Computers and Mathematics with Applications 58 (2009) 678–685

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Application of particle swarm optimization algorithm for solving bi-level linear programming problem

R.J. Kuo [a,*], C.C. Huang [b]

[a] Department of Industrial Management, National Taiwan University of Science and Technology, No. 43, Section 4, Kee-Lung Road, Taipei, 106, Taiwan, ROC

[b] Department of Industrial Engineering and Management, National Taipei University of Technology, No. 1, Section 3, Chung-Hsiao East Road, Taipei, 106, Taiwan, ROC

ARTICLE INFO

Article history:
Received 23 December 2008
Accepted 2 February 2009

Keywords:
Bi-level linear programming
Particle swarm optimization algorithm
Genetic algorithm

ABSTRACT

Bi-level linear programming is a technique for modeling decentralized decision. It consists of the upper-level and lower-level objectives. This paper attempts to develop an efficient method based on particle swarm optimization (PSO) algorithm with swarm intelligence. The performance of the proposed method is ascertained by comparing the results with genetic algorithm (GA) using four problems in the literature and an example of supply chain model. The results illustrate that the PSO algorithm outperforms GA in accuracy.

1. Introduction

Multi-level programming techniques are developed to solve decentralized planning problems with multiple decision makers in a hierarchical organization. The bi-level linear programming problem (BLPP) is a special case of multi-level linear programming problems with a two-level structure [1,2]. Most of the mathematical programming models deal with a single decision maker and a single objective function and are used for centralized planning systems. The BLPP on the other hand is developed for decentralized planning systems in which the upper level is termed as the leader and the lower level pertains to the objective of the follower. In the BLPPs, each decision maker tries to optimize its own objective function without considering the objective of the other party, but the decision of each party affects the objective value of the other party as well as the decision space.

There already have been some methods for solving BLPPs, like methods based on vertex enumeration and meta-heuristics. In this study, an attempt is made to employ particle swarm optimization (PSO) algorithm for solving BLPPs due to its promising performance in optimization problems. Four problems taken from the literature are adopted to test the proposed algorithm's performance. The experimental results indicate that PSO algorithm outperforms genetic algorithm (GA) in accuracy and has better stability. In addition, an example of supply chain model also reveals that PSO algorithm is a suitable approach for solving BLPPs.

The rest of this paper is organized as follows. Section 2 provides basic concept of BLPPs and PSO algorithm. The proposed PSO algorithm for solving BLPPs will be presented in Section 3, while Section 4 makes a thorough discussion on computational experiences. Finally, the concluding remarks are made in Section 5.

2. Background

This section will briefly present the background for bi-level programming and PSO algorithm.

* Corresponding author.
E-mail address: rjkuo@mail.ntust.edu.tw (R.J. Kuo).

0898-1221/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.02.028

2.1. Bi-level linear programming problem (BLPP)

Stackelberg game is a leader–follower strategy and an N-people nonzero-sum game. In two-person nonzero-sum games [3], the objectives of the players are neither exactly opposite nor do they coincide with each other, and the loss of one of them is not equal to the other. According to the hierarchy of Stackelberg game with leader and follower, it can develop to become a multi-level programming problem. Based on the decomposition principle for linear programs [4], many organizations have rested heavily on the principle to optimize hierarchical systems. The assumption is that all variables have been controlled by centers, and the objectives of centers can decompose into the objectives of all subunits. However, some academics like Bialas and Karwan [5,6] argued that these problems are characterized by a hierarchy of planners, each independently controlling a set of decision variables, disjoint from the others.

BLPP is a special case of multi-level linear programming problems. Assume that the higher-level decision maker has control over X and lower-level decision maker has control over Y. Then, we have $x \in X \subset R^n$, $y \in Y \subset R^m$ and $F : X \times Y \to R^1$. The BLPP can be stated as follows:

$$\begin{aligned} \text{P1}: &\quad \min_{x \in X} \quad F(x, y) = c_1 x + d_1 y, \\ \text{P2}: &\quad \min_{y \in Y} \quad f(x, y) = c_2 x + d_2 y, \\ &\quad \text{subject to } A_2 x + B_2 y \leq b, \end{aligned} \tag{1}$$

where $c_1, c_2 \in R^n$, $d_1, d_2 \in R^m$, $b \in R^p$, $A \in R^{p \times n}$, $B \in R^{p \times m}$. $P1$ is the higher-level decision maker and $P2$ is the lower-level decision maker. According to the differences of the requirements in these models, there may be some extra limitations of x and y, like the limitation of integers or the limitation of upper and lower bound. In the process of solving, once the leader has chosen an x, the x of the follower's objective function has become a constant. Thus, the objective function of follower is simplified to $\min_{y \in Y} f(y) = d_2 y$.

Based on these we have the following definitions [7]:

Definition 1. 1. Constraint set of the problem:

$$S = \{(x, y) : x \in X, y \in Y, Ax + By \leq b\}. \tag{2}$$

2. Feasible set for the follower for each x:

$$S(x) = \{y \in Y : By \leq b - A\}. \tag{3}$$

3. Projection of M onto the leader's decision space:

$$S(x) = \{x \in X : \exists y \in Y, Ax + By \leq b\}. \tag{4}$$

4. Follower rational reaction set for $x \in$ S (X):

$$P(x) = \{y \in Y : y \in \arg\min[f(\hat{x}, \hat{y}) : \hat{y} \in S(x)]\}, \tag{5}$$

$$\text{and } \arg\min\{[f(\hat{x}, \hat{y}) : \hat{y} \in S(x)]\} = \{f(x, y) \leq f(x, \hat{y}), \hat{y} \in S(x)\}. \tag{6}$$

5. Inducible region:

$$IR = \{(x, y) : (x, y) \in S, y \in P(x)\}. \tag{7}$$

Definition 2. If $\{(x, y) \in P$ (x) $|x \in S(X)\}$, (x, y) is the feasible solution of the BLPP.

BLPP is equivalent to a feasible region consisting of piecewise-linear constraints to minimize the objective function F of the leader.

Definition 3. For $\forall (x, y) \in IR$, if $\exists_{(x^*, y^*)} \in IR$, $F(x^*, y^*) \leq F(x, y)$, then (x^*, y^*) is an optimal solution of problem.

To solve the BLPPs, it must be careful that when x on the upper level has been fixed, the corresponding solution on the lower level has not been the only one. There have been some effects on the objective function of the lower level, but it has had a large differentiation on the upper level. To overcome the problem, Bialas and Karwan [8] suggested replacing the f by $f + \varepsilon F$, where $\varepsilon > 0$ is a small factor. The idea is to have the upper decision maker share a small part of his earning with the decision maker, to make this latter choose the appropriate solution to the upper objective. In the organization, this can be regarded as sharing some profits of the upper manager to all departments to assure that all departments can work hard for the whole objectives in the company. In general, to check the effect of multiple optima, we can solve the BLPPs: once with the lower objective function equal to $f + \varepsilon F$, and a second time with that function equal to $f - \varepsilon F$. And optimal solution is obtained if it is optimal in both cases. Thus, it is the optimal solution.

The solution of BLPP may not be Pareto-optimal. In other words, it may be a feasible solution to get a better objective function of a lower or upper level but does not influence the objective function value of any other levels. Wen and Hsu [9]

have treated how to find the efficient compromise solution at the situation of non-degeneracy. It means that when solving the BLPP and the solution is non-Pareto optimal, we can look for the efficient compromise solution according to the advice of the author. The upper -level decision maker can retain the original rewards, and the lower one can get the extra profits. Shih [10] advanced that it cannot assure BLPP could get the Pareto-optimal or the efficient solution only when the objections of both levels are the same. In other words, it may get the Pareto-optimal when they cooperate absolutely with each other.

According to [1], the existing methods for solving BLPPs can be divided into the following four categories: (1) methods based on vertex enumeration, (2) methods based on Kuhn–Tucker conditions, (3) fuzzy approach [11], and (4) method based on meta-heuristics. We will give more detailed discussion for the last one, since this study will focus on applying meta-heuristics for solving BLPPs.

In the category of meta-heuristics, Mathieu et al. [12] firstly proposed a GA-based bi-level programming algorithm (GABBA) for solving BLPPs. Later, Hejazi et al. [1] presented a method based on GA. It is ascertained by comparing the results with method proposed in [13]. Oduguwa and Roy [14] developed a bi-level GA, which is an elitist optimization algorithm developed to encourage limited asymmetric cooperation between the two players, to solve different classes of the BLPPs within a single framework. Yin [15] also proposed a GA-based approach for solving two BLPPs, road pricing and reserve capacity of signal-controlled road network. Wang et al. [16] proposed an evolutionary algorithm for solving nonlinear bi-level programming problem. Recently, Wang et al. [17] developed a GA which adopted some techniques to guarantee not only the initial chromosomes but also the chromosomes generated by GA are all feasible. Sahin and Cirit [18] presented an algorithm based on simulated annealing.

Besides GA, a hybrid tabu-ascent algorithm (HTA) was developed by Gendreau et al. [13]. Wen and Huang [19] employed a simple tabu search for solving a mixed-integer BLPP. The 0-1 decision variables are controlled by upper -level decision maker. They [20] also presented a tabu search approach to solve a BLPP with all real variables. Recently, Gupta et al. [21] also applied tabu search for solving the BLPP in the area of chemical engineering. In addition, Shih et al. [22] developed a neural network model for BLPP. The revised version is presented by Lan et al. [23].

2.2. Particle swarm optimization (PSO)

The PSO algorithm shares many similarities with evolutionary computation techniques such as GAs. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, the PSO algorithm has no evolutionary operators, such as crossover and mutation. In the PSO algorithm, the potential solutions, called particles, move through the problem space by following the current optimal particles.

Originally, the framework of PSO algorithm was designed by Eberhart and Kennedy [24] in 1995. Particle i is represented as $X_i = (x_{i1}, x_{i2}, \ldots, X_{iD})$, which represents a potential solution to a problem in D-dimensional space. Each particle keeps a memory of its previous best position, P_{best}, and a velocity along each dimension, represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. For each iteration, the position of the particle with the best fitness value in the search space, designated as g, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle [24,25]. The method could be divided into GBEST and LBEST versions, whose main difference is their definition of the best. In the GBEST version, the particle swarm optimizer keeps track of the overall best value, and its location, obtaining thus far by any particle in the population, which is called *gbest* (P_{gd}). For the LBEST version, in addition to *gbest*, each particle keeps track of the best solution, called *lbest* (P_{gd}), and it is attained within a local topological neighborhood of particles. However, the particle velocities in each dimension are held to a maximal velocity, $V_{\max}$, and the velocity in that dimension is limited to $V_{\max}$. Later, inertial weight was developed to better balance exploration and exploitation in order to eliminate the need for $V_{\max}$ [26,27]. The updating rule is given by

$$V_{id}^{new} = W \cdot V_{id}^{old} + c_1 \cdot rand_1 \cdot (P_{id} - X_{id}) + c_2 \cdot rand_2 \cdot (P_{gd} - X_{id}) \tag{8}$$

$$X_{id}^{new} = X_{id}^{old} + V_{id}^{new} \tag{9}$$

where W is the inertia weight, c_1 and c_2 determine the relative influence of the social and cognition components (learning factors), while $rand_1$ and $rand_2$ denote two random numbers uniformly distributed in the interval [0, 1]. Recent work done by Clerc indicates that the use of a *constriction factor* may be necessary to ensure convergence of the PSO algorithm [28]. PSO algorithm has been applied for many optimization problems, such as scheduling and traveling sales problems.

3. Methodology

The PSO concept originated as a simulation of a simplified social system. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. Due to this merit, this study employs PSO algorithm for solving the BLPPs.

The BLPP is developed for decentralized planning systems in which the upper level is termed as the leader and the lower level pertains to the objective of the follower. In the BLPP, each decision maker tries to optimize its own objective function without considering the objective of the other party, but the decision of each party affects the objective value of the other

Table 1
BLPPs and their corresponding characteristics.

Source	Wen and Hsu [29]	Bialas and Karwan [6]	Liu and Hart [30]	Bard and Falk [31]
Number of variables	2	2	2	4
Range of variables	$\geqq 0$	$\geqq 0$	$\geqq 0$	$\geqq 0$

party as well as the decision space. This section will present the way to apply PSO algorithm for BLPPs. The PSO algorithm for solving BLPPs is presented as follows:

Step 1: Set up parameters including population size (the number of particles), Maximal velocity ($V_{\max}$), inertial weight (w), and two learning factors, (c_1 and c_2). Two random variables, $rand_1$ and $rand_2$, are in the interval [0, 1].

Step 2: Set up the searching range for x, which will influence the searching speed. Initialize each particle randomly with initial position, X_{id}, within the pre-specified range and velocity, V_{id}, in the range of maximal speed, $V_{\max}$. This study adopts the float coding method [17] to generate the random numbers for the upper-level variables. Thus, every particle represents the real dimensional position. Then, program for variable ys in the lower level. Each particle's position is represented as:

$$X_{id} = (x_{i1}, \ldots, x_{in}, y_{i1}, \ldots y_{im}). \tag{10}$$

Step 3: Calculate every particle's fitness value using

$$F = c^t X_{id}. \tag{11}$$

Step 4: Update the local best position, P_{id}, and global best position, P_{gd}.

Step 5: According to the updated P_{id} and P_{gd}, use Eqs. (12) and (13) to update the velocity and position for every particle.

$$V_{id}^{new} = W \cdot V_{id}^{old} + c_1 \cdot rand_1 \cdot (P_{id}^{old} - X_{id}^{old}) + c_2 \cdot rand_2 \cdot (P_{gd}^{old} - X_{id}^{old}) \tag{12}$$

$$X_{id}^{new1} = X_{id}^{old} + V_{id}^{new}. \tag{13}$$

Every particle's velocity is constrained by the pre-determined maximal velocity, $V_{\max}$, and every particle's position, X_{id}, should be within the specified range:

$$l \leq x_i \leq u. \tag{14}$$

Step 6: Stop if the specified number of generations is satisfied; otherwise, go back to Step 3.

4. Computational experiences

In order to test the proposed PSO algorithm, four examples as presented in Appendix were taken from the literature. These examples were selected in a manner to demonstrate the capability of the algorithm which is able to handle a variety of problem conditions and complexities. In addition, an example of supply chain model is also formulated in order to verify PSO algorithm's feasibility.

The description of the four BLPPs is presented in Table 1. For the purpose of choosing problems with different levels of complexities, they have been selected from different sources. For each problem, 30 runs were simulated. This was done to ensure that the selection of seed values for the random number generator used did not have a major influence on the solution obtained. All simulation was implemented on a personal computer with Intel Pentium Duo CPU 2.8 GHz and 1 GB RAM using MATLAB.

For the comparison purpose, this study treats solution obtained from Lingo as the best solution. Since each problem is run for 30 times, error rate and standard deviation are calculated, respectively. The standard deviation is based on the upper-level objective function. They are illustrated in Eqs. (15) and (16), respectively as follows:

$$\text{Error rate} = \frac{\left|f^* - f_M^*\right|}{f^*} \times 100\% \tag{15}$$

where f^* : optimal solution based on Lingo, and
f_M^* : optimal solution obtained from M method.

$$\text{Standard deviation} = \sqrt{\frac{1}{N}\sum_{i}^{N}\left(f_{Mi}^* - f_{MA}^*\right)^2} \tag{16}$$

where $i = 1, \ldots, N$ (This study uses $N = 30$)
$f_{Mi}^* =$ The optimal solution for M method in ith run.
$f_{MA}^* =$ The average of N optimal solutions for M method.

Table 2 lists the corresponding parameter setup for these two algorithms, PSO algorithm and GA.

Table 2
Parameter setup.

Method	GA	PSO
Parameter	Population: 20 Crossover rate: 0.8 Mutation rate: 0.1 Generations: 200	Population: 20 V_{max}: 10 Inertial weight: 1.2–0.2 Iterations: 200

Table 3
The best solutions of problem 1.

	FNN	GA	PSO	Lingo
X_1	17.5000	17.4528	17.4535	17.4545
X_2	10.9000	10.9055	10.9070	10.90909
f_1	85.0909	85.0551	85.0700	85.0909
$f_1$error rate	0	0.04%	0.02%	N/A
f_2	−50.2000	−50.16937	−50.17450	−50.18182
$f_2$error rate	0.036%	0.038%	0.015%	N/A

Table 4
Average and standard deviation of problem 1.

	GA	PSO
x_1	17.43289	17.4417
x_2	10.86578	10.8788
f_1	84.657812	84.85119
f_2	−50.03023	−50.0781
$f_1$error rate	0.51%	0.28%
$f_2$error rate	0.30%	0.21%
$f_1$standard deviation	0.38650	0.189965

Table 5
The best solutions of Problem 2.

	GA	PSO	Lingo
x_1	15.9984	15.9999	16
x_2	10.9968	10.9998	11
f_1	10.9968	10.9998	11
$f_1$error rate	0.03%	0.002%	N/A
f_2	−10.9968	−10.9998	−11
f_2 error rate	0.03%	0.002%	N/A

Table 6
Average and standard deviation of problem 2.

	GA	PSO
x_1	15.9041	15.99806
x_2	10.8082	10.9961
f_1	10.8082	10.9961
f_2	−10.8082	−10.9961
f_1 error rate	1.74%	0.04%
f_2 error rate	1.74%	0.04%
f_1 standard deviation	0.168034	0.004014

4.1. Problem results

(1) Problem 1

Problem 1 is the example taken from Wen and Hsu [29]. Since in [22], the problem is solved by fuzzy neural network (FNN), the results are also presented in Table 3. However, only the best solution instead of average is provided. The penality parameter is $K = 1.0$ and the training rate is $\mu = 0.005$. In Table 3, we found that PSO algorithm has better results than GA both for upper and lower objectives, f_1 and f_2. Regarding FNN, PSO algorithm has smaller f_2 error rate. But, FNN has smaller f_1 error rate. Table 4 illustrates that average and standard deviation values for PSO algorithm are smaller than those of GA.

(2) Problem 2

Problem 2 is the example taken from [6]. In problem 2, the objectives of upper and lower levels are just conflict. Table 5 illustrates that PSO algorithm has better solution than GA. In addition, PSO algorithm's average value and standard deviation are also much smaller than those of GA as shown in Table 6.

Table 7
The best solutions of Problem 3 for different methods.

	GA	PSO	Lingo
x_1	3.9994	4	4
x_2	3.9974	4	4
f_1	15.9917	16	16
f_1 error rate	0.05%	0	N/A
f_2	−3.94636	−4	−4
f_2 error rate	1.34%	0	N/A

Table 8
The average and standard deviation of Problem 3 for different methods.

	GA	PSO
x_1	3.986583	3.99908
x_2	3.946363	3.996343
f_1	15.82567	15.98811
f_2	−3.946363	−3.996343
f_1 error rate	17.43%	1.19%
f_2 error rate	1.34%	0.09%
f_1 standard deviation	0.192652	0.009664

Table 9
Results of Problem 4.

Range [0, 1]	GA	PSO	Lingo
Parameter setup	Population: 20 Crossover rate: 0.9 Mutation rate: 0.1 Generations: N/A	Population: 20 V_{max}: 10 Inertia weight: 1.2–0.2 Iterations: 150	N/A
x_1	0.000	0.0004	0
x_2	0.898	0.8996	0.9
y_1	0.000	0	0
y_2	0.599	0.5995	0.6
y_3	0.399	0.3993	0.4
f_1	29.1480	29.1788	29.2
f_1 error rate	0.18%	0.07%	N/A
f_2	−3.1930	−3.1977	−3.2
f_2 error rate	0.22%	0.07%	

Table 10
The average of Problem 4 for different methods.

	GA	PSO	Lingo
x_1	0.15705	0.02192	0
x_2	0.86495	0.86693	0.9
y_1	0	0	0
y_2	0.47192	0.56335	0.6
y_3	0.51592	0.34108	0.4
f_1	21.52948	24.81256	26
f_1 error rate	17.19%	4.57%	N/A
f_2	−3.39072	−3.1977	−3.2
f_2 error rate	5.96%	6.21%	N/A
f_1 error rate	3.14432	1.55374	N/A

(3) Problem 3

Problem 3 is the example taken from [30] for discussing BLPP. Due to low problem complexity, it is much easier to demonstrate the fast convergence characteristics of PSO algorithm. In addition, the accuracy is significantly better than that of GA. In 30 runs, the standard deviation of PSO, 0.009664, is much smaller than that of GA which is 0.192652. This shows that PSO algorithm is much more stable than GA. The detailed results are depicted in Tables 7 and 8.

(4) Problem 4

Problem 4 with higher dimensionality taken from [31] is employed to test PSO algorithm's solving capability. Wang et al. [17] also apply GA to solve this problem. The parameter setup and corresponding solution are tabulated in Table 9.

Since Wang et al. [17] only present the one-run solution, this study codes the GA according to Wang et al.'s method. The experimental results are listed in Table 10. PSO algorithm also have better performance than GA.

Table 11
Results for supply chain model.

Methods	PSO	GA	Lingo
D_{11}	29.97433	29.88833	30
D_{21}	0	0	0
X_1	30	30	30
X_2	20	20	20
f_1	797.4329	790.1857	800
f_2	4046.535	4034.925	4050
$f1$ error rate	−0.32%	−1.227%	
$f2$ error rate	−0.09%	−0.372%	
$f1$ standard deviation	3.144077	7.13298	

4.2. Supply chain model results

From the above simulation results, we can conclude that PSO algorithm is a promising method for solving BLPPs. In this subsection, a supply chain model is formulated and solved by using the proposed method. The current supply chain model is with two distribution centers and one assembly factory. The variable notation for supply chain model is as follows:

i: The number of distribution centers,
j: The number of assembly factories,
D_{ij}: The demand of the jth assembly factory from the ith distribution center,
P_{ij}: The product price that the ith distribution center provides to the jth assembly factory,
X_i: The total amount of products that the ith distribution center has,
$C_i(X_i)$: The unit cost for the ith distribution center to purchase product X_i,
W_i: The capacity constraint for the ith distribution center,
A_j: The total amount of products that the jth assembly factory needs,
$S_{ij}(D_{ij})$: The unit cost of product that the jth assembly factory order from the ith distribution center,
$T_{ij}(D_{ij})$: The unit transportation cost of product being delivered from the ith distribution center to the jth assembly factory, and
h_j: The unit holding cost for the jth assembly factory.

This study assumes that distribution centers belong to the upper level, while assembly factories are the lower level. The objective of the upper level is to maximize the total profits for distribution centers, while the objective of the lower level is to minimize the total costs for assembly factories.

The related parameters are set as: $i = 2, j = 1, P_{11} = 100, P_{21} = 150, C_1(X_1) = 40, C_2(X_2) = 50, W_1 = 30, W_2 = 20, S_{11}(D_{11}) = 10, S_{21}(D_{21}) = 15, T_{11}(D_{11}) = 20, T_{21}(D_{21}) = 25$, and $h_1 = 5$. Thus, the corresponding model can be represented as:

$$
\begin{aligned}
&\text{MAX} \quad f_1 = 100D_{11} + 150D_{21} - 40X_1 - 50X_2\\
&\text{Min} \quad f_2 = 135D_{11} + 195D_{21}\\
&\text{s.t.} \quad D_{11} \le X_1\\
&\qquad D_{21} \le X_2\\
&\qquad X_1 \le 30\\
&\qquad X_2 \le 20\\
&\qquad X_1 + X_2 \ge 50\\
&\qquad X_i \ge 0, \qquad D_{ij} \ge 0, \quad i = 1, 2 \text{ and } j = 1.
\end{aligned}
$$

The computational results are tabulated in Table 11. It revels that PSO algorithm has more accurate solution than GA. Especially PSO algorithm's standard deviation is smaller than that of GA. It can provide more feasible solution most of the time.

5. Conclusions

This study has proposed a PSO-based method for BLPPs. The experimental results of four problems taken from the literature illustrate that the proposed PSO algorithm outperforms GA for most of the problems. Besides, PSO algorithm has smaller standard deviation. This implies that PSO algorithm has better stability compared to GA. In addition, the computational time of PSO algorithm is also smaller than that of GA. However, the current algorithm is only suitable for BLPPs. It is promising to extent the proposed method for multi-level linear programming problems in the future. Besides, it is feasible to gather virtues of PSO algorithm and GA to ascend learning efficiency.

Acknowledgement

This study is finally supported by the National Science Council of Taiwan Government under contract number: NSC 96-2221-E-011 -178 -MY3. This support is appreciated.

Appendix

Problem 1	Problem 2
Max $f_1 = -2x_1 + 11x_2$ where x_2 solves Max $f_2 = -x_1 - 3x_2$ s.t. $x_1 - 2x_2 \le 4$ $2x_1 - x_2 \le 24$ $3x_1 + 4x_2 \le 96$ $x_1 + 7x_2 \le 126$ $-4x_1 + 5x_2 \le 65$ $x_1 + 4x_2 \ge 8$ $x_1, x_2 \ge 0$	Max x_2 where x_2 solves Max $-x_2$ s.t. $-x_1 - 2x_2 \le 10$ $x_1 - 2x_2 \le 6$ $2x_1 - x_2 \le 21$ $x_1 + 2x_2 \le 38$ $-x_1 + 2x_2 \le 18$ $x_1, x_2 \ge 0$
Problem 3	**Problem 4**
Max $x_1 + 3x_2$ where y solves Max $-x_2$ s.t. $-x_1 + x_2 \le 3$ $x_1 + 2x_2 \le 12$ $4x_1 - x_2 \le 12$ $x_1, x_2 \ge 0$	Max $8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$ where y solves Max $-x_1 - 2x_2 - y_1 - y_2 - 2y_3$ s.t. $y_1 - y_2 - y_3 \ge -1$ $-2x_1 + y_1 - 2y_2 + 0.5y_3 \ge -1$ $-2x_2 - 2y_1 + y_2 + 0.5y_3 \ge -1$ $x_1, x_2, y_1, y_2, y_3 \ge 0$

References

[1] S.R. Hejazi, A.I. Memarian, G. Jahanshahloo, M.M. Sepehri, Linear bi-level programming solution by genetic algorithm, Computers and Operations Research 29 (2002) 1913–1925.
[2] B. Colson, P. Marcotte, G. Savard, Bi-level programming: A survey, A Quarterly Journal of Operation Research 3 (2005) 87–107.
[3] R. Luce, H. Raiffa, Game and Decisions, Wiley, New York, 1957.
[4] G.B. Dantzing, P. Wolfe, Decomposition principle for linear programs, Operations Research 8 (1) (1960) 101–111.
[5] W.F. Bialas, M.H. Karwan, On two-level optimization, IEEE Transaction on Automatic Control 27 (1982) 211–214.
[6] W.F. Bialas, M.H. Karwan, Two-level linear programming, Management Science 30 (8) (1984) 1004–1020.
[7] J.F. Bard, Practical Bilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers, Boston, 1998.
[8] W.F. Bialas, M.H. Karwan, Multilevel linear programming, Research Report No. 78-1, Operations Research Program, Department of Industrial Engineering, State University of New York at Buffalo, 1978.
[9] U.P. Wen, S.T. Hsu, Linear Bi-level programming problems—A review, Journal of Operations Research Society 42 (2) (1991) 125–133.
[10] H.S. Shih, Fuzzy multi-level optimization, Ph.D. Dissertation, Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan, Kansas, USA, 1995.
[11] M. Sakawa, I. Nishizaki, Interactive fuzzy programming for two-level nonconvex programming problems with fuzzy parameters through genetic algorithms, Fuzzy sets and Systems 127 (2002) 185–197.
[12] R. Mathieu, L. Pittard, G. Anandalingam, Genetic algorithm based approach to bi-level linear programming, Operations Research 28 (1) (1994) 1–21.
[13] M. Gendreau, P. Marcotte, G. Savard, A hybrid Tabu-ascent algorithm for the linear bilevel programming problem, Journal of Global Optimization 8 (3) (1996) 217–233.
[14] V. Oduguwa, R. Roy, Bi-level optimization using genetic algorithm, IEEE International Conference on Artificial Intelligence Systems, 2002, pp. 322–327.
[15] Y. Yin, Genetic-algorithms-based approach for bilevel programming models, Journal of Transportation Engineering (2000) 115–120.
[16] Y. Wang, Y. Jiao, H. Li, An evolutionary algorithm for solving nonlinear bi-level programming problem based on a new constraint-handling scheme, IEEE Transactions on Systems, Man and Cybernetics, Part (c) 35 (2) (2005) 221–232.
[17] G.M. Wang, X.J. Wang, Z.P. Wan, Y.L. Chen, Genetic algorithms for solving linear bilevel programming, Proceedings of the IEEE Sixth International Conference on Parallel and Distributed Computing, 2005.
[18] H.K. Sahin, R.A. Ciric, A dual temperature simulated annealing approach for solving bilevel programming problem, Computers and Chemical Engineering 23 (1998) 11–25.
[19] U.P. Wen, A.D. Huang, A simple Tabu Search method to solve the mixed-integer problem bi-level programming problem, European Journal of Operational Research 88 (1996) 563–571.
[20] U.P. Wen, A.D. Huang, A Tabu search approach for solving the linear bilevel programming problem, Journal of the Chinese Institute of Industrial Engineers 13 (1996) 113–119.
[21] R.J.K. Gupta, H.S. Kusumakar, V.K. Jayaraman, B.D. Kulkarni, A Tabu search based approach for solving a class of bilevel programming problems in chemical engineering, Journal of Heuristics 9 (2003) 307–319.
[22] H.S. Shih, U.P. Wen, E.S. Lee, K.M. Lan, H.C. Hsiao, A neural network approach to multi-objective and multilevel programming problems, Computers and Mathematics with Applications 48 (2004) 95–108.
[23] K.M. Lan, U.P. Wen, H.S. Shih, E.S. Lee, A hybrid neural network approach to bilevel programming problems, Applied Mathematics Letters 20 (2007) 880–884.
[24] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya Japan, 1995, pp. 39–43.
[25] F. Wang, Y. Qiu, A modified particle swarm optimizer with roulette selection operator, Proceedings of Natural Language Processing and Knowledge Engineering, 2005, pp. 765–768.
[26] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, 1998, pp. 69–73.
[27] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, evolutionary programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, 1998, pp. 591–600.
[28] M. Clerc, The swarm and the queen: Toward a deterministic and adaptive particle swarm optimization, Proceedings of ICEC'99, 1999, pp. 1951–1957.
[29] U.P. Wen, S. Hsu, Algorithms for solving integer two-level linear programming problem, Computers and Operational Research 17 (2) (1991) 133–142.
[30] Y.H. Liu, S.M. Hart, Characterizing an optimal solution to the linear bilevel programming problem, European Journal of Operational Research 79 (1994) 164–166.
[31] J.F. Bard, J.E. Falk, An explicit solution to the multi-level programming problem, Computers and Operations Research 9 (1) (1982) 77–100.