Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)

An interactive fuzzy satisficing method through particle swarm optimization for multiobjective nonlinear programming problems

Takeshi Matsui
Graduate School of Engineering
Hiroshima University, Japan
Email: matsui@msl.sys.hiroshima-u.ac.jp

Masatoshi Sakawa
Graduate School of Engineering
Hiroshima University, Japan
Email: sakawa@msl.sys.hiroshima-u.ac.jp

Kosuke Kato
Graduate School of Engineering
Hiroshima University, Japan
Email: kato@msl.sys.hiroshima-u.ac.jp

Takeshi Uno
Graduate School of Engineering
Hiroshima University, Japan
Email: uno@msl.sys.hiroshima-u.ac.jp

Koichi Tamada
Graduate School of Engineering
Hiroshima University, Japan
Email: tamada@msl.sys.hiroshima-u.ac.jp

Abstract— **Particle swarm optimization (PSO) was proposed by Kennedy et al. as a general approximate solution method for nonlinear programming problems. Its efficiency has been shown, but there have been left some shortcomings of the method. Thus, the authors proposed a revised PSO (rPSO) method incorporating the homomorphous mapping and the multiple stretching in order to cope with these shortcomings. In this paper, we construct an interactive fuzzy satisficing method for multiobjective nonlinear programming problems based on the rPSO. Furthermore, in order to obtain better solutions in consideration of the property of multiobjective programming problems, we incorporate the direction to nondominated solutions into the rPSO. Finally, we show the efficiency of the proposed method by applying it to numerical examples.**

I. Introduction

In general nonlinear programming problems to find a solution which minimizes an objective function under given constraints, one whose objective function and constraint region are convex is called a convex programming problem. For such convex programming problems, there have been proposed many efficient solution method as the successive quadratic programming method and the general gradient method. Unfortunately, there have not been proposed any decisive solution method for nonconvex programming problems. As practical solution methods, meta-heuristic optimization methods as the simulated annealing method and the genetic algorithm have been proposed. In recent years, however, more speedy and more accurate optimization methods have been desired because the size of actual problems has been increasing.

As a new optimization method, particle swarm optimization (PSO) was proposed by Kennedy et al. [2]. PSO is a search method simulating the social behavior that each individual in the population acts by using both the knowledge owned by it and that owned by the population, and they search better points by constituting the population. The authors proposed a revised PSO (rPSO) by incorporating the homomorphous mapping and the multiple stretching technique in order to deal with shortcomings of the original PSO as the concentration to local solution and the inapplicability of constrained problems [5].

In recent years, with the diversification of social requirements, the demand for the programs with multiple objective functions, which may be conflicting with each other, rather than a single-objective function, has been increasing (e.g. maximizing the total profit and minimizing the amount of pollution in a production planning). Since there does not always exist a complete optimal solution which optimizes all objectives simultaneously for multiobjective programming problems, the Pareto optimal solution or non-inferior solution, is defined, where a solution is Pareto optimal if any improvement of one objective function can be achieved only at the expense of at least one of the other objective functions. For such multiobjective optimization problems, fuzzy programming approaches (e.g. H.-J. Zimmermann [12], H. Rommelfanger [7]), considering the imprecise nature of the DM's judgments in multiobjective optimization problems, seem to be very applicable and promising. In the application of the fuzzy set theory into multiobjective linear programming problems started by Zimmermann [11], it has been implicitly assumed that the fuzzy decision or the minimum-operator of Bellman and Zadeh [1] is the proper representation of the DM's fuzzy preferences. Thereby, M. Sakawa et al. have proposed interactive fuzzy satisficing methods to derive satisficing solutions for the decision maker along with checking the local preference of the decision maker through interactions for various multiobjective programming problems [9].

In this research, focusing on multiobjective nonlinear programming problems, we attempt to derive satisficing solutions through the interactive fuzzy satificing method. Since problems solved in the interactive fuzzy satificing method for multiobjective nonlinear programming problems are nonlinear programming problems, we adopt rPSO [5] as solution methods to them. In particular, we consider measures to improve the performance of rPSO in applying it to solving the augmented minimax problem.

II. Multiobjective nonlinear programming problems

In this research, we consider multiobjective nonlinear programming problem as follows:

$$\left.\begin{array}{rl} \text{minimize} & f_l(\boldsymbol{x}),\ l=1,2,\ldots,k \\ \text{subject to} & g_i(\boldsymbol{x}) \le 0,\ i=1,2,\ldots,m \\ & l_j \le x_j \le u_j,\ j=1,2,\ldots,n \\ & \boldsymbol{x}=(x_1,x_2,\ldots,x_n)^T \in R^n \end{array}\right\} \quad (1)$$

where $f_l(\cdot)$, $g_i(\cdot)$ are linear or nonlinear functions, l_j and u_j are the lower limit and the upper limit of each decision variable x_j. In addition, we the feasible region of (1) by X.

III. An interactive fuzzy satisficing method

In order to consider the imprecise nature of the decision maker's judgments for each objective function in (1), if we introduce the fuzzy goals such as "$f_l(\boldsymbol{x})$ should be substantially less than or equal to a certain value", (1) can be rewritten as:

$$\underset{\boldsymbol{x}\in X}{\text{maximize}}\ (\mu_1(f_1(\boldsymbol{x})),\ldots,\mu_k(f_k(\boldsymbol{x}))) \quad (2)$$

where $\mu_l(\cdot)$ is the membership function to quantify the fuzzy goal for the l th objective function in (1).

Since (2) is regarded as a fuzzy multiobjective decision making problem, there rarely exist a complete optimal solution that simultaneously optimizes all objective functions. As a reasonable solution concept for the fuzzy multiobjective decision making problem, M. Sakawa et al. defined M-Pareto optimality on the basis of membership function values by directly extending the Pareto optimality in the ordinary multiobjective programming problem [8]. In the interactive fuzzy satisficing method, in order to generate a candidate for the satisficing solution which is also M-Pareto optimal, the decision maker is asked to specify the aspiration levels of achievement for all membership functions, called the reference membership levels [8]. For the decision maker's reference membership levels $\bar{\mu}_l$, $l=1,\ldots,k$, the corresponding M-Pareto optimal solution, which is nearest to the requirements in the minimax sense or better than that if the reference membership levels are attainable, is obtained by solving the following augmented minimax problem (3).

$$\begin{array}{ll} \underset{\boldsymbol{x}\in X}{\text{minimize}} & \max\limits_{l=1,\ldots,k}\{(\bar{\mu}_l-\mu_l(f_l(\boldsymbol{x}))) \\ & \qquad +\rho\sum\limits_{i=1}^{k}(\bar{\mu}_i-\mu_i(f_i(\boldsymbol{x})))\} \end{array} \quad (3)$$

where ρ is a sufficiently small positive number.

We can now construct the interactive algorithm in order to derive the satisficing solution for the decision maker from the M-Pareto optimal solution set. The procedure of an interactive fuzzy satisficing method is summarized as follows.

Step 1: Under a given constraint, minimal value and maximum one of each objective function are calculated by solving following problems.

$$\underset{\boldsymbol{x}\in X}{\text{minimize}} \quad f_l(\boldsymbol{x}),\ l=1,2,\ldots,k \quad (4)$$

$$\underset{\boldsymbol{x}\in X}{\text{maximize}} \quad f_l(\boldsymbol{x}),\ l=1,2,\ldots,k \quad (5)$$

Step 2: In consideration of individual minimal value and maximum one of each objective function, the decision maker subjectively specifies membership functions $\mu_l(f_l(\boldsymbol{x}))$, $l=1,\ldots,k$ to quantify fuzzy goals for objective functions. Next, the decision maker sets initial reference membership function values $\bar{\mu}_l$, $l=1,\ldots,k$.

Step 3: We solve the following augmented minimax problem corresponding to current reference membership function values (3).

Step 4: If the decision maker is satisfied with the solution obtained in Step 3, the interactive procedure is finished. Otherwise, the decision maker updates reference membership function values $\bar{\mu}_l$, $l=1,2,\ldots,k$ based on current membership function values and objective function values, and return to Step 3.

IV. Particle swarm optimization

Particle swarm optimization [2] is based on the social behavior that a population of individuals adapts to its environment by returning to promising regions that were previously discovered [3]. This adaptation to the environment is a stochastic process that depends on both the memory of each individual, called particle, and the knowledge gained by the population, called swarm.

In the numerical implementation of this simplified social model, each particle has four attributes: the position vector in the search space, the velocity vector and the best position in its track and the best position of the swarm. The process can be outlined as follows.

Step 1: Generate the initial swarm involving N particles at random.

Step 2: Calculate the new velocity vector of each particle, based on its attributes.

Step 3: Calculate the new position of each particle from the current positon and its new velocity vector.

Step 4: If the termination condition is satisfied, stop. Otherwise, go to Step 2.

To be more specific, the new velocity vector of the i-th particle at time t, $\boldsymbol{v}_i^{t+1}$, is calculated by the following scheme introduced by Shi and Eberhart [10].

$$\boldsymbol{v}_i^{t+1} := \omega^t\boldsymbol{v}_i^t + c_1R_1^t(\boldsymbol{p}_i^t-\boldsymbol{x}_i^t) + c_2R_2^t(\boldsymbol{p}_g^t-\boldsymbol{x}_i^t) \quad (6)$$

In (6), R_1^t and R_2^t are random numbers between 0 and 1, $\boldsymbol{p}_i^t$ is the best position of the i-th particle in its track and $\boldsymbol{p}_g^t$ is the best position of the swarm. There are three problem dependent parameters, the inertia of the particle ω^t, and two trust parameters c_1, c_2.

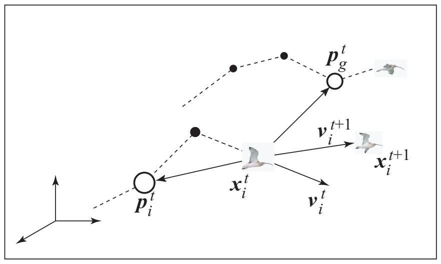

Fig. 1. Movement of a particle in PSO.

Then, the new position of the i-th particle at time t, x_i^{t+1}, is calculated from (7).

$$x_i^{t+1} := x_i^t + v_i^{t+1} \tag{7}$$

where x_i^t is the current position of the i-th particle at time t. The i-th particle calculates the next search direction vector v_i^{t+1} by (6) in consideration of the current search direction vector v_i^t, the direction vector going from the current search position x_i^t to the best position in its track p_i^t and the direction vector going from the current search position x_i^t to the best position of the swarm p_g^t, moves from the current position x_i^t to the next search position x_i^{t+1} calculated by (7). The parameter ω^t controls the amount of the move to search globally in early stage and to search localy by decreasing ω^t gradually.

The searching procedure of PSO is shown in Fig. 1. Comparing the evaluation value of a particle after movement, $f(x_i^{t+1})$, with that of the best position in its track, $f(p_i^t)$, if $f(x_i^{t+1})$ is better than $f(p_i^t)$, then the best position in its track is updated as $p_i^t := x_i^{t+1}$. Futhermore, if $f(p_i^{t+1})$ is better than $f(p_g^t)$, then the best position in the swarm is updated as $p_g^{t+1} := p_i^{t+1}$.

Such a PSO technique includes two problems. One is that particles concentrate on the best search positon of the swarm and they cannot easily escape from the local optimal solution since the move direction vector v_i^{t+1} calculated by (6) always includes the direction vector to the best search position of the swarm. Another is that a particle after move is not always feasible for problems with constraints.

V. Improvement of particle swarm optimization

In this study, to prevent the concentration and stop at local optimal solutions of particles in the simple PSO, we introduce the modification of move schemes of a particle, the leaving act and the multiple stretching technique. In addition, in order to treat constraints, we divide the swarm into two subswarms. In one subswarm, since the move of a particle to the infeasible region are not accepted, if a particle becomes infeasible after a move, it is repaired to be feasible. In the other subswarm, the move of a particle to the infeasible region are accepted.

A. Move of a particle

We think about the move from current search position x_i^t of the i-th particle.

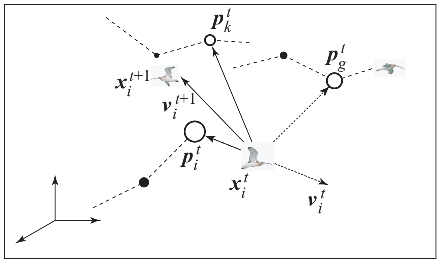

Fig. 2. The new search direction when the best search point of a particle is renewed at the previous search point.

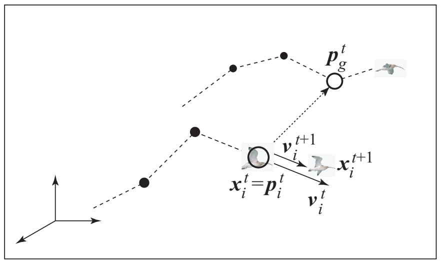

Fig. 3. The new search direction when the best search point of a particle is renewed at the current search point.

First, if the previous search position x_i^{t-1} is the best positon of a particle in its track p_i^t, the next search position x_i^{t+1} moves near the best position in the swarm p_g^t with high possibility. Thereby, as shown in Fig.2, we change the equation (6) to determine the next search direction v_i^{t+1} as

$$v_i^{t+1} := c_1 R_1^t (p_i^t - x_i^t) + c_2 R_2^t (p_k^t - x_i^t) \tag{8}$$

where p_i^t is the best position of the i-th particle. By the change of the search direction determination scheme, we can relax concentration of particles to p_g^t.

Next, in case that the current search position x_i^t is the best position of a particle in its track p_i^t, the direction to current search position is desirable. Thus, as in Fig.3, we change the equation (6) to determine the next search direction v_i^{t+1} as

$$v_i^{t+1} := \omega^t v_i^t \tag{9}$$

B. Division of the swarm into two subswarms

In application of PSO to optimization problems with constraints, a particle after move is not always feasible if we use the updating equation of search position mentioned above. To deal with such a situation, we divide the swarm into two subswarms. In one subswarm, since the move of a particle to the infeasible region is not accepted, if a particle becomes infeasible after a move, it is repaired to be feasible. To be more specific, with respect to infeasible particles which violate constraints after move, we repair its search position to be feasible by the bisection method on the direction from the search position before move, x_i^t, to that after move, x_i^{t+1}. In the other subswarm, the move of a particle to the infeasible region are accepted.

C. Secession

Since particles tend to concentrate on the best position of the swarm as the search goes forward in PSO, the global search becomes difficult. Thus, we introduce following leaving acts of a particle.

(1) [Secession I] A particle moves at random to a point in the feasible region.

(2) [Secession II] A particle moves at random to a point on the boundary of the feasible region.

(3) [Secession III] A particle moves at random to a point in a direction of some coordinate axis.

D. Multiple stretching technique

Stretching technique is suggested to prevent the stop at a local optimal solution of particles in PSO by Parsopoulos et al. [6]. It enables particles to escape from the current local optimal solution and not to approach the same local optimal solution again by changing original evaluation function $f(x)$ to other evaluation function $H(x)$ defined as follows.

$$G(\boldsymbol{x}) = f(\boldsymbol{x}) + \gamma_1 \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \Big(\mathrm{sign}\big(f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}})\big) + 1\Big) \quad (10)$$

$$H(\boldsymbol{x}) = G(\boldsymbol{x}) + \gamma_2 \frac{\mathrm{sign}\big(f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}})\big) + 1}{\tanh\Big(\mu\big(G(\boldsymbol{x}) - G(\bar{\boldsymbol{x}})\big)\Big)} \quad (11)$$

Here $\bar{\boldsymbol{x}}$ is the current local optimal solution, and γ_1, γ_2, μ are parameters. In addition, sign$(\cdot)$ is defined function as follows.

$$\mathrm{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \quad (12)$$

In the function $G(\boldsymbol{x})$, the second term is the penalty depending on the distance between a search position $\boldsymbol{x}$ and the current local optimal solution $\bar{\boldsymbol{x}}$. The term is equal to 0 for $\boldsymbol{x}$ whose objective function value $f(\boldsymbol{x})$ is better than $f(\bar{\boldsymbol{x}})$, while it takes a value depending on the distance between $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$ for $\boldsymbol{x}$ whose $f(\boldsymbol{x})$ is worse than $f(\bar{\boldsymbol{x}})$. Next, the function $H(\boldsymbol{x})$ is defined using $G(\boldsymbol{x})$ expressed in (10). The value of $H(\boldsymbol{x})$ for a search position $\boldsymbol{x}$ is equal to the objective function value $f(\boldsymbol{x})$ of $\boldsymbol{x}$ if $f(\boldsymbol{x})$ of $\boldsymbol{x}$ is better than that of $\bar{\boldsymbol{x}}$, while it takes a very large value if $f(\boldsymbol{x})$ of $\boldsymbol{x}$ is worse than that of $\bar{\boldsymbol{x}}$. Using $H(\boldsymbol{x})$ as the new evaluation function, particles can escape from the current local optimal solution and search a new region which may include better solutions than the current local optimal solution.

Although the stretching technique [6] enables particles to escape from the current local optimal solution, they may stop at the same local optimal solution again when we apply the stretching technique to the next local optimal solution. Thus we use multiple Stretching technique corresponding to plural local solution. To be concrete, we consider the following functions for m local optimal solutions $\bar{\boldsymbol{x}}_k$, $k = 1, \ldots, m$.

$$G_k(\boldsymbol{x}) = f(\boldsymbol{x}) + \gamma_1 \|\boldsymbol{x} - \bar{\boldsymbol{x}}_k\| \Big(\mathrm{sign}\big(f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}_{\min})\big) + 1\Big) \quad (13)$$

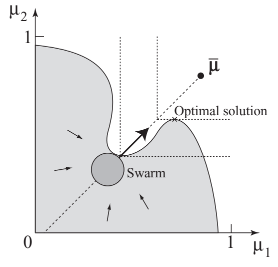

Fig. 4. Application of rPSO to the augemented minimax problem.

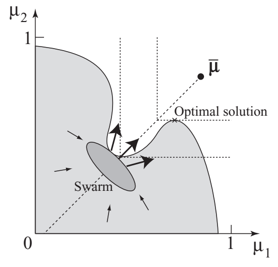

Fig. 5. Application of the proposed PSO to the augemented minimax problem.

$$H_k(\boldsymbol{x}) = G_k(\boldsymbol{x}) + \gamma_2 \frac{\mathrm{sign}\big(f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}_{\min})\big) + 1}{\tanh\Big(\mu\big(G_k(\boldsymbol{x}) - G_k(\bar{\boldsymbol{x}}_k)\big)\Big)} \quad (14)$$

$$S(\boldsymbol{x}) = \sum_{k=1}^{m} H_k(\boldsymbol{x})/m \quad (15)$$

Here, $\bar{\boldsymbol{x}}_{\min}$ is the best among m local optimal solutions. The value of $S(\boldsymbol{x})$ for a search position $\boldsymbol{x}$ is equal to the objective function value $f(\boldsymbol{x})$ of $\boldsymbol{x}$ if $f(\boldsymbol{x})$ is better than that of $\bar{\boldsymbol{x}}_{\min}$, while it takes a very large value if the distance between $\boldsymbol{x}$ and the nearest local optimal solution is less than a certain value. Otherwise, it takes a value depending on the distance.

E. Incorporation of the direction to non-dominated particles in the swarm

Since the shape of the objective function of the augmented minimax problem (3) solved in the interactive fuzzy satisficing method for multiobjective nonlinear programming problems often becomes complex, there does not always exist the optimal solution int the seach direction of the swarm, shown in Fig.(4).

Thereby, we pay attention to the fact that the optimal solution to the augmented minimax problem is one of M-Pareto optimal solutions to (2) and introduce the direction to a non-dominated particle (an approximate M-Pareto optimal solution) to carry out search about M-Pareto optimality and search about the optimality with respect to the objective function of the augmented minimax problem shown as Fig.(5).

In order to incorporate the new search direction, we revise the equation to determine the next search direction as:

$$\boldsymbol{v}_i^{t+1} = \omega^t \boldsymbol{v}_i^t + c_3 R_3^t (\boldsymbol{x}_{\lambda_k}^t - \boldsymbol{x}_i^t) \quad (16)$$

where $\boldsymbol{x}_{\lambda_k}^t$ is a certain nondominated particle in the current swarm, c_3 is a parameter and R_3^t is the uniform random number in the interval $[0, 1]$.

Moreover, it is considered that the current search position and the direction are favorable since the best position of a particle is updated when the current search position is the best position of a particle. So that a particle decides the next search direction $\boldsymbol{v}_i^{t+1}$ by the translation equation (9)same as Morihara[5].

By the swarm to move in this way, we can do search about Pareto optimality.

In rPSO [5], the next search position is decided by vector to a particle k instead of the best search position of the swarm in order to relax that a particle concentrates on the best of the swarm when the best position of a particle is updated just before that. The technique is not suitable to solve the augmented minimax problem since it is hard to consider the vector is the direction provided better solution when any particle k is dominated.

Thus we chage the next search position $\boldsymbol{v}_i^{t+1}$ by following equation (17) using the vector to the current seach position of any non-dominated solution λ_k insead of the best position of any particle k.

$$\boldsymbol{v}_i^{t+1} = c_1 R_1^t (\boldsymbol{p}_i^t - \boldsymbol{x}_i^t) + c_2 R_2^t (\boldsymbol{x}_{\lambda_k}^t - \boldsymbol{x}_i^t) \quad (17)$$

With this modification, the swarm moving by traditional updating equation acts leaving behavior to the direction to the non-dominated solution. In addition, we can do both search about Pareto optimality proper and prevent a particle from concentrating on the local solution excessively in the searching process.

F. Algorithm of the proposed PSO method

The algorithm of the proposed PSO method based on rPSO for multiobjective nonlinear programming problems (MOrPSO) is summarized as follows.

Step 1: Find one feasible solution by PSO in consideration of the degree of violation of constraints, and set it a basic point solution of the homomorphous mapping $\boldsymbol{r}$. Set $t := 0$, go to Step 2.

Step 2: Generate the initial search position of each particle with the guarantee on feasibility using the homomorphous mapping. To be concrete, after generating N points randomly in a hypercube, map them into the feasible region by the homomorphous mapping with the basic point solution $\boldsymbol{r}$. Regard these points as initial search positions of particles $\boldsymbol{x}_i^0$, $i = 1, \ldots, N$. In addition, use the initial search position of each particle $\boldsymbol{x}_i^0$ as the best position of itself $\boldsymbol{p}_i^0$. Then, find the best position among those, and use it as the best position of the initial swarm $\boldsymbol{p}_g^0$. Moreover, find nondominated particles based on membership function values for objective functions. Go to Step 3.

Step 3: Determine the value of ω^t. If the current particle makes use of the information of nondominated particles, go to Step 4. Otherwise, if it does that of the best position of the swarm, go to Step 5.

Step 4: Calculate the direction $\boldsymbol{v}_i^{t+1}$ to next search position $\boldsymbol{x}_i^{t+1}$ of the partilce using the transition scheme according to its situation based on the current position $\boldsymbol{x}_{\lambda_k}$ of any nondominated particle. Then, move it using eq.(7), and go to Step 6.

Step 5: Calculate the direction $\boldsymbol{v}_i^{t+1}$ to next search position $\boldsymbol{x}_i^{t+1}$ of the partilce using the transition scheme according to its situation based on $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_g^t$. Then, move it using eq.(7), and go to Step 6.

Step 6: For paticles to be repaired about infeasibility by the bisection method, check whether the search position of the paticle after transition $\boldsymbol{x}_i^{t+1}$ is feasible or not. If it is infeasible, repair it to be feasible using the bisection method. Go to Step 7.

Step 7: Decide whether the multiple stretching technique is carried out or not. If it is carried out, go to Step 8. Otherwise, go to Step 9.

Step 8: Using the evaluation function $S(\cdot)$ derived from the objective function $f(\cdot)$, evaluate the search position of each particle after transition $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$, and go to Step 9.

Step 9: Using the objective function $f(\cdot)$, evaluate the search position of each particle after transition $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$, and go to Step 10.

Step 10: If the evaluation value of a particle calculated in Step 8 or Step 9 is better than that of the best search position of itself $\boldsymbol{p}_i^t$, update the best position of itself as $\boldsymbol{p}_i^{t+1} := \boldsymbol{x}_i^{t+1}$, and goto Step 11. Otherwise, set $\boldsymbol{p}_i^{t+1} := \boldsymbol{p}_i^t$, and go to Step 11.

Step 11: If the evaluation value of a partilce calculated in Step 8 or Step 9 is better than that of the best search position of the swarm $\boldsymbol{p}_g^t$, update the best position of the swarm as $\boldsymbol{p}_g^{t+1} := \boldsymbol{x}_{i_{\min}}^{t+1}$, and go to Step 12. Otherwise, set $\boldsymbol{p}_g^{t+1} := \boldsymbol{p}_g^{t+1}$, and go to Step 12.

Step 12: Compare membership function values for each particle, and update the set of nondominated particles in the swarm, and go to Step 13.

Step 13: If the condition of the secession is satisfied, apply the secession procedure to each particle, and go to Step 14.

Step 14: If $t = T_{\max}$ (the maximal search generation number). Otherwise, set $t := t + 1$, and return to Step 3.

Here, the termination condition assumes that a number of iterations of the procedure exceeds a maximum number of interations. And an application condition of multiple Stretching technique for a particle i assumes that the best solution of a particle does not update constant turn.

TABLE I
RESULTS OF APPLICATION OF THREE METHODS TO AUGMENTED MINIMAX PROBLEMS.

	interactive	1st	2nd	time
	$(\bar\mu_1, \bar\mu_2, \bar\mu_3)$	$(1.0, 1.0, 1.0)$	$(0.8, 1.0, 1.0)$	(sec)
MOrPSO (proposed)	best	0.1430	0.0823	9.03
	average	0.1452	0.0838	
	worst	0.1488	0.0859	
rPSO [5]	best	0.1434	0.0826	8.55
	average	0.1466	0.0854	
	worst	0.1674	0.1102	
RGENOCOP V [9]	best	0.1825	0.1406	42.90
	average	0.1880	0.1568	
	worst	0.1952	0.1811	

VI. NUMERICAL EXAMPLE

In order to show the efficiency of the proposed PSO (MOrPSO), we consider the following multiobjective nonlinear programming problem.

$$\begin{aligned}
&\text{minimize}\\
&\quad f_1(\boldsymbol{x}) = 7x_1^2 - x_2^2 + x_1x_2 - 14x_1 - 16x_2 + 8(x_3-10)^2\\
&\qquad +4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2\\
&\qquad +7(x_8-11)^2 + 2(x_9-10)^2 + x_{10}^2 + 45\\
&\text{minimize}\\
&\quad f_2(\boldsymbol{x}) = (x_1-5)^2 + 5(x_2-12)^2 + 0.5x_3^4 + 3(x_4-11)^2\\
&\qquad +0.2x_5^5 + 7x_6^2 + 0.1x_7^4 - 4x_6x_7 - 10x_6 - 8x_7\\
&\qquad +x_8^2 + 3(x_9-5)^2 + (x_{10}-5)^2\\
&\text{minimize}\\
&\quad f_3(\boldsymbol{x}) = x_1^3 + (x_2-5)^2 + 3(x_3-9)^2 - 12x_3 + 2x_4^3\\
&\qquad +4x_5^2 + (x_6-5)^2 + 6x_7^2 + 7(x_7-2)x_8^2\\
&\qquad -x_9x_{10} + 4x_9^3 + 5x_1 - 8x_1x_7\\
&\text{subject to}\\
&-3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4\\
&\qquad\qquad -2x_5x_6x_8 + 120 \geq 0\\
&-5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0\\
&-x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 - 6x_5x_6 \geq 0\\
&-0.5(x_1-8)^2 - 2(x_2-4)^2 - 3x_5^2 + x_5x_8 + 30 \geq 0\\
&3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0\\
&4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 105\\
&10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0\\
&-8x_1 + 2x_2 + 5x_9 - 17x_{10} \leq 12\\
&-5.0 \leq x_j \leq 10.0,\ j = 1, \ldots, 10
\end{aligned}$$

We apply the original rPSO [5], RGENOCOP V [9] and the proposed PSO (MOrPSO) to minimax problems solved in the interactive fuzzy satisficing method for the above problem. The results obtained by these three methods are shown in Table I. In these experiments, we set the swarm size $N = 70$, the maximal search generation number $T_{\max} = 5000$. In addition, we use the following membership functions: $\mu_{f_1}(\boldsymbol{x}) = (1500 - f_1(\boldsymbol{x}))/1420$, $\mu_{f_2}(\boldsymbol{x}) = (3500 - f_2(\boldsymbol{x}))/3300$, $\mu_{f_3}(\boldsymbol{x}) = (3100 - f_3(\boldsymbol{x}))/3050$.

From table I, in the application of rPSO [5], we can get better solutions in the sense of best, average and worst than those obtained by RGENOCOP V [9]. This indicates that rPSO is better than RGENOCOP V about the accuracy of solutions. However, as for the difference between best and worst, that for rPSO is larger than that for RGENOCOP V, i.e., rPSO is worse than RGENOCOP V with respect to the precision of solutions.

On the other hand, the results obtained by the prooposed MOrPSO are better than those by rPSO in the sense of best, average, worst, the difference between best and worst. Therefore, MOrPSO proposed in this paper is the best solution method among these methods with respect to both the accuracy of solutions and the precision of solutions.

VII. CONCLUSION

In this paper, we focused on multiobjective nonlinear programming problems and proposed a new PSO technique which is efficient for in applying the interactive fuzzy satisficing method. In particular, considering the features of augmented minimax problems solved in the tnteractive fuzzy satisficing method, we incorporated the new direction determination scheme into rPSO. Finally, we showed the efficiency of the proposed MOrPSO by applying it to numerical examples.

ACKNOWLEDGMENT

This research was partially supported by The 21st Century COE Program on “Hyper Human Technology toward the 21st Century Industrial Revolution”.

REFERENCES

[1] R.E. Bellman, L.A. Zadeh, Decision making in a fuzzy environment, Management Science, vol. 17, pp. 141–164, 1970.
[2] J. Kennedy and R.C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948, 1995.
[3] J. Kennedy, W.M. Spears, Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator, Proceedings of IEEE International Conference on Evolutionary Computation, pp. 78–83, 1998.
[4] S. Koziel, Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation, vol. 7, no. 1, pp. 19–44, 1999.
[5] K. Morihara, M. Sakawa, K. Kato, H. Katagiri, T. Uno, Heuristic solution method based on particle swarm optimization for nonlinear programming problems, Proceedings of 9th Japan Society for Fuzzy Theory and Intelligent Informatics Chugoku / Shikoku Branch Office Meeting, pp. 21–24, 2004.
[6] K.E. Parsopoulos, M.N. Varahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing, no. 1, pp. 235–306, 2002.
[7] H. Rommelfanger, Fuzzy linear programming and applications, European Journal of Operational Research, vol. 92, pp. 512–527, 1996.
[8] M. Sakawa, Fuzzy Sets and Interactive Multiobjective Optimization, Plenum Press, New York, 1993.
[9] M. Sakawa, K. Kato, T. Suzuki, An interactvie fuzzy satisficing method for multiobjective non-convex programming problems through genetic algorithms, Proceedings of 8th Japan Society for Fuzzy Theory and Systems Chugoku / Shikoku Branch Office Meeting, pp. 33–36, 2002.
[10] Y.H. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73, 1998.
[11] H.-J. Zimmermann, Fuzzy programming and linear programming with several objective functions, Fuzzy Sets and Systems, vol. 1 pp. 45–55, 1978.
[12] H.-J. Zimmermann, Fuzzy mathematical programming, Computers & Operations Research, vol. 10, pp. 291–298, 1983.