Stackelberg Solutions to Noncooperative Two-level Nonlinear Programming Problems through Particle Swarm Optimization

Takeshi MATSUI* and Masatoshi SAKAWA*

*Faculty of Engineering, Hiroshima University

Abstract: In this paper, we focus on two-level nonlinear programming problems with no coordination between the decision maker at the upper level (the leader) and the decision maker at the lower level (the follower), and propose a computational method through particle swarm optimization (PSO) for obtaining Stackelberg solutions to two-level nonlinear programming problems. Furthermore, we carry out numerical experiments in order to demonstrate the feasibility and effectiveness of the proposed method by comparing with existing methods.

Keywords Two-level nonlinear programming, Stackelberg solutions, Particle swarm optimization.

1. Introduction

In the real world, we often encounter the situations where there are two or more decision makers in an organization with a hierarchical structure, and they make decisions in turn or at the same time so as to optimize their objective functions. Decision making problems in decentralized organizations are often modeled as Stackelberg games [24], and they are formulated as two-level mathematical programming problems [21, 22]. In the context of two-level programming, the decision maker at the upper level first specifies a strategy, and then the decision maker at the lower level specifies a strategy so as to optimize the objective with full knowledge of the action of the decision maker at the upper level. In conventional multi-level mathematical programming models employing the solution concept of Stackelberg equilibrium, it is assumed that there is no communication among decision makers, or they do not make any binding agreement even if there exists such communication.

Computational methods for obtaining Stackelberg solutions to two-level linear programming problems are classified roughly into three categories: the vertex enumeration approach [3], the Kuhn-Tucker approach [2, 3, 5, 11], and the penalty function approach [25]. The subsequent works on two-level programming problems under noncooperative behavior of the decision makers have been appearing [6, 8, 10, 16, 17, 18], including some applications to aluminium production [15], pollution control policy determination [1], tax credits determination for biofuel producers [7], pricing in competitive electricity markets [9], supply chain planning [20] and so forth.

However, processing time of solution methods for noncooperative two-level linear programming problems, for example, Kib best method by Bialas et al. [2] and Branch-and-Bound method by Hansen et al. [11], may exponentially increases at worst as the size of problem increases since they are strict solution methods through enumeration. In order to obtain the (approximate) Stackelberg solution with a practically reasonable time, approximate solution methods were presented through particle swarm optimization (PSO) [19], and evolutionary multi-agent systems (EMAS) [12].

In this paper, we propose a new efficient PSO-based computational method for obtaining (approximate) Stackelberg solutions to two-level nonlinear programming problems.

2. Two-level programming problems

In this paper, we consider two-level programming problems formulated as follows:
minimize \( f_1(x_1, x_2) \)
minimize \( f_2(x_1, x_2) \)
subject to \( g_i(x_1, x_2) \leq 0, \ i = 1, \ldots, m \) \( x_1 \in R^{n_1}, x_2 \in R^{n_2} \) \( (1) \)

where \( x_1 \) is an \( n_1 \) dimensional decision variable column vector for the DM at the upper level (DM1), \( x_2 \) is an \( n_2 \) dimensional decision variable column vector for the DM at the lower level (DM2), \( f_i(x_1, x_2) \) is the objective function for DM1, \( g_i(x_1, x_2) \) is the objective function for DM2 and \( g_i(x_1, x_2) \), \( i = 1, 2, \ldots, m \) are constraint functions. In general, \( f_i() \), \( i = 1, 2, \ldots, m \) are nonlinear. In (1), if the DM at the upper level (DM1) adopts a decision \( x_1 \), the DM at the lower level (DM2) is supposed to select a decision to minimize \( f_2() \) in the feasible region of (1) under the DM1’s decision, \( x_2(x_1) \), called a rational response. Then, the optimal solution (Stackelberg solution) to (1) is the point \( (x_1^*, x_2^*(x_1^*)) \) which minimizes \( f_2() \) in the inducible region (IR) which is the set of points \( (x_1, x_2(x_1)) \) for all possible decisions \( x_1 \). Fig. 1 illustrates an example of a Stackelberg solution for a two-level linear programming problem.

![Stackelberg solution](image)

**Fig. 1. An example of Stackelberg solution**

### 3. Particle swarm optimization (PSO)

A particle swarm optimization [13] method is based on the social behavior that a population of individuals adapts to its environment by returning to promising regions that were previously discovered. 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 is characterized by the position vector consisting of three attributes such as the current direction vector, the best position in its track and the best position in the swarm. The fundamental procedures of particle swarm optimization can be summarized as follows:

**Step 1:** Generate the initial swarm involving \( N \) particles at random.

**Step 2:** Calculate the new direction vector for each particle based on its attributes.

**Step 3:** Calculate the new search position of each particle from the current search position and its new direction vector.

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

In the procedures of particle swarm optimization, as introduced by Shi and Eberhart [23], the new direction vector of the \( i \)-th particle at time \( t \), \( v_{i}^{t+1} \), is calculated by the following scheme:

\[
v_{i}^{t+1} := \omega v_{i}^{t} + c_1 R_1 (p_i^t - x_i^t) + c_2 R_2 (p_{g_i}^t - x_i^t) \quad (2)
\]

Here, \( R_1 \) and \( R_2 \) are random numbers between 0 and 1. \( p_i^t \) is the best position of the \( i \)-th particle in its track at time \( t \) and \( p_{g_i}^t \) is the best position of the swarm at time \( t \). There are three parameters such as the inertia of the particle \( \omega \), and two parameters \( c_1 \), \( c_2 \).

Then, the new position of the \( i \)-th particle at time \( t \), \( x_{i}^{t+1} \) is calculated as:

\[
x_{i}^{t+1} := x_{i}^{t} + v_{i}^{t+1} \quad (3)
\]

where \( x_{i}^{t} \) is the current position of the \( i \)-th particle at time \( t \). After the \( i \)-th particle calculates the next search direction vector \( v_{i}^{t+1} \) by (2) 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 search position in its track \( p_i^t \) and the direction vector going from the current search position \( x_{i}^{t} \) to the best search position of the swarm \( p_{g_i}^t \), it moves from the current position \( x_{i}^{t} \) to the next search position \( x_{i}^{t+1} \) calculated by (3). In general, the parameter \( \omega \) is set to large values in the early stage for global search, while it is set to small val-
ues in the late stage for local search. For example, it is determined as:

$$
\omega^t = \begin{cases} 
\omega^0 - t \cdot (\omega^0 - \omega^T_{\text{max}}) / 0.75 \cdot T_{\text{max}} & t \leq 0.75 \cdot T_{\text{max}} \\
\omega^T_{\text{max}}, & t > 0.75 \cdot T_{\text{max}}
\end{cases}
$$

(4)

where \( t \) is the current time, \( T_{\text{max}} \) is the maximal value of time, \( \omega^0 \) is the initial value of \( \omega \) and \( \omega^T_{\text{max}} \) is the final value of \( \omega \).

If the next search position of the \( i \)-th particle at time \( t+1 \), \( \mathbf{x}^{t+1} \), is better than the best search position in its track at time \( t \), \( \mathbf{p}^t_i \), i.e., \( f(\mathbf{x}^{t+1}_i) \leq f(\mathbf{p}^t_i) \), the best search position in its track is updated as \( \mathbf{p}^{t+1}_i := \mathbf{x}^{t+1}_i \).

Otherwise, it is updated as \( \mathbf{p}^{t+1}_i := \mathbf{p}^t_i \). Similarly, if \( p^{t+1}_{g'} \) is better than the best position of the swarm, \( p^t_{g'} \), i.e., \( f(p^{t+1}_{g'}) \leq f(p^t_{g'}) \), then the best search position of the swarm is updated as \( p^{t+1}_{g'} := p^{t+1}_{g'} \).

Otherwise, it is updated as \( p^{t+1}_{g'} := p^t_{g'} \).

Unfortunately however, it should be emphasized here that, in the original PSO method, there are drawbacks that it is not directly applicable to constrained problems and it is liable to stopping around local solutions.

4. PSO for two-level nonlinear programming problems

This section devotes to introducing some basic ideas of a new PSO for two-level nonlinear programming problems. In applying the original PSO directly to the nonlinear case, there often occur two problems; one is that it is difficult to obtain feasible particles for nonlinear programming problems, and the other is that the search is liable to stopping at a certain local solution.

In order to resolve the former problem, we incorporate the ideas of homomorphous mapping used in [14], a bisection method into the proposed method in order to obtain feasible particles.

On the other hand, for tackling the latter problem, in order to widen the search area, a multiple stretching and secession are introduced.

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 is not accepted, if a particle becomes infeasible after move, it is repaired to be feasible. In the other subswarm, the move of particle to the infeasible region is accepted.

The procedure of the proposed PSO-based algorithm is summarized as follows.

Step 1: Generate \( N \) particles by using the homomorphous mapping [14]. Let initial search position of each particle be \((\mathbf{x}^0_{i,1}, \mathbf{x}^0_{i,2})\).

Step 2: Calculate the (approximate) rational response \( \mathbf{x}^0_{i,2} \) for \( \mathbf{x}^0_{i,1} \) by rPSO [13]. Let the initial best position of the particle \( \mathbf{p}^0_i := (\mathbf{x}^0_{i,1}, \mathbf{x}^0_{i,2}) \) and let the best position among \((\mathbf{x}^0_{i,1}, \mathbf{x}^0_{i,2}), i=1, ..., N\) be the initial best position of the swarm \( \mathbf{p}^0_{g'} \).

Step 3: Let \( t := 0 \).

Step 4: Let \( \mathbf{p}^{t+1}_{i,1} := \mathbf{p}^t_i, i=1, ..., N, \mathbf{p}^{t+1}_{g'} := \mathbf{p}^t_{g'} \).

Step 5: Calculate the value of \( \omega \) by (4). For each particle, using the information of \( \mathbf{p}^t_i \) and \( \mathbf{p}^t_{g'} \), determine the direction vector \( \mathbf{v}^t_{i,1} \) to the next search position \( \mathbf{x}^{t+1}_{i,1} \) by the modified move schemes explained in [13]. Next, move it to the next search position by (3) and go to Step 6.

Step 6: Evaluate each particle by the objective function value for \( \mathbf{x}^{t+1}_{i,1}, i=1, ..., N \). Determine whether \( t := 100 \cdot T_{\text{step}} \), \( T_{\text{step}} = 1, ..., T_{\text{max}}/100 \) or not. If it is, go to Step 7. Otherwise, go to Step 8.

Step 7: Update the rational response (the best position of the particle \( \mathbf{p}^t_i := (\mathbf{x}^t_{i,1}, \mathbf{x}^t_{i,2}) \)) and go to Step 8.

Step 8: Stop if \( t = T_{\text{max}} \) (the maximal value of time). Otherwise, let \( t := t+1 \) and return to Step 3.

5. Numerical experiments

In order to demonstrate the feasibility and efficiency of the proposed method, we compare the performance of the proposed PSO for obtaining Stackelberg solutions (SPO) with that of EMAS [12] and PSO [19] through the experimental results for 3 two-level nonlinear programming problems. The number of trial is 40 for SPO, EMAS and PSO. Tab. 1-3 show the results obtained by these methods: the best objective function value of 40 trials, the average one, the worst one, the worst one and the average computational time. In these experiments, the parameters of EMAS are set as the number of agent = 1000. On the other hand, the parameters of SPO and
PSO are set as the size of the swarm = 70, the maximal value of time $T_{\text{max}} = 5000$ for all problems.

**Tab.1. Results for a problem with $n_1 = 1$, $n_2 = 1$ and $m = 2$**

<table>
<thead>
<tr>
<th>method</th>
<th>SPSO</th>
<th>EMAS</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>62.31249</td>
<td>62.31249</td>
<td>62.31249</td>
</tr>
<tr>
<td>average</td>
<td>62.31249</td>
<td>62.31250</td>
<td>62.31249</td>
</tr>
<tr>
<td>worst</td>
<td>62.31250</td>
<td>62.31250</td>
<td>62.31250</td>
</tr>
<tr>
<td>time (sec)</td>
<td>11.15</td>
<td>14.70</td>
<td>441.73</td>
</tr>
</tbody>
</table>

**Tab.2. Results for a problem with $n_1 = 5$, $n_2 = 5$ and $m = 6$**

<table>
<thead>
<tr>
<th>method</th>
<th>SPSO</th>
<th>EMAS</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>-210.00000</td>
<td>-199.51502</td>
<td>-189.16253</td>
</tr>
<tr>
<td>average</td>
<td>-209.99999</td>
<td>-196.16092</td>
<td>-186.83158</td>
</tr>
<tr>
<td>worst</td>
<td>-209.99999</td>
<td>-192.90440</td>
<td>-184.67067</td>
</tr>
<tr>
<td>time (sec)</td>
<td>74.148</td>
<td>94.303</td>
<td>1357.313</td>
</tr>
</tbody>
</table>

**Tab.3. Results for a problem with $n_1 = 4$, $n_2 = 4$ and $m = 3$**

<table>
<thead>
<tr>
<th>method</th>
<th>SPSO</th>
<th>EMAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>442.5933626</td>
<td>461.087664</td>
</tr>
<tr>
<td>average</td>
<td>442.5933815</td>
<td>463.110938</td>
</tr>
<tr>
<td>worst</td>
<td>442.5933981</td>
<td>466.620022</td>
</tr>
<tr>
<td>time (sec)</td>
<td>73.226</td>
<td>149.3074</td>
</tr>
</tbody>
</table>

For the first problem, as shown in Tab. 1, the average computational time of SPSO is shorter than that of EMAS and PSO. For the second problem, as shown in Tab. 2, SPSO is better than EMAS and PSO with respect to the best objective function value, the average one, the worst one and the average computational time. For the third problem, as shown in Tab. 3, the proposed PSO (SPSO) is better than EMAS with respect to the best objective function value, the average one, the worst one and the average computational time.

From these results, SPSO is very consistent in locating the area of a global optimal solution; in these cases the difference between the values of the objective function at the global solution and the best solution found was due to the shape of the landscape. It is indicated that SPSO is superior to EMAS and PSO, and that SPSO is promising as a computational method for obtaining (approximate) Stackelberg solutions to nonlinear two-level programming problems.

**6.Conclusion**

In this paper, focusing on two-level nonlinear programming problems with no coordination, we proposed a new computational method through particle swarm optimization for obtaining (approximate) Stackelberg solutions. The feasibility and efficiency of the proposed method were demonstrated by comparing with the performance of existing approximation method, EMAS and PSO, through the numerical experiments. Extensions of the proposed method to multi-level programming, actual optimization problems and so forth will be considered in the near future.

**References**


Takeshi MATSUI
He received the M.E degree in management information from Hiroshima Prefectural University in 2005, and the Dr. degree of engineering from Hiroshima University in 2008. He has been working as a research assistant at Hiroshima University. His present research interests include the soft computing techniques and their applications.

Masatsushi SAKAWA
He received the M.E degree in electrical engineering from Kyoto University in 1972, and the Dr. degree of engineering in 1975. He has worked as a prof. of faculty of engineering at Hiroshima University. He worked as an assistant prof. of faculty of engineering at Kobe University from 1975 to 1981, and worked as an associate prof. of faculty of engineering at Kobe University from 1981 to 1987, and worked as a prof. of faculty of engineering at Iwate University from 1987 to 1990. His present research interests include mathematical programming under uncertain environments.