Chapter from the book *Particle Swarm Optimization*
Downloaded from: http://www.intechopen.com/books/particle_swarm_optimization

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
1. 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 methods 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 (Kennedy & Eberhart, 1995). 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 (Matsui et al., 2008).

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. (Zimmermann, 1983), (Rommelfanger, 1996), 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...
It has been implicitly assumed that the fuzzy decision or the minimum-operator of (Bellman & Zadeh, 1970) 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 (Sakawa et al., 2002).

In this paper, focusing on multiobjective nonlinear programming problems, we attempt to derive satisficing solutions through the interactive fuzzy satisficing method. Since problems solved in the interactive fuzzy satisficing method for multiobjective nonlinear programming problems are nonlinear programming problems, we adopt rPSO (Matsui et al., 2008) 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.

2. Multiobjective nonlinear programming problems

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

\[
\begin{align*}
\text{minimize} & \quad f_l(x), \quad l = 1, 2, \ldots, k \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, 2, \ldots, m \\
& \quad l_j \leq x_j \leq u_j, \quad j = 1, 2, \ldots, n \\
& \quad x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n
\end{align*}
\]

where \( f_l(\cdot) \) and \( g(\cdot) \) 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 denote the feasible region of (1) by \( X \).

3. 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 "\( f_l(x) \) should be substantially less than or equal to a certain value", (1) can be rewritten as:

\[
\begin{align*}
\text{maximize} & \quad (\mu_1(f_1(x)), \ldots, \mu_k(f_k(x))) \\
\text{subject to} & \quad x \in X
\end{align*}
\]

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 exists a complete optimal solution that simultaneously optimizes all objective functions. As a reasonable solution concept for the fuzzy multiobjective decision making problem, M. Sakawa defined M-Pareto optimality on the basis of membership function values by directly extending the Pareto optimality in the ordinary multiobjective programming problem (Sakawa, 1993). 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 (Sakawa, 1993). For the decision maker's reference membership levels \( \mu_l, \quad l = 1, \ldots, k \), the corresponding M-Pareto optimal solution, which is nearest to the
requirements in the minimax sense or better than it if the reference membership levels are attainable, is obtained by solving the following augmented minimax problem (3).

\[
\begin{align*}
\text{minimize} & \quad \max_{l=1, \ldots, k} \left\{ (\mu_i - \mu_i(f_i(x))) + \rho \sum_{i=1}^{k} (\bar{\mu}_i - \mu_i(f_i(x))) \right\} \\
\end{align*}
\]

where \( \rho \) 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.

\[
\begin{align*}
\text{minimize} & \quad f_i(x), \ l = 1, 2, \ldots, k \\
\text{maximize} & \quad f_i(x), \ l = 1, 2, \ldots, k \\
\end{align*}
\]

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

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 \( \bar{\mu}_i, \ l = 1, 2, \ldots, k \) based on current membership function values and objective function values, and return to Step 3.

4. Particle swarm optimization

Particle swarm optimization (Kennedy & Eberhart, 1995) is based on the social behavior that a population of individuals adapts to its environment by returning to promising regions that were previously discovered (Kennedy & Spears, 1998). 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 $t$, $v^{t+1}_i$, is calculated by the following scheme introduced by (Shi & Eberhart, 1998).

$$v^{t+1}_i := \omega^t v^t_i + c_1 R^t_1 (p^t_i - x^t_i) + c_2 R^t_2 (p^t_g - x^t_i)$$  \hspace{1cm} (6)

In (6), $R^t_1$ and $R^t_2$ are random numbers between 0 and 1, $p^t_i$ is the best position of the $i$-th particle in its track and $p^t_g$ is the best position of the swarm. There are three problem dependent parameters, the inertia of the particle $\omega^t$, and two trust parameters $c_1$, $c_2$. Then, the new position of the $i$-th particle at time $t$, $x^{t+1}_i$, is calculated from (7).

$$x^{t+1}_i := x^t_i + v^{t+1}_i$$  \hspace{1cm} (7)

where $x^t_i$ is the current position of the $i$-th particle at time $t$. The $i$-th particle calculates the next search direction vector $v^{t+1}_i$ by (6) in consideration of the current search direction vector $v^t_i$, the direction vector going from the current search position $x^t_i$ to the best position in its track $p^t_i$ and the direction vector going from the current search position $x^t_i$ to the best position of the swarm $p^t_g$, moves from the current position $x^t_i$ to the next search position $x^{t+1}_i$ calculated by (7). The parameter $\omega^t$ controls the amount of the move to search globally in early stage and to search locally by decreasing $\omega^t$ gradually.

The searching procedure of PSO is shown in Fig. 1.

Comparing the evaluation value of a particle after movement, $f(x^{t+1}_i)$, with that of the best position in its track, $f(p^t_i)$, if $f(x^{t+1}_i)$ is better than $f(p^t_i)$, then the best position in its track is updated as $p^t_i := x^{t+1}_i$. Furthermore, if $f(p^{t+1}_i)$ is better than $f(p^t_g)$, then the best position in the swarm is updated as $p^{t+1}_g := p^{t+1}_i$. 
In the original PSO method, however, there are drawbacks that it is not directly applicable to constrained problems and it is liable to stopping around local optimal solutions. To deal with these drawbacks of the original PSO method, we incorporate the bisection method and a homomorphous mapping to carry out the search considering constraints. In addition, we proposed the multiple stretching technique and modified move schemes of particles to restrain the stopping around local optimal solutions (Matsui et al., 2008). Thus, we applied rPSO for interactive fuzzy multiobjective nonlinear programming problems and proposed multiobjective revised PSO (MOrPSO) method incorporating move scheme to the nondominated particle in order to search effectively for the augmented minimax problems (Matsui et al., 2007). In the application of large-scale augmented minimax problem, MOrPSO method is superior than rPSO method on efficiency. On the other hand, MOrPSO method is inferior on accuracy.

5. Improvement of MOrPSO

We show the results of the applicaltion of the original rPSO (Matsui et al., 2008) and MOrPSO (Matsui et al., 2007) to the augmented minimax problem for multiobjective nonlinear programming problem with \( l = 2, n = 55 \) and \( m = 100 \) in Table 1. In these experiments we set the swarm size \( N = 70 \), the maximal search generation number \( T_{\text{max}} = 5000 \). In addition, we use the following membership functions: \( \mu_1 = 1.0, \mu_2 = 1.0 \).

<table>
<thead>
<tr>
<th>method</th>
<th>objective function value (minimize)</th>
<th>computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best</td>
<td>average</td>
</tr>
<tr>
<td>rPSO</td>
<td>0.3464</td>
<td>0.4471</td>
</tr>
<tr>
<td>MOrPSO</td>
<td>0.3614</td>
<td>0.4095</td>
</tr>
</tbody>
</table>

Table 1. Results of the application to the augmented minimax problem

From Table 1, MOrPSO method is superior than rPSO method on efficiency in the average value, the worst one and computational time. However, the best value of MOrPSO method is worse than that of rPSO method, MOrPSO method is inferior on accuracy. We consider
the case that the search accuracy turns worse incorporating the direction to nondominated particle (approximate M-Pareto optimal solution) in MOrPSO method. In this paper, we improve the search accuracy incorporating external archives to record nondominated particles in the swarm. Here, as recorded nondominated particle increases in archives, computational time increases in order to judge whether a particle is nondominated. Therefore, there is many computational time that we record all the nondominated particles to archives. Thus we divide membership function space with hypercube shown in Fig. 2 and record a number of nondominated particle included in each hypercube.

![Figure 2. reduction of archives with grid (l=2)](image)

When a number of nondominated particle recorded in archives is greater than a fixed number, we delete one particle from hypercube with many numbers of nondominated particle and record new solution (particle). We consider that can reduce computational time and express approximate M-Pareto optimal front by a few particles incorporating reduction of archives. We show the results of the application of MOrPSO method incorporating reduction of archives (MOrPSO-1) to the above same problem in Table 2.

<table>
<thead>
<tr>
<th>method</th>
<th>objective function value (minimize)</th>
<th>computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best</td>
<td>average</td>
</tr>
<tr>
<td>MOrPSO-1</td>
<td>0.4379</td>
<td>0.4708</td>
</tr>
</tbody>
</table>

Table 2. Results of the application to the augmented minimax problem

From Table 2, it is clear that all of the best, average, worst value and computational time obtained by MOrPSO-1 are worse than those obtained by MOrPSO. We consider that a particle moves using nondominated particle which is not useful since all nondominated particles in archives are used in search. Thus, in the membership function space, we consider that information of a particle existing far from the reference membership value is hard to contribute to search and introduce threshold value for selection of nondominated particle as shown in Fig. 3.
We show the results of the application of MOrPSO method incorporating limitation by threshold value (MOrPSO-2) to the above same problem in Table 3.

<table>
<thead>
<tr>
<th>method</th>
<th>objective function value (minimize)</th>
<th>computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best</td>
<td>average</td>
</tr>
<tr>
<td>MOrPSO-2</td>
<td>0.2993</td>
<td>0.3430</td>
</tr>
</tbody>
</table>

Table 3. Results of the application to the augmented minimax problem

From Table 3, in the application of MOrPSO method incorporating limitation by threshold value (MOrPSO-2), we can get better solutions in the sense of best, average and worst than those obtained by rPSO and MOrPSO.

In order to show the efficiency of the proposed PSO, we consider the multiobjective nonlinear programming problem with \( l = 2 \) and \( n = 100 \). In these experiments, we set the swarm size \( N = 100 \), the maximal search generation number \( T_{\text{max}} = 5000 \). In addition, we use the following reference membership function values: \( \mu_1 = 1.0, \mu_2 = 1.0 \).

<table>
<thead>
<tr>
<th>method</th>
<th>objective function value (minimize)</th>
<th>computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best</td>
<td>average</td>
</tr>
<tr>
<td>rPSO [6]</td>
<td>0.2547</td>
<td>0.2783</td>
</tr>
<tr>
<td>MOrPSO [7]</td>
<td>0.1950</td>
<td>0.2033</td>
</tr>
<tr>
<td>MOrPSO-2</td>
<td>0.2018</td>
<td>0.2320</td>
</tr>
</tbody>
</table>

Table 4. Results of the application to the augmented minimax problem

From Table 4, it is clear that all of the best, average, worst value and computational time obtained by MOrPSO-2 are worse than those obtained by MOrPSO. We consider that it
occurs to make no use of the information of nondominated particle in search since there is a few nondominated particle information stored in archives of MOrPSO-2 and only the best value of each objective function and the best value limb of the augmented minimax problem are saved. Therefore, we propose MOrPSO with external archives (MOrPSO-EA) using nondominated particle in the swarm same as MOrPSO in order to store various nondominated particle as possible in normal search. And the results of the application are shown in Table 5.

<table>
<thead>
<tr>
<th>method</th>
<th>objective function value (minimize)</th>
<th>computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best</td>
<td>average</td>
</tr>
<tr>
<td>MOrPSO</td>
<td>0.1950</td>
<td>0.2033</td>
</tr>
<tr>
<td>MOrPSO-EA</td>
<td>0.1746</td>
<td>0.1787</td>
</tr>
</tbody>
</table>

Table 5. Results of the application to the augmented minimax problem

From Table 5, in the application of MOrPSO-EA, we can get better solutions in the sense of best, average and worst than those obtained by MOrPSO.

6. Numerical examples

In MOrPSO, it searches globally in the early generation and locally decreasing $\omega t$. However, we consider that necessity to search globally in the early generation is low after the second time since the information of nondominated particle to current generation is stored in archives in proposed MOrPSO. Therefore, we consider that the proposed MOrPSO can search locally in the early generation.

In order to show the efficiency of the proposed MOrPSO, we consider the multiobjective nonlinear programming problem with $l = 2$ and $n = 100$ and $m = 55$. In these experiments, we set the maximal search generation number of MOrPSO and the proposed MOrPSO (MOrPSO-EA) in the 1st interactive $T_{max} = 5000$ and in the 2nd and 3rd interactive $T_{max} = 3000$. We show the results of the application are shown in Table 6 and 7.

<table>
<thead>
<tr>
<th>interactive</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\mu}_1$</td>
<td>1.0</td>
<td>1.0</td>
<td>0.85</td>
</tr>
<tr>
<td>$\bar{\mu}_2$</td>
<td>1.0</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>$\mu_1(x)$</td>
<td>0.6157</td>
<td>0.8003</td>
<td>0.7575</td>
</tr>
<tr>
<td>$\mu_2(x)$</td>
<td>0.6157</td>
<td>0.5003</td>
<td>0.6075</td>
</tr>
<tr>
<td>minimax value</td>
<td>0.3844</td>
<td>0.1997</td>
<td>0.0925</td>
</tr>
<tr>
<td>time (sec)</td>
<td>127.20</td>
<td>128.89</td>
<td>131.25</td>
</tr>
</tbody>
</table>

Table 6. Interactive fuzzy programming through MOrPSO
Particle Swarm Optimization with External Archives for Interactive Fuzzy Multiobjective Nonlinear Programming

<table>
<thead>
<tr>
<th>interactive</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\mu}_1$</td>
<td>1.0</td>
<td>1.0</td>
<td>0.85</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>1.0</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>$\mu_1(x)$</td>
<td>0.7183</td>
<td>0.8458</td>
<td>0.8042</td>
</tr>
<tr>
<td>$\mu_2(x)$</td>
<td>0.7183</td>
<td>0.5458</td>
<td>0.6542</td>
</tr>
<tr>
<td>minimax value</td>
<td>0.2817</td>
<td>0.1542</td>
<td>0.0458</td>
</tr>
<tr>
<td>time (sec)</td>
<td>157.37</td>
<td>98.58</td>
<td>101.16</td>
</tr>
</tbody>
</table>

Table 7. Interactive fuzzy programming through MOrPSO-EA (proposed)

From Table 6 and 7, MOrPSO-EA is superior than MOrPSO on accuracy. In addition, we can decrease total computational time by reducing the maximal search generation number.

6. Conclusion

In this research, we focused on multiobjective nonlinear programming problems and proposed a new MOrPSO technique which is accuracy for in applying the interactive fuzzy satisficing method. In particular, considering the features of augmented minimax problems solved in the interactive fuzzy satisficing method, we incorporated use of external archives, reduction of archives and the limitation of threshold value. Finally, we showed the efficiency of the proposed MOrPSO by applying it to numerical examples.

7. References


Particle swarm optimization (PSO) is a population based stochastic optimization technique influenced by the social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. This book represents the contributions of the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following: