5

Stackelberg Solutions to Noncooperative Two-Level Nonlinear Programming Problems through Evolutionary Multi-Agent Systems

Masatoshi Sakawa, Hideki Katagiri and Takeshi Matsui
Faculty of Engineering, Hiroshima University
Japan

1. Introduction

In the real world, we often encounter 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 (Simaan & Cruz Jr., 1973), and they are formulated as two-level mathematical programming problems (Shimizu et al, 1997; Sakawa & Nishizaki, 2009). 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 (Bialas & Karwan, 1984), the Kuhn-Tucker approach (Bard & Falk, 1982; Bard & Moore, 1990; Bialas & Karwan, 1984; Hansen et al, 1992), and the penalty function approach (White & Anandalingam, 1993). The subsequent works on two-level programming problems under noncooperative behavior of the decision makers have been appearing (Nishizaki & Sakawa, 1999; Nishizaki & Sakawa, 2000; Gumus & Floudas, 2001; Nishizaki et al., 2003; Colson et al., 2005; Faisca et al., 2007) including some applications to aluminium production process (Nicholls, 1996), pollution control policy determination (Amouzegar & Moshirvaziri, 1999), tax credits determination for biofuel producers (Dempe & Bard, 2001), pricing in competitive electricity markets (Fampa et al, 2008), supply chain planning (Roghanian et al., 2007) and so forth.
However, processing time of solution methods for noncooperative two-level linear programming problems, for example, Kth Best method by Bialas et al. (1982) and Branch-and-Bound method by Hansen et al. (1992), may exponentially increases at worst as the size of problem increases since they are strict solution methods based on enumeration. In order to obtain the (approximate) Stackelberg solution with a practically reasonable time, approximate solution methods were presented through genetic algorithms (Niwa et al., 1999) and particle swarm optimization (PSO) (Niwa et al., 2006).

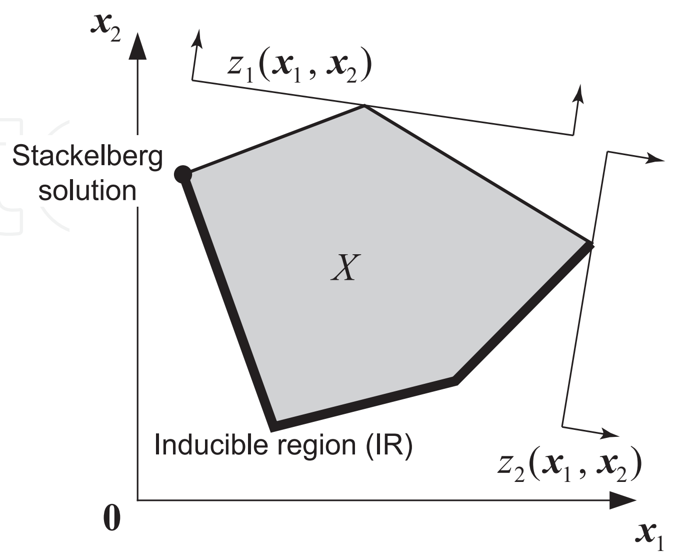
As one of the most promising approximate solution methods, Socha et al. (2002) proposed a fast computational method through an evolutionary multi-agent system (EMAS) for obtaining (approximate) Pareto optimal solution sets for multiobjective programming problems. However, there is no study on the EMAS-based method for solving two-level nonlinear programming problems.
In this chapter, we propose an efficient EMAS-based computational method for obtaining (approximate) Stackelberg solutions to two-level nonlinear programming problems.

2. Two-level programming problems and solution concepts

In this chapter, we consider two-level programming problems formulated as follows:

$$\left.\begin{array}{l} \underset{x_1}{\text{minimize}}\ f_1(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ \quad\text{where } \boldsymbol{x}_2 \text{ solves} \\ \underset{x_2}{\text{minimize}}\ f_2(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ \text{subject to } g_i(\boldsymbol{x}_1, \boldsymbol{x}_2) \le 0,\ i = 1, 2, \ldots, m \end{array}\right\} \tag{1}$$

where $\boldsymbol{x}_1$ is an n_1 dimensional decision variable column vector for the DM at the upper level (DM1), $\boldsymbol{x}_2$ is an n_2 dimensional decision variable column vector for the DM at the lower level (DM2), $f_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is the objective function for DM1, $f_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is the objective function for DM2 and $g_i(\boldsymbol{x}_1, \boldsymbol{x}_2)$, i=1, 2,…, m are constraint functions. In general, $f_l(\cdot)$, l=1,2 and $g_i(\cdot)$, i=1,2, …, m are nonlinear. In (1), if the DM at the upper level (DM1) adopts a decision $\boldsymbol{x}_1$, the DM at the lower level (DM2) is supposed to select a decision to minimize $f_2(\cdot)$ in the feasible region of (1) under the DM1's decision, $\hat{x}_2(\hat{x}_1)$, called a rational reaction. Then, the optimal solution (Stackelberg solution) to (1) is the point $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*(\boldsymbol{x}_1^*))$ which minimizes $f_1(\cdot)$ in the inducible region (IR) which is the set of points $(\hat{x}_1, \hat{x}_2(\hat{x}_1))$ for all possible decisions $\hat{x}_1$. Figure 1 illustrates an example of a Stackelberg solution for a two-level linear programming problem.



$\boldsymbol{x}_2$
$z_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$
Stackelberg solution
X
Inducible region (IR)
$z_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
0
$\boldsymbol{x}_1$

Fig. 1. An example of Stackelberg solution

3. EMAS for two-level programming problems

In this section, we outline the framework of a computational method through EMAS for obtaining Stackelberg solutions to two-level programming problems.

In general, EMAS-based methods consist of N agents $\boldsymbol{a}_r$, $r = 1, 2, \ldots, N$, each of which is characterized by some attributes, i.e., the current position $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$, the upper level objective function value $f_1(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$, the lower level objective function value $f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$, the upper level energy e_r^U and the lower level energy e_r^L. In the EMAS-based method, the attributes of agents are updated through the evolutionary process with some operations like energy exchange, reproduction and move.

First, carry out the search to the direction of improving the lower level objective function by moving each agent toward IR. To be more specific, in the upper level decision variable space, if there exists at least one agent in the neighborhood of each agent $\boldsymbol{a}_r$, select one of agents, $\boldsymbol{a}_{r'}$, as a communicating opponent, and compare the lower level objective function value of $\boldsymbol{a}_r$ with that of $\boldsymbol{a}_{r'}$. Then, the superior agent gains the lower level energy from the inferior one. The inferior agent is killed if the lower level energy of it becomes empty. Otherwise, it is moved according to some rule whose details are described later in Section 4. If there exists no agent in the neighborhood of $\boldsymbol{a}_r$, move $\boldsymbol{a}_r$ to the position where the lower level objective function value becomes better by changing $\boldsymbol{x}_{r2}$.

Next, carry out the search to the direction of improving the upper level objective function by moving each agent near IR toward a Stackelberg solution. For each agent $\boldsymbol{a}_r$, after selecting an agent, $\boldsymbol{a}_{r''}$, which is the nearest agent around IR as a communicating opponent, compare the upper level objective function value of $\boldsymbol{a}_r$ with that of $\boldsymbol{a}_{r''}$. Then, the superior agent gains the upper level energy from the inferior one. The inferior agent is not killed even if its lower level energy of it becomes empty. It gains the same amount of the upper level energy as the initial value. This supplement is done to maintain the number of agents with nonzero upper level energy. After exchanging the upper level energies between the superior and inferior agents, if the upper level energy of the inferior agent is sufficiently large, generate a new agent in the direction from the inferior one to the superior one.

By repeating these procedures, agents with large upper level objective function values congregate around IR, which means the current position of the agent with the largest upper level objective function value can be regarded as the (approximate) Stackelberg solution.

The procedure is summarized as follows.

Step 1. Generate N agents $\boldsymbol{a}_r$, $r = 1,2, \ldots, N$ at random.

Step 2. Let $T := 1$.

Step 3. Let $r := 1$.

Step 4. For the rth agent $\boldsymbol{a}_r$, carry out the search to the direction of improving the lower level objective function value in order to move the current position of the agent toward IR.

Step 5. If the lower level energy of the rth agent $\boldsymbol{a}_r$ is greater than a threshold, i.e., the current position of the agent is regarded as being in IR, carry out the search to the direction of improving the upper level objective function value so that the agent is moved toward the Stackelberg solution.

Step 6. If $r = N$, go to step 7. Otherwise, let $r := r+1$ and return to step 4.

Step 7. If $T = T_{\max}$, terminate the procedure and the current best solution is regarded as a Stackelberg solution. Otherwise, let $T := T+1$ and return to step 3.

4. EMAS for two-level nonlinear programming problems

This section devotes to introducing some basic ideas of a new EMAS for two-level nonlinear programming problems. In applying the original EMAS directly to the nonlinear case, there often occur two problems; one is that it is difficult to obtain feasible initial agents generated randomly in step 1, and the other is that the judgment of the existence of an agent around IR by the amount of the lower level energy is insufficient in most cases since the shape of IR for nonlinear cases is fairly complicated in general than that for the linear case.

In order to resolve the former problem, we incorporate the idea of homomorphous mapping used in (Koziel & Michalewicz, 1999) into the proposed method in order to generate feasible initial agents. In addition, in order to widen the search area, we permit the reproduction in the infeasible region together with the search of the infeasible region by infeasible agents.

On the other hand, for tacking the latter problem, we utilize the Kuhn-Tucker condition of problem (2), which is the necessary condition for the current position to be in IR, in order to obtain the rational reaction $\bar{x}_2(\bar{x}_1)$ corresponding to $\bar{x}_1$ for the purpose of more accurately check whether an agent with the current position ($\bar{x}_1$, $\bar{x}_2$) exists around IR or not.

$$\left.\begin{array}{ll}\text{minimize} & f_2(\bar{x}_1, x_2)\\ \text{subject to} & g_i(\bar{x}_1, x_2) \le 0,\ i=1, 2, \dots, m\end{array}\right\} \tag{2}$$

The Kuhn-Tucker condition of (2) is expressed as follows.

$$\nabla f_2\left(\bar{x}_1, x_2^*\right) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i\left(\bar{x}_1, x_2^*\right) = 0, \lambda^* \ge \mathbf{0},$$

$$g_i\left(\bar{x}_1, x_2^*\right) \le 0,\ \lambda_i^* g_i\left(\bar{x}_1, x_2^*\right) = 0, i = 1, 2, \dots, m$$

It should be noted here that the best position $(\tilde{x}_1, \tilde{x}_2)$ obtained through the search by EMAS does not always exist in IR even if it satisfies the Kuhn-Tucker condition. In order to find a rational reaction for the upper level decision $\tilde{x}_1$ of the best solution, the following problem with respect to the lower level objective function is solved.

$$\left.\begin{array}{ll}\text{minimize} & f(\tilde{x}_1, x_2)\\ \text{subject to} & g_i(\tilde{x}_1, x_2) \le 0,\ i=1, 2, \dots, m\end{array}\right\} \tag{3}$$

Since (3) is a single-objective nonlinear programming problem, we can solve the problem by using an approximate solution method based on PSO (Matsui et al, 2007) to check whether the (approximate) optimal solution (x^*_2) is equal to $\tilde{x}_2$ or not. If (x^*_2) = $\tilde{x}_2$, we can regard $(\tilde{x}_1, \tilde{x}_2)$ as the (approximate) Stackelberg solution since it exists in IR. Otherwise, repeatedly check whether the current position of an agent satisfying Kuhn-Tucker condition exists in IR or not in the same manner mentioned above in order of the quality of the upper level objective function value. If an agent whose current position exists in IR, the position is regarded as the (approximate) Stackelberg solution.

5. Detailed procedure of the proposed EMAS

In this section, we describe the details of the computational procedures in the proposed EMAS for obtaining (approximate) Stackelberg solutions to noncooperative two-level programming problems.

In the proposed EMAS-based method, we use N agents $\boldsymbol{a}_r$, r=1, 2 …, N, each of which is characterized by some attributes, i.e., the current position ($\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$), the upper level objective function value $f_1(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$, the lower level objective function value $f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$, the energy e_r and the agent state variable k_r. If Kuhn-Tucker condition is satisfied for the rth agent $\boldsymbol{a}_r$ whose current position is in the feasible region, let k_r=1. If not, let k_r=0. For agents in the infeasible region, let k_r<0.

The procedure of the proposed EMAS-based method is summarized as follows.

Step 1. Generate N agents $\boldsymbol{a}_r$, r=1, 2, …, N by using the homomorphous mapping (Koziel & Michalewicz, 1999).

Step 2. Let T := 1.

Step 3. Let r := 1.

Step 4. If k_r=0, go to step 5. If k_r=1, go to step 6. If k_r<0, go to step 8.

Step 5. Carry out the search to the direction of improving the lower level objective function value. To be more specific, for the rth agent $\boldsymbol{a}_r$, choose an agent $\boldsymbol{a}_{r'}$ randomly. Then, compare $f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$ with $f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r'2})$. If $f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}) < f_2(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r'2})$, let $\boldsymbol{x}_{p2} := \boldsymbol{x}_{r2}$ and $\boldsymbol{x}_{p'2} := \boldsymbol{x}_{r'2}$. Otherwise, let $\boldsymbol{x}_{p2} := \boldsymbol{x}_{r'2}$ and $\boldsymbol{x}_{p'2} := \boldsymbol{x}_{r2}$.
Then, update $\boldsymbol{x}_{p'2}$ by the following scheme:

$$\boldsymbol{x}_{p'2} := \boldsymbol{x}_{p'2} + 2R\left(\boldsymbol{x}_{p2} - \boldsymbol{x}_{p'2}\right) \tag{4}$$

where R is a uniform random number in [0,1]. Repeat the comparison between $f_2(\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2})$ and $f_2(\boldsymbol{x}_{p'1}, \boldsymbol{x}_{p'2})$ and the update of $\boldsymbol{x}_{p'2}$ n times. Let the final $\boldsymbol{x}_{p2}$ be $\boldsymbol{x}_{r2}$. If the current position of $\boldsymbol{a}_r$ satisfies Kuhn-Tucker condition, let k_r:=1 and go to step 6. Otherwise, go to step 9.

Step 6. If k_r=1, carry out the search to the direction of improving the upper level objective function value. To be more specific, choose an agent $\boldsymbol{a}_{r''}$ which satisfies Kuhn-Tucker condition at random as a comparing opponent and compare the upper level objective function value of $\boldsymbol{a}_r$ with that of $\boldsymbol{a}_{r''}$. Then, the superior agent gains the energy from the inferior one. Let the superior agent denote $\boldsymbol{a}_{r''}$ and the inferior one denote $\boldsymbol{a}_r$. If the inferior agent is killed by the disappearance of energy, go to step 7. Otherwise, go to step 9.

Step 7. Carry out the reproduction of the killed agent. Then, reproduce $\boldsymbol{a}_r$ with the current position which is determined as:

$$\left(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}\right) := \left(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}\right) + 2R\left(\left(\boldsymbol{x}_{r''1}, \boldsymbol{x}_{r''2}\right) - \left(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}\right)\right). \tag{5}$$

where R is a uniform random number in [0,1]. If the current position of the reproduced $\boldsymbol{a}_r$ is infeasible, let k_r:=-1 and go to step 8. Otherwise, let k_r:=0 and go to step 9.

Step 8. Carry out the search in the infeasible region. If $-t_1 \le k_r < 0$, go to substep 8-1. If $-t_2 \le k_r < -t_1$, go to substep 8-2. If $kr = -t_2$, go to substep 8-3. Here, let $0<t_1<t_2$.
Substep 8-1: Let an agent with the current position ($x_{r,\,1}$, $x_{r,\,2}$, …, $x_{r^*,\,i}$, …, $\mathrm{x}_{r,\,n2}$) denote $\boldsymbol{a}_{r'}$. Here, r^* is randomly chosen from among {1, 2, …, N} and i is randomly

chosen from among {1, 2, …, n_1+n_2}. Compare $f_1(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$ with $f_2(\boldsymbol{x}_{r'1}, \boldsymbol{x}_{r'2})$. Let the superior agent and the inferior one denote $\boldsymbol{a}_s$ and $\boldsymbol{a}_{s'}$, respectively. Then, update $(\boldsymbol{x}_{s'1}, \boldsymbol{x}_{s'2})$ by the following scheme:

$$\left(\boldsymbol{x}_{s'1},\boldsymbol{x}_{s'2}\right) = \left(\boldsymbol{x}_{s'1},\boldsymbol{x}_{s'2}\right) + 2R\left(\left(\boldsymbol{x}_{s1},\boldsymbol{x}_{s2}\right) - \left(\boldsymbol{x}_{s'1},\boldsymbol{x}_{s'2}\right)\right) \tag{6}$$

where R is a uniform random number in [0,1]. Repeat the comparison between $f_1(\boldsymbol{x}_{s1}, \boldsymbol{x}_{s2})$ and $f_1(\boldsymbol{x}_{s'1}, \boldsymbol{x}_{s'2})$ and the update of $(\boldsymbol{x}_{s'1}, \boldsymbol{x}_{s'2})$ n' times. Let the final $(\boldsymbol{x}_{s1}, \boldsymbol{x}_{s2})$ be $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$. If $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$ is feasible, let k_r:=0. Otherwise, let k_r:=k_r-1. Go to step 9.
Substep 8-2: Choose an agent $\boldsymbol{a}_{r'}$ randomly. Compare $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$ with $(\boldsymbol{x}_{r'1}, \boldsymbol{x}_{r'2})$ using the following function

$$h\left(\boldsymbol{x}_1,\boldsymbol{x}_2\right) = \sum_{i=1}^{m} g_i\left(\boldsymbol{x}_1,\boldsymbol{x}_2\right), g_i\left(\boldsymbol{x}_1,\boldsymbol{x}_2\right) > 0$$

which is the degree of violation of the constraints. Let the superior position denote $(\boldsymbol{x}_{u1}, \boldsymbol{x}_{u2})$ and the inferior one denote $(\boldsymbol{x}_{u'1}, \boldsymbol{x}_{u'2})$. Then, update $(\boldsymbol{x}_{u'1}, \boldsymbol{x}_{u'2})$ by the following scheme:

$$\left(\boldsymbol{x}_{u'1},\boldsymbol{x}_{u'2}\right) = \left(\boldsymbol{x}_{u'1},\boldsymbol{x}_{u'2}\right) + 2R\left(\left(\boldsymbol{x}_{u1},\boldsymbol{x}_{u2}\right) - \left(\boldsymbol{x}_{u'1},\boldsymbol{x}_{u'2}\right)\right)$$

where R is a uniform random number in [0,1]. Repeat the comparison between $h(\boldsymbol{x}_{u1}, \boldsymbol{x}_{u2})$ and $h(\boldsymbol{x}_{u'1}, \boldsymbol{x}_{u'2})$ and the update of $(\boldsymbol{x}_{u'1}, \boldsymbol{x}_{u'2})$ n'' times. Let the final $(\boldsymbol{x}_{u1}, \boldsymbol{x}_{u2})$ be $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$. If $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2})$ is feasible, let k_r:=0. Otherwise, let k_r:=k_r-1. Go to step 9.
Substep 8-3: Choose an agent $\boldsymbol{a}_{r'}$ whose current position is feasible randomly, and move $\boldsymbol{a}_r$ to the feasible region by the bisection method between $\boldsymbol{a}_{r'}$ and $\boldsymbol{a}_r$. Go to step 9.

Step 9. If $r = N$, go to step 10. Otherwise, let $r := r+1$ and return to step 4.
Step 10. If $T = T_{\max}$, go to step 11. Otherwise, let $T := T+1$ and return to step 3.
Step 11. Check whether the current position of the best agent exists in IR or not by solving (3) through the revised PSO method (Matsui et al., 2007), which is one of most promising solution methods for nonlinear programming problems. If the current position of the best agent exists in IR, we can regard it as the (approximate) Stackelberg solution. Otherwise, repeatedly check whether the current position of an agent satisfying Kuhn-Tucker condition exists in IR or not in the same manner mentioned above in order of the quality of the upper level objective function value. If an agent whose current position satisfies Kuhn-Tucker condition and exists in IR, the position is regarded as the (approximate) Stackelberg solution, and the solution procedure is terminated.

6. Numerical examples

In order to investigate the efficiency of the proposed method, we conduct some numerical experiments.
First, we consider a two-level nonlinear programming problem with 4 decision variables and 8 constraints (P1) and one with 6 decision variables and 10 constraints (P2), and compare the computational time of generating the initial population including 1000 agents by the homomorphous mapping (Koziel & Michalewicz, 1999) with that by the random method. The results are shown as in table 1.

	Computational time (sec.)	
	P1	P2
Homomorphous mapping	1.500	3.109
Random method	177.650	2691.710

Table 1. Comparison of computational times of generating the initial population including 1000 agents

The results in table 1 show the effectiveness of the use of the homomorphous mapping in generating the initial population.
Second, in order to investigate the efficiency of substeps 8-2 and 8-3, we compare the result of EMAS without 8-2, 8-3 with that of EMAS with 8-2, 8-3 by setting both the number of agents and the maximal generation number to 1000s. Results are shown as in table 2.

	Upper level objective function	Lower level Objective function
EMAS without 8-2 and 8-3	-14.999991	0.999997
EMAS with 8-2 and 8-3	-24.0	0.0
Optimal value	-24.0	0.0

Table 2. The efficiency of substeps 8-2 and 8-3

The results in table 2 show that both substeps 8-2 and 8-3 are worth being introduced in the proposed method.
Next, in order to investigate the efficiency of substep 8-1 for improving the upper level objective function value, we apply EMAS without 8-1 and one with 8-1.

	Upper level objective function	
	Problem A (8 decision variables)	Problem B (20 decision variables)
EMAS without 8-1	453.620022	-0.202166
EMAS with 8-1	452.087664	-0.286156

Table 3. The efficiency of substeps 8-1

To be more specific, problem A and B are formulated as:
Problem A: Upper level decision variables $\boldsymbol{x}_1 = (x_1, x_2, x_3, x_4)$, lower level decision variables: $\boldsymbol{x}_2 = (x_5, x_6, x_7, x_8)$.

$$\begin{array}{ll}
\underset{\boldsymbol{x}_1}{\text{minimize}} & f_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = x_1^3+(x_2-2)^2-x_3x_4-2x_5^2+x_6^4+(x_7-x_8)^2 \\
& \quad \text{where } \boldsymbol{x}_2 \text{ solves} \\
\underset{\boldsymbol{x}_2}{\text{minimize}} & f_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = -2x_1^2+x_2^3+2x_3^2-x_4^4+3x_5^3-2x_6+5x_7+4x_8^2 \\
\text{subject to} & 4x_1^2-3x_2^3+5(x_3-4)^2-6x_4-x_5x_6-3x_7^4+5x_8 \le 0 \\
& 2x_1x_7-x_2^2-3x_3-4x_4^2+x_5^2-x_6^3-2x_8^2 \le -12 \\
& 5x_1+x_2-6x_3+4x_4-6x_5-x_6-3x_7+x_8 \le 5 \\
& -5 \le x_j \le 5,\ j = 1, \ldots, 8,
\end{array}$$

Problem B: Upper level decision variables $\boldsymbol{x}_1 = (x_1, \ldots, x_{10})$, lower level decision variables $\boldsymbol{x}_2 = (x_{11}, \ldots, x_{20})$.

$$\underset{\boldsymbol{x}_1}{\text{minimize}} \quad f_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = -\left| \frac{\sum_{j=1}^{20} cos^4(x_j) - 2\prod_{j=1}^{20} cos^2(x_j)}{\sqrt{\sum_{j=1}^{20} j x_j^2}} \right|$$

where $\boldsymbol{x}_2$ solves

$$\underset{\boldsymbol{x}_2}{\text{minimize}} \quad f_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = (x_1-x_{14})(x_{17}-x_6)-(x_4-x_{11})(x_{18}-x_7)+(x_8-x_{12})(x_5-x_{19}) -(x_{13}-x_3)(x_{10}-x_{16})+(x_{20}-x_9)(x_2-x_{15})$$

$$\text{subject to} \quad 0.75-\prod_{j=1}^{20} x_j \le 0$$

$$\prod_{j=1}^{20} x_j -7.5 \cdot 20 \le 0$$

$$0 \le x_j \le 10, j = 1, \ldots, 20.$$

From the results in table 3, the procedure of substep 8-1 is meaningful for enhancing the efficiency of the proposed method.
Furthermore, in order to investigate the efficiency of the proposed EMAS, we compare the result obtained by it with that by two-level PSO method (Niwa et al., 2006) in the application of both methods to a two-level nonlinear programming problem with 10 decision variables and 3 constraints. In the numerical experiment, the number of agents is 1000, the maximal generation number is 1000 and the number of trials is 10. Table 4 shows the best value, the average value, the worst value of the upper level objective function obtained by the proposed EMAS in 10 trials, the best value obtained by the two-level PSO (Niwa et al., 2006) and the average computational time.

	Upper level objective function value	Computational time (sec.)
Best	-199.515027	94.3028
Average	-196.160929	
Worst	-192.904405	
PSO	-184.600761	1357.313

Table 4. Comparison of the proposed EMAS with two-level PSO method

Table 4 shows that the proposed EMAS is superior to PSO because the best value obtained by PSO is worse than the worst value of the proposed EMAS.
Finally, table 5 shows the effect of the number of agents and the number of generations on computational time. Table 5 shows the computational time of EMAS linearly increases as the number of agents and the number of generations.

	The number of agents		
	1000	2000	3000
Generation 1000	165.375	330.500	489.343
Generation 2000	344.203	689.984	1034.234
Generation 3000	525.718	1054.718	1587.437

Table 5. Effect of the number of agents and the number of generations on computational time (sec)

7. Conclusion

In this chapter, we discussed an efficient approximate solution method based on evolutionary multi-agent systems to obtain Stackelberg solutions to noncooperative two-level programming problems. In particular, we proposed a new EMAS by incorporating the concept of homomorphous mapping to generate feasible initial agents, the theory of the Kuhn-Tucker condition for checking whether an agent exists around IR or not, and the idea of reproduction in the infeasible region, together with the introduction of the searching process of the infeasible region by infeasible agents in order to widen the search area. Furthermore, we showed the efficiency of the proposed EMAS by comparing it with an existing method, the two-level PSO method, through some numerical experiments. From the numerical experimental results, it is indicated that the proposed EMAS is superior to the two-level PSO method, and that the proposed EMAS is promising as an optimization method for two-level nonlinear programming problems. In the near future, we will extend the proposed method to noncooperative and cooperative multi-level programming.

8. References

M. A. Amouzegar & K. Moshirvaziri. (1999). Determining optimal pollution control policies: an application of bilevel programming, *European Journal of Operational Research*, Vol. 119, No. 1, pp. 100--120

W. F. Bialas & M. H. Karwan. (1982). On two-level optimization, *IEEE Transactions on Automatic Control*, Vol. AC-27, No. 1, pp. 211–214

W. F. Bialas & M. H. Karwan. (1984). Two-level linear programming, *Management Science*, Vol. 30, No. 8, pp. 1004—1020

J. F. Bard & J. E. Falk. (1982). An explicit solution to the multi-level programming problem, *Computer and Operations Research*, Vol. 9, No. 1, pp. 77—100

J. F. Bard & J. T. Moore. (1990). A branch and bound algorithm for the bilevel programming, *SIAM Journal on Scientific and Statistical Computing*, Vol. 11, No. 2, pp. 281--292

B. Colson, P. Marcotte, G. Savard. (2005). A trust-region method for nonlinear bilevel programming: algorithm and computational experience, *Computational Optimization and Applications*, Vol. 30, No. 3, pp. 211—227

S. Dempe & J. F. Bard. (2001). Bundle trust-region algorithm for bilinear bilevel programming, *Journal of Optimization Theory and Applications*, Vol. 110, No. 2, pp. 265--288

N. P. Faisca, V. Dua, B. Rustem, P. M. Saraiva, E. N. Pistikopoulos. (2007) Parametric global optimization for bilevel programming, *Journal of Global Optimization*, Vol. 38, No. 4, pp. 609—623

M. Fampa, L. A. Barroso, D. Candal, L. Simonetti. (2008). Bilevel optimization applied to strategic pricing in competitive electricity markets, *Computational Optimization and Applications*, Vol. 39, No. 2, pp. 121--142

Z. H. GUMUS & C. A. Floudas. (2001). Global optimization of nonlinear bilevel programming problems, *Journal of Global Optimization*, Vol. 20, No. 1, pp. 1--31

P. Hansen, B. Jaumard, G. Savard. (1992). New branch-and-bound rules for linear bilevel programming, *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 5, pp. 1194--1217

S. Koziel, Z. Michalewicz. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation*, Vo. 7, No. 1, pp. 19--44

T. Matsui, K. Kato, M. Sakawa, T. Uno, K. Morihara. (2007). Particle swarm optimization based heuristics for nonlinear programming problems, *Proceedings of International MultiConference of Engineers and Computer Scientists 2007*, pp. 2312—2317

M. G. Nicholls. (1996). The applications of non-linear bi-level programming to be aluminium industry, *Journal of Global Optimization*, Vol. 8, No. 3, pp. 245--261

I. Nishizaki & M. Sakawa. (1999). Stackelberg solutions to multiobjective two-level linear programming problems, *Journal of Optimization Theory and Applications*, Vol. 103, No. 1, pp. 161—182

I. Nishizaki & M. Sakawa. (2000). Computational methods through genetic algorithms for obtaining Stackelberg solutions to two-level mixed zero-one programming problems, *Cybernetics and Systems: An International Journal*, Vol. 31, No. 2, pp. 203—221

I. Nishizaki, M. Sakawa, H. Katagiri. (2003). Stackelberg solutions to multiobjective two-level linear programming problems with random variable coefficients, *Central European Journal of Operations Research*, Vol. 11, No. 3, pp. 281--296

K. Niwa, Ichiro Nishizaki, M. Sakawa. (1999). Computational methods for obtaining Stackelberg Solutions to two-level non-linear programming problems, *In: Proceedings of Second Asia-Pacific Conference on Industrial Engineering and Management Systems*, pp. 489—492

K. Niwa, K. Kato, I. Nishizaki, M. Sakawa. (2006). Computational methods through particle swarm optimization for obtaining Stackelberg solutions to two-level nonlinear programming problems, *In: Proceedings of 22nd Fuzzy System Symposium*, pp. 229--230 (in Japanese)

E. Roghanian, S. J. Sadjadi, M. B. Aryanezhad. (2007). A probabilistic bi-level linear multiobjective programming problem to supply chain planning, *Applied Mathematics and Computation*, Vol. 188, No. 1, pp. 786--800

M. Sakawa & I. Nishizaki. (2001). Interactive fuzzy programming for multi-level nonconvex nonlinear programming problems, *In: Dynamical Aspects in Fuzzy Decision Making (Ed. Y. Yoshida)*, Physica-Verlag, Heidelberg

M. Sakawa & I. Nishizaki. (2009). *Cooperative and Noncooperative Multi-Level Programming*, Springer, New York

K. Shimizu, Y. Ishizuka, J. F. Bard. (1997). *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer Academic Publishers, Boston

M. Simaan & J. B. Cruz Jr. (1973). On the Stackelberg strategy in nonzero-sum games, Journal of Optimization Theory and Applications, Vol. 11, No. 5, pp. 533--555

K. Socha & M. Kisiel-Dorohinicki. (2002). Agent-based evolutionary multiobjective optimization, *Proceedings of Congress on Evolutionary Computation*, pp. 109--114

D. J. White & G. Anandalingam. (1993). A penalty function approach for solving bi-level linear programs, *Journal of Global Optimization*, Vol. 3, No. 4, pp. 397--419