributed. Figure 13 shows the low f_2 range of the non-dominated solutions for this problem, to clearly illustrate the distribution of the solutions obtained by the MOGA method.

As shown in these figures, the MOGA method did not provide a wide range of solutions, and the objective function values were slightly poorer compared with those of the proposed method, especially in the range where the total volume f_1 was low, although many useful solutions were obtained in the range where f_1 was large. On the other hand, the proposed method obtained better solutions in the lower total volume range because stress constraints are explicitly handled, which enhances the searching efficiency in this range.

Table 3 presents comparative data for the proposed and MOGA methods. The number of iterations used in the proposed method was significantly smaller than that in the MOGA method, in which the term "generation" is used rather than iteration. The proposed method uses a smaller number of iterations than the MOGA because two linear programming problems must be solved for each point,

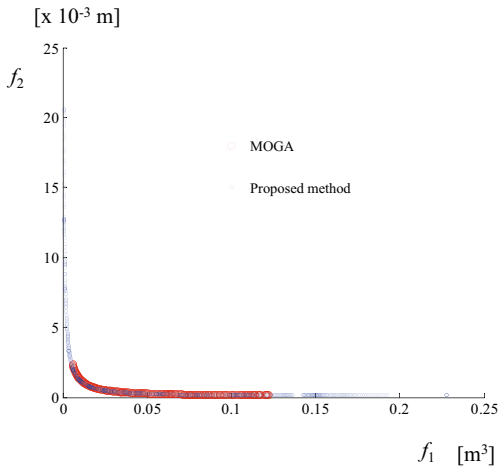

Fig. 12 Non-dominated solutions obtained in Example 4

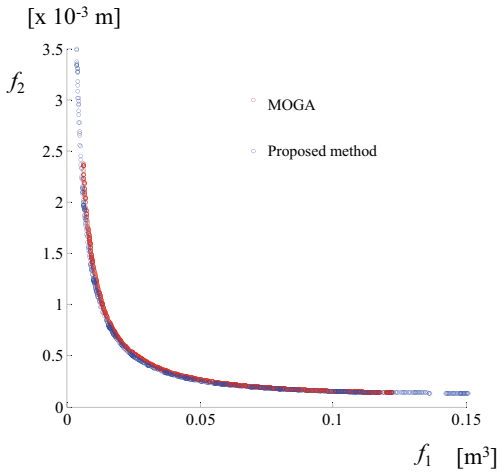

Fig. 13 Non-dominated solutions in Example 4 (magnified)

during every iteration, and sensitivity calculations are more computationally costly than plain function calls.

5.5 Example 5

In this last example, the proposed method is used to solve the topology optimization problem illustrated in Fig. 14, which is called Messerschmitt-Bölkow-Blohm (MBB) beam problem. The objective functions are the mean compliance and total volume, and both are minimized. The density method is used in this problem and the design domain is discretized into 60×40 elements. This problem is therefore a large-scale problem with 2,400 design variables, and the range of each design variable was set to [0, 1]. We note that a metaheuristics-based multiobjective optimization method could not be applied to such a large-scale problem.

The multiobjective topology optimization is formulated as follows.

$$f_1 = \mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u} \tag{25}$$

$$f_2 = V \tag{26}$$

Table 3 Comparison of two optimization methods in Example 4

	Proposed method	MOGA
Computational time	1,503.5 [sec]	1,501.2 [sec]
Number of iterations	182	1,231
Number of non-dominated solutions	840	1,711
Function calls	3,640	123,100
Sensitivity calculation	3,640	-
Hypervolume	0.9889	0.5077
Diversity	1.4821	1.0205

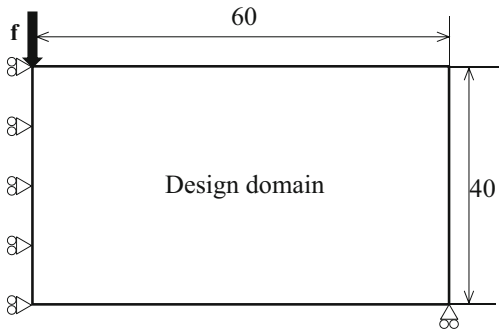

Fig. 14 Topology optimization problem

subject to:

$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \tag{27}$$

$$\mathbf{K} + u = \mathbf{f}, \tag{28}$$

where $\mathbf{u}$ and $\mathbf{f}$ are respectively the displacement and force vector, $\mathbf{K}$ is the stiffness matrix, and V denotes the total volume.

The mean compliance problem is self-adjoint, and the design sensitivity can be calculated using the following equation.

$$\frac{\partial f_1}{\partial x_i} = -\mathbf{u}^{\mathrm{T}} \frac{\partial \mathbf{K}}{\partial x_i} \mathbf{u} \tag{29}$$

where x_i is the design variable for i-th element. The sensitivity of the total volume with respect to every design variable is 1 since the all elements are square and of equal size in this problem.

In this calculation, the computation time was set to 2000 second, and the optimization process executed 169 iterations; therefore, $60 \times 169 = 10,140$ function calls and the same number of calculation of design sensitivities were conducted. Figure 15 shows the non-dominated solutions obtained by the proposed method when K was set to 60. Four selected points, A–D in this figure, have corresponding configurations shown in Fig. 16. The illustrated configurations demonstrate that the proposed method can effectively obtain non-dominated solutions for topology optimization problems that include a large number of design variables, since design sensitivities are used when updating the design variables. We note that conventional multiobjective optimization methods employing metaheuristic techniques, classified as direct methods since design sensitivities are not used in the optimization process, are almost always unsuitable for such large-scale problems.

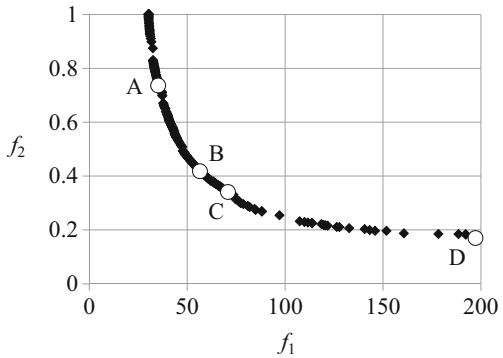

Fig. 15 Non-dominated solutions in Example 5

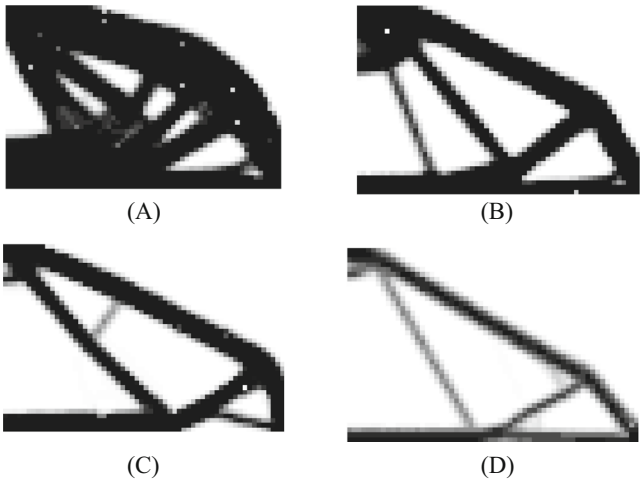

Fig. 16 Examples of obtained solutions in Example 5

6 Conclusions

This paper proposed a new gradient-based multiobjective optimization method to obtain Pareto-optimal solutions. The proposed method uses a weighting method to convert a multiobjective optimization problem to a single objective optimization problem, and a linear programming problem is solved to determine adaptive weighting coefficients for each point while considering the point's position relative to all other points in the objective function space. The converted single objective optimization problem is linearly approximated, and the design variables are updated by a linear programming technique. The proposed method was applied to five examples, three test functions and two structural optimization problems, and the results obtained in all examples demonstrated its effectiveness. The proposed method is based on the weighting method, which makes it unsuitable for non-convex problems. In future work, we hope to extend the proposed method for application to non-convex problems.

References

Cagnina LC, Esquivel SC, Coello CAC (2011) Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. Eng Optim 43(8):843–866

Cai Z, Wang Y (2006) A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Trans Evol Comput 10(6):658–675

Charnes A, Cooper WW (1977) Goal programming and multiple objective optimizations: Part 1. Eur J Oper Res 1(1):39–54

Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units. Eur J Oper Res 2(6):429–444

Choi KK, Kim NH (2004) Structural sensitivity analysis and optimization 1: linear systems. Springer

Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279

Das I, Dennis JE (1998) Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657

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

Fliege J, Drummond LG, Svaiter BF (2009) Newton's method for multiobjective optimization. SIAM J Optim 20(2): 602–626

Fliege J, Svaiter BF (2000) Steepest descent methods for multicriteria optimization. Math Methods Oper Res 51(3):479–494

Geoffrion AM (1968) Proper efficiency and the theory of vector maximization. J Math Anal Appl 22(3):618–630

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional

Jin Y, Olhofer M, Sendhoff B (2001) Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how. In: Proceedings of the genetic and evolutionary computation conference (GECCO2001), pp. 1042–1049

Kim IY, De Weck O (2005) Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Struct Multidiscip Optim 29(2):149–158

Kotinis M (2011) Implementing co-evolution and parallelization in a multi-objective particle swarm optimizer. Eng Optim 43(6):635–656

Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multi-objective optimization: evolutionary computation 10(3):1–21

Mahmoodabadi M, Bagheri A, Nariman-zadeh N, Jamali A (2012) A new optimization algorithm based on a combination of particle swarm optimization, convergence and divergence operators for single-objective and multiobjective problems. Eng Optim 44(10):1167–1186

Messac A (1996) Physical programming: effective optimization for computational design. AIAA j 34(1):149–158

Messac A, Ismail-Yahaya A, Mattson CA (2003) The normalized normal constraint method for generating the pareto frontier. Struct Multidiscip Optim 25(2):86–98

Montazeri-Gh M, Jafari S, Ilkhani M (2012) Application of particle swarm optimization in gas turbine engine fuel controller gain tuning. Eng Optim 44(2):225–240

Nourbakhsh A, Safikhani H, Derakhshan S (2011) The comparison of multi-objective particle swarm optimization and NSGA II algorithm: applications in centrifugal pumps. Eng Optim 43(10):1095–1113

Preuss M, Naujoks B, Rudolph G (2006) Pareto set and EMOA behavior for simple multimodal multiobjective functions. Parallel Probl Solving Nat-PPSN IX:513–522

Qu S, Goh M, Chan FT (2011) Quasi-newton methods for solving multiobjective optimization. Oper Res Lett 39(5):397–399

Qu BY, Suganthan PN (2011) Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. Eng Optim 43(4):403–416

Reyes-Sierra M, Coello CC (2006) Multi-objective particle swarm optimizers: A survey of the state-of-the-art. Int J Comput Intell Res 2(3):287–308

Sharma D, Deb K, Kishore N (2014) Customized evolutionary optimization procedure for generating minimum weight compliant mechanisms. Eng Optim 46(1):39–60

Tamaki H, Kita H, Kobayashi S (1996) Multi-objective optimization by genetic algorithms: a review. In: Proceedings of 1996 IEEE international conference on evolutionary computation

Wang SY, Tai K (2005) Structural topology design optimization using genetic algorithms with a bit-array representation. Comput Methods Appl Mech Eng 194(36-38):3749–3770

Wang SY, Tai K, Wang MY (2006) An enhanced genetic algorithm for structural topology optimization. int J Numer Methods Eng 65(1):18–44

Zadeh L (1963) Optimality and non-scalar-valued performance criteria. IEEE Trans Autom Control 8(1):59–60

Zhao SZ, Suganthan P (2011) Two-lbests based multiobjective particle swarm optimizer. Eng Optim 43(1):1–17

Zitzler E, Thiele L (1999) Multiobjective evlolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271