Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 378568, 10 pages
http://dx.doi.org/10.1155/2013/378568

Research Article

A Hybrid Estimation of Distribution Algorithm and Nelder-Mead Simplex Method for Solving a Class of Nonlinear Bilevel Programming Problems

Aihong Ren,[1] Yuping Wang,[1] and Fei Jia[2]

[1] *School of Computer Science and Technology, Xidian University, Xi'an 710071, China*
[2] *School of Science, Xidian University, Xi'an 710071, China*

Correspondence should be addressed to Aihong Ren; raih2003@yahoo.com.cn

Received 26 March 2013; Accepted 14 July 2013

Academic Editor: Yansheng Liu

We propose a hybrid algorithm based on estimation of distribution algorithm (EDA) and Nelder-Mead simplex method (NM) to solve a class of nonlinear bilevel programming problems where the follower's problem is linear with respect to the lower level variable. The bilevel programming is an NP-hard optimization problem, for which EDA-NM is applied as a new tool aiming at obtaining global optimal solutions of such a problem. In fact, EDA-NM is very easy to be implemented since it does not require gradients information. Moreover, the hybrid algorithm intends to produce faster and more accurate convergence. In the proposed approach, for fixed upper level variable, we make use of the optimality conditions of linear programming to deal with the follower's problem and obtain its optimal solution. Further, the leader's objective function is taken as the fitness function. Based on these schemes, the hybrid algorithm is designed by combining EDA with NM. To verify the performance of EDA-NM, simulations on some test problems are made, and the results demonstrate that the proposed algorithm has a better performance than the compared algorithms. Finally, the proposed approach is used to solve a practical example about pollution charges problem.

1. Introduction

The bilevel programming problem (BLP) is a nested optimization problem with two levels (viz., the upper and lower levels) in a hierarchy. The upper level decision maker (the leader) and the lower level decision maker (the follower) control their own sets of decision variables, respectively, and have their own objective functions and constraints. The leader makes a decision first, and thereafter the follower chooses his/her strategy according to the leader's action. Anticipating the reaction of the follower, the leader selects the parameters so as to optimize his/her own objective function. The leader can influence, but cannot control, the decision of the follower. As optimization problems with hierarchical structure, the bilevel programming problem can be widely used in such areas as resource allocation, decentralized control, network design, and so forth [1]. Over the past decades, the bilevel programming problem has been increasingly addressed in the literature including some useful reviews [1, 2], surveys [3, 4], and good textbooks [5, 6].

However, it is extremely difficult to solve the bilevel programming problem due to its nonconvexity and noncontinuity, especially the nonlinear bilevel programming problem. For this context, many researchers have devoted themselves into developing efficient algorithms for solving the problem over the years. To date, there already have been some traditional approaches for solving such a problem, such as vertex enumeration method, approach based on the Kuhn-Tucker condition, descent approach, and penalty function approach. Most of these algorithms can accurately work out the optimal solution, but they are computationally costly even when solving small sized problems. As the scale of the problem increases, these exact algorithms could no longer afford the computation efficiently within reasonable time duration.

To overcome the above shortcomings, intelligent algorithms (such as evolutionary algorithm, genetic algorithm, tuba search approach, and simulated annealing) have been widely used to solve different problems in optimal areas and have been extended to deal with the bilevel programming for their good characteristics. For the linear bilevel programming problem, Mathieu et al. [7] firstly proposed a genetic algorithm for solving the problem. Subsequently, other kinds of intelligent algorithms [8–11] have been appearing for the same problem. For the nonlinear case, Wang et al. [12] presented an evolutionary algorithm for solving nonlinear bilevel programming problems with nonconvex objective functions. Deb and Sinha [13] designed a hybrid evolutionary-cum-local-search-based algorithm for dealing with bilevel multiobjective programming problems. In addition, a novel particle swarm optimization based on CHKS smoothing function was proposed in [14]. By using duality conditions, an evolutionary algorithm was developed to cope with a class of bilevel programming with a linear lower level problem [15]. Wan et al. [16] addressed a hybrid intelligent algorithm by combining the particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems.

In recent years, an increasing interest has been concentrated on a class of probabilistic and graphical model based evolutionary computational methods, commonly called as the estimation of distribution algorithm (EDA) [17, 18]. Compared with other evolutionary algorithms, the EDA do not use crossover or mutation. Instead, it selects the best solutions from the current population and explicitly extracts global statistical information from the selected solutions. The EDA has some advantages to solve complex optimization problems, especially a high convergent reliability and low time consumption, and has been used widely in many real world problems. Nevertheless, to the best of our knowledge, there is no research work about the EDA for solving the bilevel programming problem. To enhance the performance of the EDA in our work, a local search algorithm, the Nelder-Mead simplex method, is integrated with the EDA to solve the bilevel programming.

In this paper, we consider a class of nonlinear bilevel programming problems where the follower's problem is linear only with respect to the lower level variable. To deal with such problems, based on the optimality results of linear programming, a hybrid algorithm is proposed by combing the EDA with a local simplex search technique. As a mater of fact, it can effectively integrate the characteristic of global search in the EDA and the capability of local search in simplex search technique to avoid converging ahead of time and to raise the accuracy of problem solving, as well as to make the search converge faster than pure EDA. Numerical simulations on some test problems are carried out, and the results demonstrate that the proposed algorithm has a better performance than the compared algorithms.

The remainder of the paper is organized as follows. Section 2 describes the formulation of the nonlinear bilevel programming problem and introduces the related definitions. Section 3 proposes a hybrid algorithm by combining the estimation of distribution algorithm and the Nelder-Mead simplex method for solving nonlinear bilevel programming problems. We conduct the simulations on the proposed algorithm and compare the results with some other algorithms for some test problems in Section 4. Finally, a conclusion is given in Section 5.

2. A Class of Nonlinear Bilevel Programming Problems

In this paper, we restrict our attention to the following nonlinear bilevel programming problem (NBLP):

$$\begin{aligned} &\min_{x\in X} \quad F(x,y), \quad \text{where } y \text{ solves},\\ &\min_{y\in Y} \quad f(x,y) = a(x)^T y + b(x), \\ &\text{s.t.} \quad C(x)\,y \le d(x), \quad y \ge 0, \end{aligned} \tag{1}$$

where $x \in X \subset R^n$, $y \in Y \subset R^m$ are the decision variables under the control of the upper and lower level problems, respectively; the sets X and Y represent box sets of vectors x and y; $F(x, y)$, $f(x, y)$ are objective functions of the upper and lower level problems, respectively. $a : R^n \to R^m$, $b : R^n \to R$, $d : R^n \to R^p$, and $C : R^n \to R^{p\times m}$ whose rank is p.

Next we give the following definitions of the nonlinear bilevel programming problem:

(1) constraint region:

$$S = \{(x,y) \in X \times Y \mid C(x)\,y \le d(x), y \ge 0\}; \tag{2}$$

(2) for fixed x, the feasible region of the lower level problem:

$$S(x) = \{y \in Y \mid C(x)\,y \le d(x), y \ge 0\}; \tag{3}$$

(3) projection of S onto the upper level's decision space:

$$S_1 = \{x \in X \mid \text{there exists } y \text{ such that } (x,y) \in S\}; \tag{4}$$

(4) for each fixed $x \in S_1$, the rational reaction set of the lower level problem:

$$M(x) = \{y \in Y \mid y \in \operatorname{argmin}\{f(x,v), v \in S(x)\}\}; \tag{5}$$

(5) inducible region:

$$\mathrm{IR} = \{(x,y) \in S \mid y \in M(x_1)\}. \tag{6}$$

In order to ensure that the nonlinear bilevel programming problem is well posed, assume that in our work the constraint region S is nonempty and compact, and the lower level decision maker has some room to respond for all decisions taken by the upper level decision maker; that is $M(x) \neq \emptyset$.

It is worth mentioning that the solution set $M(x)$ of the lower level problem is not always a singleton for a given $x \in S_1$. In this case, the upper level problem is not a well-defined optimization problem. To overcome such an

unpleasant situation, there are some strategies available for the leader, such as the optimistic approach and the pessimistic approach [6]. In this paper we will avoid this unpleasant situation by assuming that there is a unique solution to the lower level problem for each fixed $x \in S_1$.

Observing that the follower's problem is linear with respect to the lower level variable. For fixed x, it is obvious that the follower's problem is a linear programming problem. Now the optimality conditions for the follower's problem are discussed according to the optimality conditions for the standard linear programming problem. For fixed x, the term $b(x)$ is constant in the follower's problem, and we can ignore and delete it. As a result, the follower's problem can be written as the following problem by adding slack variables:

$$\begin{aligned} \min_{z} \quad & f(x,z) = \overline{a}(x)^T z, \\ \text{s.t.} \quad & \overline{C}(x)\, z = d(x), \quad z \geq 0, \end{aligned} \tag{7}$$

where $\overline{C}(x) = (C(x), I)$, $z = (y, y_0) \in R^{m+p}$, $\overline{a}(x) = (a(x), 0)$, and y_0 is a slack vector.

For given x, the term $\overline{C}(x)$ is an $p \times (m + p)$ constant matrix, and its rank is p. Assume that $d(x) \geq 0$. Given a set of indices $B \subset \{1, \ldots, (m+p)\}$, let B be a basis of problem (7), and let $N = \{1, \ldots, (m + p)\} - B$ be a nonbasis. A variable z_i with index i is called basic if $i \in B$, nonbasic otherwise. Then we split $\overline{C}(x)$, $\overline{a}(x)$, and z into basis and nonbasis parts by means of B and N as index sets. Denoted by $\overline{C}(x) = (\overline{C}(x)_B, \overline{C}(x)_N)$, $\overline{a}(x)^T = (\overline{a}(x)_B^T, \overline{a}(x)_N^T)$, and $z = (z_B, z_N)$. According to the theory of the simplex method, an optimal basis of problem (7) is characterized by the following inequalities:

$$\begin{aligned} \overline{C}(x)_B^{-1} d(x) \geq 0, \\ \overline{a}(x)^T - \overline{a}_B(x)^T \overline{C}(x)_B^{-1} \overline{C}(x) \geq 0. \end{aligned} \tag{8}$$

As a matter of fact, the inequalities (8) can be considered as the optimality conditions for the follower's problem. If the inequalities hold, then an optimal solution $z(x) = (y(x), y_0(x))$ is provided to the follower's problem; that is to say, the basic variable values of the solution are taken as $\overline{C}(x)_B^{-1} d(x)$, while the values of nonbasic variables are 0. In essence, the presence of such a basis guarantees that there exists an optimal solution to the follower's problem.

3. Hybrid EDA-NM for Solving Nonlinear Bilevel Programming Problems

3.1. Overview of Estimation of Distribution Algorithm. The estimation of distribution algorithm (EDA), first proposed by Mühlenbein and Paaß [27] and later developed by Pelikan [18], is a new class of evolutionary optimization techniques. Essentially, the EDA is also population-based algorithms similar to other evolutionary algorithms. The main difference between the EDA and other evolutionary algorithms is the way in which new offspring is generated in each generation. There is neither crossover nor mutation operator in the EDA, while the algorithm selects the most promising individuals from the current population, constructs a probability model based on statistical information from the selected individuals, and then generates new offspring through sampling from the constructed probability model.

The EDA works generally as follows.

Step 1. Randomly generate initial population.

Step 2. Select population of promising individuals from the current population and build the probability model of the selected individuals.

Step 3. Sample new offspring from the probability model.

Step 4. Replace some or all of the individuals in the previous population by the new offspring.

Step 5. If the stopping condition is met, stop. Otherwise, go to Step 2.

The EDA is able to overcome some drawbacks exhibited by traditional evolutionary algorithms. One of the biggest advantages of the EDA over other evolutionary algorithms is its ability to adapt their operators to the structure of the problem. Furthermore, this important difference allows the EDA to solve some problems for which other algorithms scale poorly. Nevertheless, the EDA pays more attention to global exploration, while its exploitation capability is relatively limited. So, an effective EDA should balance the exploration and the exploitation abilities. A very natural way to improve the performance of the EDA is to hybridize local search with the EDA.

3.2. Overview of Nelder-Mead Simplex Search Method. The Nelder-Mead simplex search method (NM) [28] is a local search method developed for nonlinear and deterministic optimization without the need for gradient information. The NM algorithm can be implemented based on four basic procedures: reflection, expansion, contraction, and shrinkage depending on the values at the vertices and center of the simplex. To be more specific, an N-dimensional simplex is defined as the convex hull of $N+1$ vertices. If any vertex of an nondegenerate simplex is taken as the origin, then the rest N vertices define vector directions that span the N-dimensional vector space. The extreme point of the simplex with the worst function value is moved, and the reflected point is generated. On one hand, if the reflected point is better than the best point, the expansion of the simplex is performed in one or another direction to take larger steps. On the other hand, a contraction step will be taken when the reflected point is the worst point in the new simplex, restricting the search on a smaller region. If the earlier worst point is better than the contracted one, the shrinkage step is performed. Through these operations, the simplex can improve itself and come closer and closer to a local optimum point sequentially. In the following, the above fundamental operations are given:

reflection: $X_{\text{refl}} = (1 + \alpha) X_{\text{cent}} - \alpha X_{\text{high}}$,

expansion: $X_{\text{exp}} = \gamma X_{\text{refl}} - (1 - \gamma) X_{\text{cent}}$,

contraction: $X_{\text{cont}} = \beta X_{\text{high}} - (1-\beta)X_{\text{cent}}$,

shrink: $X_i = \delta X_i - (1-\delta)X_{\text{low}}$,

where α, β, γ, and δ are constants, X_{high} is the worst vector corresponding to the maximum function value, and X_{cent} is the center of the simplex excluding X_{high} in the minimization case.

In essence, The NM can be regarded as a direct line-search method of steepest descent kind. Therefore, it is extremely flexible and very useful for exploring difficult problems. Moreover, the NM usually offers less computational costs when compared with evolutionary computational methods. However, the convergence properties of the NM are in general poor.

3.3. The Proposed Algorithm. As is stated in previous section, the estimation of distribution algorithm is one of the global search algorithms. The algorithm is less likely to be entrapped in local optima but requires much computational effort. While the Nelder-Mead simplex method is a very efficient local search procedure, its convergence is extremely sensitive to the selected starting point. Therefore, the main idea of integrating the estimation of distribution algorithm with the Nelder-Mead simplex method in our work is to combine their advantages and avoid disadvantages. Furthermore, an effective algorithm should balance the exploration and the exploitation abilities. Consequently, to enhance the performance of the estimation of distribution algorithm, we use the Nelder-Mead simplex method as the local search mechanism to improve the efficiency of the search in exploitation.

3.3.1. The Structure of the Proposed Algorithm. As far as the performance of evolutionary algorithms is concerned, a good initialization method is very important for improving the convergence speed and the quality of the final solution. For this purpose, uniform design, proposed by Fang [29] and Wang and Fang [30], is applied to design starting experiments, so that initial points could be scattered over the feasible region as uniformly as possible and the number of uniform design points should be required as small as possible. For more details on uniform design please refer to [31].

The initial population consists of N individuals associated with the upper level variable. To generate the initial population, uniform design is first used to generate N points $\overline{x}^1, \overline{x}^2, \ldots, \overline{x}^N$ from the search space of the upper level variable. To solve the follower's problem for given $\overline{x}^i$, $i = 1, 2, \ldots, N$, the inequalities (8) are checked for feasibility. If it is feasible, the initial point can be put into the initial population. If it is not feasible, the initial point can be discarded. Then, a new sampled point from the search space of the upper level variable is randomly generated, and we repeat the procedure until the inequalities (8) are satisfied, then select this sampled point to join the initial population. It is worth mentioning that the initial population is constructed by N different individuals to guarantee the initial population with certain quality and enough diversity.

In our work, the fitness function of the individual is defined as the leader's objective function. A total of N individuals are sorted according to the fitness function. Taking into account the balance between global exploration and local exploitation, the current population is divided into two parts, and population size is set $N = M + (n+1)$, where the top M individuals are updated by the estimation of distribution algorithm and the last $n+1$ individuals are adjusted by the Nelder-Mead simplex method.

In the process of the EDA, a subset of the promising individuals from the top M individuals is selected with a selection method based on the fitness function. As a matter of fact, any selection method biased towards better fitness can be used, and in this paper, the traditional roulette-wheel selection is applied to select m ($m < M$) individuals from the current M individuals. It is worth mentioning that the EDA produces offspring by sampling according to a probability model rather than crossover and mutation operators like other evolutionary algorithms. Therefore, it is crucial to choose a good probabilistic model in the EDA. Here we employ a Gaussian model as a probabilistic model with very low computational cost. To be more specific, based on the statistical information extracted from these m selected individuals, a Gaussian model is constructed, and new individuals are generated according to the model. Then the inequalities (8) are checked for feasibility for each new individual, and the set of feasible individuals is formed. By this means, M new individuals are generated. The searching procedure is presented as follows.

Step 1. From ranked population, and select the top M individuals.

Step 2. Use roulette wheel to select a set of the promising m individuals from the current M individuals.

Step 3. Build a Gaussian model based on the statistical information extracted from these m selected individuals, and generate new individuals.

Step 4. For each new individual, check the inequalities (8).

At the local exploitation part, the algorithm works like the basic NM stated in previous section. In order to start up the NM, one has to define the initial simplex, in principle composed of $n+1$ distinct vectors. For this purpose, the last $n+1$ individuals from the current population based on rank-based fitness form the initial simplex thus avoiding extra function evaluations. Meanwhile the best point, the worst point and the second worst point are determined, and the center of the simplex excluding the worst point is computed. Then the operations of this method are to rescale the simplex based on the local behavior of the function by using four basic procedures: reflection, expansion, contraction, and shrinkage. For newly generated points at each step in these procedures, it is noted that the follower's problem is solved to get its optimal solution, so that we can compute its fitness value and continue the next process. To this end, we test the inequalities (8) for feasibility for each new individual. If the inequalities (8) are feasible, we continue the next process; otherwise, the new individual is replaced by the old individual

and go to the next process. Through these procedures, the simplex is updated and replaced by the new simplex. In this way, we generate $n + 1$ new individuals.

The search process is outlined below.

Step 1. From ranked population, select the last $n + 1$ individuals to form a simplex.

Step 2. Apply four basic operators: reflection, expansion, contraction, and shrinkage. In each operator, the inequalities (8) are tested for feasibility for each new individual.

Step 3. Update the simplex.

The result is sorted in preparation for repeating the entire run. To guarantee the global convergence, we select the best N individuals from the current population and all the offspring generated by EDA and NM. These N individuals constitute the next population. If the algorithm is executed to the maximal number of generations and the best solution in the population has not been improved in successive generations, then stop. The best solution found in the last population is then taken as the approximate global optimal solution.

3.3.2. Steps of the Proposed Algorithm. Based on the above procedure, the steps of the proposed algorithm are listed in details as follows.

Step 1 (initialization). Generate N individuals with respect to the upper level variable by means of the uniform design technique. Check the inequalities (8) for feasibility for each individual. Generate the initial population pop(0) by N individuals satisfying the inequalities (8). Set $k = 0$.

Step 2 (evaluation and ranking). Evaluate the fitness of each individual in current population and rank them based on the fitness results.

Step 3 (update). The first M individuals are updated by the estimation of distribution algorithm, and the $n+1$ individuals in the end of the list are updated by the Nelder-Mead simplex method.

Step 4 (selection). Select the best N individuals among the current population and all the offspring. These selected individuals form the next population pop($k + 1$).

Step 5 (termination). An algorithm terminates when it satisfies a stopping criterion; otherwise, set $k = k + 1$ and go to Step 2.

4. Numerical Simulations

To illustrate the feasibility and efficiency of the proposed algorithm, a series of test problems are selected from the literature [11, 12, 15, 19–26]. These problems are composed of different classes of bilevel programming problems, namely, the leader's problems involving linear, quadratic, fractional, nondifferentiable, and nonconvex objective functions. The variety of dimension and functional forms makes it possible to fairly assess the robustness of the proposed approach within tolerably computational time. In the simulation, the proposed EDA-NM is executed to solve all the test problems on MATLAB (R2011b) platform on Inter(R) Core (TM) i7 CPU 870 at 2.93 GHz having 8 G of RAM under Windows XP.

4.1. Comparison of the Proposed EDA-NM with the Existing Algorithms. In this section, a comparison is made between the proposed EDA-NM with the existing algorithms. These approaches consist of hybrid of genetic algorithm and particle swarm optimization (HGAPSO) of Kuo and Han [11], evolutionary algorithm (EA) of Wang et al. [12], evolutionary algorithm (EA) based on duality conditions of Li and Fang [15], the monotonic approach of Tuy et al. [19], penalty function approach of Calvete and Galé [21], dual-relax penalty function approach of Wan et al. [22], and so forth. It is worth mentioning that some of these algorithms are stochastic, whereas others are deterministic. The algorithm proposed in our work is stochastic. To fairly measure the efficiency and stability of a stochastic algorithm, it is usually required to execute the algorithm in many runs and to compare the best and worst solutions found in these runs.

In our work, the proposed algorithm is executed for 50 independent runs on each of the test problems and record the following data: the best solution and the worst solution found in 50 independent runs, the values of the leader's and the follower's objective functions in the best and worst solutions, and average value (Avg.) and standard deviation (Std.) of the leader's objective function values in 50 independent runs. The parameters are set as follows: for n-dimensional problem, the population size is $N = 50$, the value of M is from range of $(N - n)/5$ to $2(N - n)/5$, reflection coefficient, expansion coefficient, contraction coefficient, and shrinkage coefficient are set 1.0, 2.0, 0.5, and 0.5, and the maximum number of iterations is $T = 50$. The algorithm stops when the maximum number of iterations is achieved and the best results are unchanged in 10 successive generations.

The results are listed in Table 1. In Table 1, (x^*, y^*), $(\overline{x}, \overline{y})$ represent the best and worst solutions found by an algorithm, respectively, and "Reference" stands for the algorithm in the corresponding references. The problems with star "$*$" are maximized models, whereas others are minimized BLPP. It can be seen from Table 1 that the best results found by EDA-NM are better than or equal to those by the compared algorithms in the references for all test problems except for problems 6, 10, and 14. For problems 4 and 9, the solutions found by EDA-NM are much better than those by the compared algorithms, which indicates that the compared algorithms cannot find the global solutions of the problems. Especially, for problem 9, the algorithm in Wan et al. [22] finds a local solution (0, 0.75, 0, 0.5, and 0) with the objective value of 10.625, while in our algorithm, all results found by EDA-NM in 50 runs are better than the results given by the compared algorithm in the references. Although the best results found by EDA-NM for problems 6, 10, and 14 are a little bit worse than the compared algorithms for these test problems, the precision of solutions found by our algorithm

Table 1: The results found by EDA-NM and compared algorithms in the corresponding references.

Number	$F(x, y)$				$f(x, y)$		Reference	
	$F(x^*, y^*)$	$F(\overline{x}, \overline{y})$	Avg.	Std.	$f(x^*, y^*)$	$f(\overline{x}, \overline{y})$	$F(x^*, y^*)$	$f(x^*, y^*)$
1* [11]	85.090909	85.090908	85.090909	$2.6e-7$	−50.181818	−50.181818	85.0909	−50.1818
2* [11]	11.0	10.999999	11.000000	$2.7e-7$	−10.999999	−11.0	11	−11
3* [11]	16.0	15.999999	16.000000	$3.1e-7$	−4.000000	−4.000000	16	−4
4* [11]	26.0	25.999999	25.999999	$2.1e-7$	−3.2	−3.199999	25.9827	−3.1932
5 [19]	22.5	22.500001	22.500000	$3.1e-7$	−4.499981	−1.500850	22.500610	−1.523438
6* [20]	469.142857	469.142856	469.142857	$1.8e-7$	8.857162	8.857344	469.14286	8.85714
7 [15]	0.000000	0.000001	0.000000	$2.0e-7$	−54.999690	−54.997967	0	5
8 [21]	252.0	252.000001	252.000001	$2.5e-7$	−8.000000	−7.999999	252	−8
9 [22]	7.500000	7.500001	7.500001	$2.6e-7$	0.000000	0.000000	10.625	−0.5
10* [23]	32.468750	32.468748	32.468749	$3.2e-7$	63.0	62.999998	32.4688	63
11 [24]	0.000000	0.000000	0.000000	$6.6e-7$	0.142411	1.967259	0	1.0440
12 [24]	0.000000	0.000000	0.000000	$3.7e-7$	0.142409	2.002728	0	1.0573
13 [25]	−469.142857	−469.142856	−469.142857	$1.7e-7$	−0.794858	−0.794922	−469.14	−0.795332
14 [26]	−1.688887	−1.688148	−1.688716	$1.8e-6$	4.333333	4.333328	−1.6889	4.3333

Table 2: The results found by EDA with EDA-NM.

Number	$F(x, y)$				$f(x, y)$		EDA-NM	
	$F(x^*, y^*)$	$F(\overline{x}, \overline{y})$	Avg.	Std.	$f(x^*, y^*)$	$f(\overline{x}, \overline{y})$	$F(x^*, y^*)$	$F(\overline{x}, \overline{y})$
1*	85.090908	85.071393	85.088957	$6.2e-3$	−50.181818	−50.175008	85.090909	85.090908
2*	11.0	10.956557	10.956557	$3.4e-5$	−10.999999	−10.999268	11.0	10.999999
3*	15.999999	15.915111	15.199612	$2.8e-5$	−4.0	−3.999974	16.0	15.999999
4*	25.999997	25.999126	25.999458	$2.1e-7$	−3.199999	−3.198546	26.0	25.999999
5	22.5	22.500000	22.500001	$2.8e-7$	−4.500064	−4.499873	22.5	22.500001
6*	469.142857	469.142856	469.142857	$2.1e-7$	8.856955	8.857432	469.142857	469.142856
7	0.000000	0.000001	0.000000	$2.0e-7$	−54.999969	−54.998378	0.000000	0.000001
8	251.999999	252.000001	251.998745	$4.5e-7$	−8.0	−8.0	252.0	252.000001
9	7.500000	7.571070	7.500001	$2.2e-2$	−0.000000	−0.000032	7.500000	7.500001
10*	32.468750	32.393623	32.150867	$1.9e-2$	62.999976	62.999996	32.468750	32.468748
11	0.000000	0.000000	0.000000	$2.8e-7$	0.791998	1.008143	0.000000	0.000000
12	0.000000	0.000000	0.000000	$3.4e-7$	0.851285	1.119564	0.000000	0.000000
13	−469.142857	−469.142846	−469.142857	$2.1e-6$	−0.794871	−0.795079	−469.142857	−469.142856
14	−1.688889	−1.688847	−1.687668	$3.4e-5$	4.333312	4.325765	−1.688887	−1.688148

is higher than that provided by the compared algorithm. In addition, Table 1 shows that all of the worst solutions found by EDA-NM in 50 independent runs for all test problems are the same as or very close to the global optimal solutions. This means that the proposed algorithm is stable.

4.2. Comparison of EDA-NM with EDA. In the previous section we have compared EDA-NM with the existing algorithms. In this section, we would compare EDA-NM with EDA. To conduct fair comparisons, we execute both algorithms for 50 independent runs on each of the test problems. The parameters of EDA are the same as those of EDA-NM. All the results are presented in Tables 2 to 3, in which Table 2 provides the comparison of the results found by EDA-NM and EDA for test problems in 50 runs, and Table 3 shows the best solutions found by EDA and EDA-NM in 50 runs and presented in the related references for all test problems.

It can be seen from Table 2 that the best solutions found by EDA-NM are better than those found by EDA, and the worst solutions found by EDA-NM are as good as or are very close to the best solutions found by EDA. This means that our algorithm can find high-quality approximate global solutions for test problems. Tables 1-2 indicate that EDA-NM has smaller standard deviation, which means that EDA-NM is a stabler method.

In the analysis of the convergence reliability, we focus on the best solutions obtained by EDA-NM and EDA to achieve predefined convergence results. The convergence property of EDA-NM and EDA for different types of test problems is shown in Figures 1, 2, 3, 4, 5, and 6 in which the superiority of EDA-NM over EDA is evident. Figures 1–6 illustrate the performances of EDA-NM and EDA for the test problems

Table 3: The best solution found by EDA-NM, EDA, and compared algorithms in the corresponding references.

Number	EDA-NM	EDA	Reference
1*	(17.454545, 10.909091)	(17.454545, 10.909091)	(17.4545, 10.9091)
2*	(16.0, 11.0)	(15.999999, 11.0)	(16, 11)
3*	(4.0, 4.000000)	(3.999999, 4.000000)	(4, 4)
4*	(0, 0.9, 0, 0.6, 0.4)	(0, 0.899999, 0, 0.599999, 0.4)	(0, 0.8992, 0, 0.5994, 0.3978)
5	(4.499992, 1.500025)	(1.500021, 4.499981)	(4.492188, 1.523438)
6*	(11.142856, 8.857143)	(11.142642, 8.857358)	(11.14286, 8.85714)
7	(25.000177, 29.999830, 5.999823, 9.999915)	(24.999974, 30.000326, 6.000026, 10.000163)	(25, 30, 5, 10)
8	(4.000000, 8.000000)	(4.000000, 7.999999)	(4, 8)
9	(0.500000, 0.500000, 0.000000, 0, 0.000000)	(0.500000, 0.500000, 0.0000000, 0, 0.000000)	(0, 0.75, 0, 0.5, 0) (0, 0.75, 0, 0.5, 0)
10*	(0.000081, 0.625000, 0.375000, 0.000192, 10.0, 5.0, 0, 10.0, 0, 0, 5.000000, 3.0)	(0.006095, 0.625000, 0.375000, 0.000223, 10.0, 5.0, 0, 10.0, 0, 0, 5.0, 3.0)	(0, 0.625, 0.375, 0, 10, 5, 0, 10, 0, 0, 5, 3)
11	(0.402746, 0.211323, 0, 0, 0)	(0.340452, 0.335910, 0, 0, 0)	(0.3298, 0.3571, 0, 0, 0)
12	(0.429411, 0.157992, 0, 0, 0)	(0.322653, 0.371508, 0, 0, 0)	(0.3255, 0.3659, 0, 0, 0)
13	(11.142768, 8.857233)	(11.14286, 8.857140)	(11.14, 8.86)
14	(8.666667, 4.333333)	(8.666656, 4.333312)	(8.6667, 4.3333)

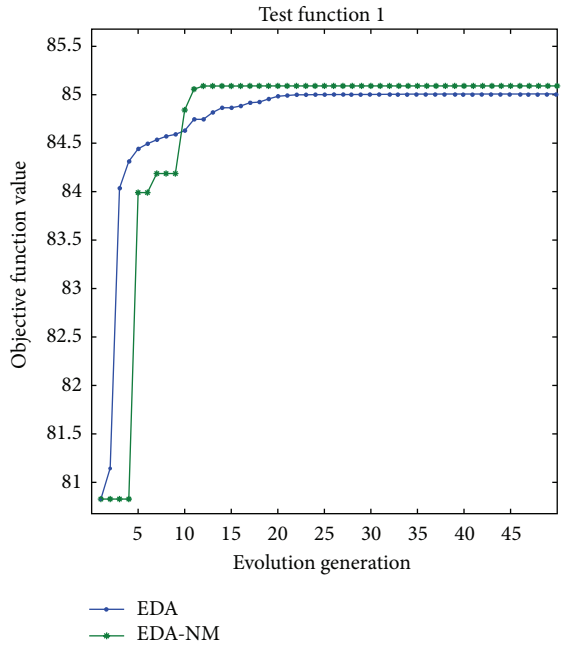

Figure 1: Convergence figure on test problem 1.

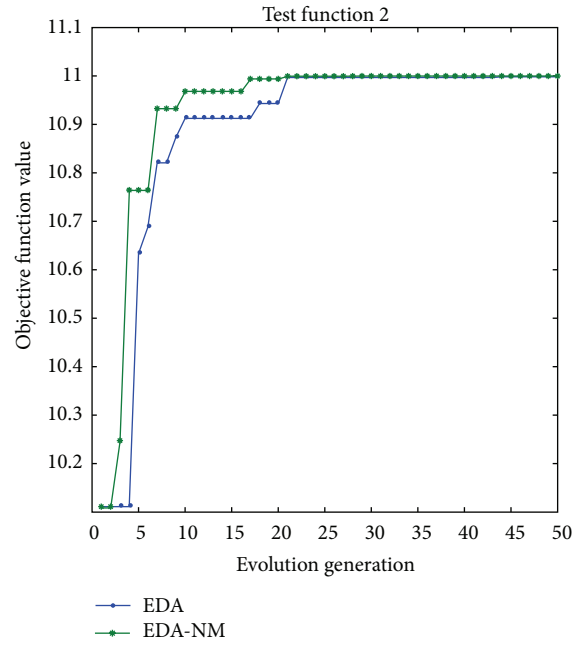

Figure 2: Convergence figure on test problem 2.

1, 2, 4, 5, 8, and 14 to find the global optimum by plotting the best fitness versus the number of iterations for a single run. From these figures, we can see that our hybrid EDA-NM algorithm converges more quickly and less likely to be trapped in local optima than EDA. Furthermore, EDA-NM method's fitness drops quickly in maximized models and is raised rapidly in minimized BLPP at the evolution process. As a result, EDA-NM performs better than EDA from the aspects of computation efficiency.

From the above discussions, it is obvious that our algorithm is effective, is stable, and performs better than the compared algorithms.

4.3. Practical Application: Pollution Charges Problem. In this section, pollution charges decision-making problem based on bilevel programming is used to test the feasibility of the proposed algorithm. Pollution charges problem can be divided into two levels including the pollution tax collected by the government development and the pollutant discharge fees

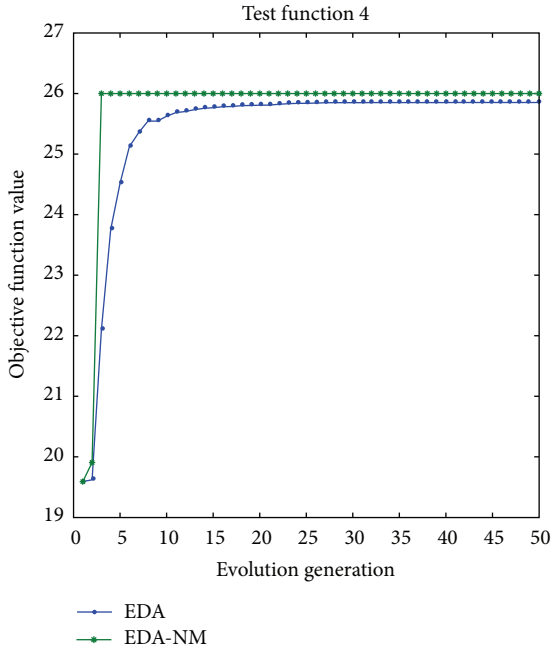

Figure 3: Convergence figure on test problem 4.

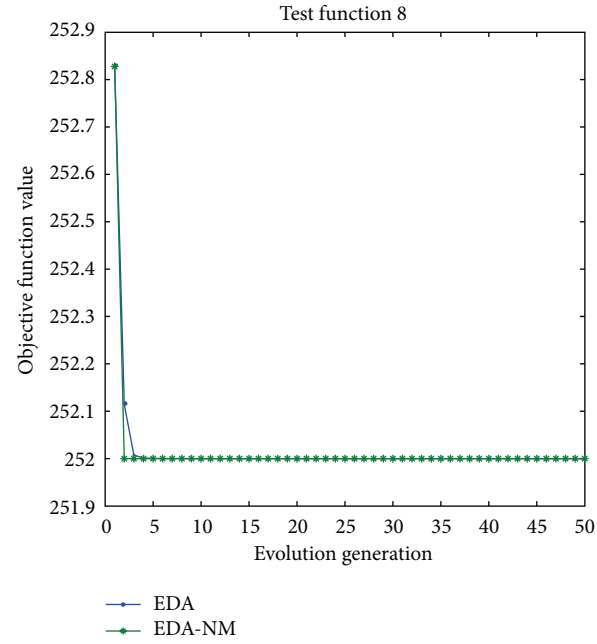

Figure 5: Convergence figure on test problem 8.

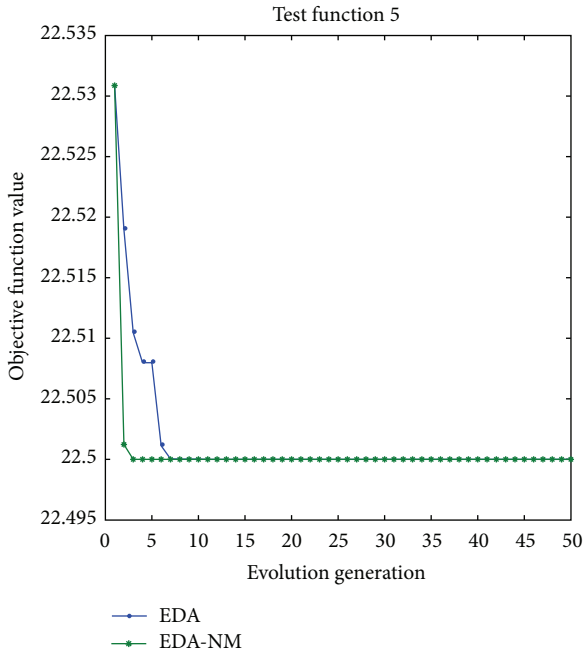

Figure 4: Convergence figure on test problem 5.

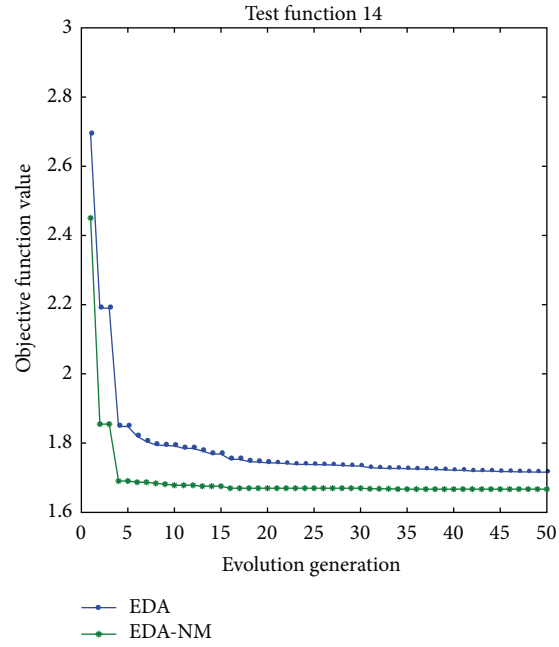

Figure 6: Convergence figure on test problem 14.

paid by the sewage enterprise, and each of them has its own objective function and decision variable. The government development first establishes pollution charges policy, and thereafter the sewage enterprise chooses its strategy according to the decision made by the government development. It is worth mentioning that the government development and the sewage enterprise independently seek their own interest, but influence each other. Therefore, pollution charges problem fits Stackelberg game model and can be solved by the bilevel programming model.

Assume that the sewage enterprise produces n kinds of pollutant substance during the manufacturing process. The following notions will be used henceforth. $x = (x_1, x_2, \ldots, x_n)$, $x \geq 0$: cost prices of pollutant substances set by the government development; $y = (y_1, y_2, \ldots, y_n)$, $y \geq 0$: the pollutant discharge from the sewage enterprise; $a = (a_1, a_2, \ldots, a_n)$: the tax collected by the government

development according to the ratio of the severity of the effects of the pollutant discharge in the environment; $b = (b_1, b_2, \ldots, b_n)$: the tax collected by the government development according to a certain percentage of the amount of the pollutant discharge from the sewage enterprise. Therefore, the total tax of the government department is $a^T x + b^T y$. Subsequently, the sewage enterprise pays for pollutant discharge fees, that is, $x^T y$. In addition, assume that the variable x and the variable y have a certain constraint: $Ax + By \le r$, where A and B are $m \times n$ matrix, $r \in R^m$, and r is the shadow price caused by pollutants. To satisfy needs of maximizing the pollution tax collected by the government development and the cost of pollutant discharge of the sewage enterprise, the nonlinear bilevel programming is formulated as follows:

$$\begin{aligned} &\max_x \quad F(x, y) = a^T x + b^T y, \quad \text{where } y \text{ solves,} \\ &\max_y \quad f(x, y) = x^T y, \\ &\text{s.t.} \quad Ax + By \le r, \\ &\qquad x \ge 0, y \ge 0. \end{aligned} \tag{9}$$

In the following, the numerical values of the parameters in above nonlinear bilevel problem are offered:

$$\begin{aligned} &\max_x \quad F(x, y) = x_1 + 2x_2 + y_1 - y_2, \quad \text{where } y \text{ solves,} \\ &\max_y \quad f(x, y) = x_1 y_1 + x_2 y_2, \\ &\text{s.t.} \quad x_1 + x_2 + y_1 + y_2 \le 6, \\ &\qquad x_1 + y_1 \le 3, \\ &\qquad x_2 - y_1 - y_2 \le -1, \\ &\qquad x \ge 0, y \ge 0. \end{aligned} \tag{10}$$

We use EDA-NM to deal with the bilevel programming problem. The parameters are selected as Section 4.1. We execute the algorithm in 20 independent runs and record the best solutions and the upper objective value as well as the lower objective value. The best optimal solution obtained by the algorithm is $(x_1, x_2, y_1, y_2) = (5/3, 5/3, 4/3, 4/3)$, and the optimal value is $f_1^* = 5$, $f_2^* = 40/9$.

5. Conclusions

In this paper, a hybrid estimation of distribution algorithm and Nelder-Mead simplex method is proposed to solve a class of nonlinear bilevel programming problems. EDA-NM is very easy to be implemented since it does not require gradients information. Moreover, the hybrid algorithm intends to produce faster and more accurate convergence. The performance of the proposed algorithm is tested on a series of test problems, and the results obtained are compared with those reported in the related references. The comparisons demonstrate that the proposed algorithm has a better performance than the compared algorithms. Moreover, the proposed approach is used to solve a practical example about pollution charges problem. Future work would involve the improvement on the probabilistic model of the EDA operator and the further refinement of the algorithm.

Acknowledgments

This work was supported by National Natural Science Foundation of China (no. 61272119), the Science Foundation of the Education, Department of Shaanxi Province of China (no. 2013JK0593), the Scientific Research Foundation of Xi'an Polytechnic University (BS1014), China Postdoctoral Science Foundation (20110491668), and National Natural Science Foundations of China (Grant nos. 11201362 and 110791).

References

[1] J. F. Bard, "Coordination of a multidivisional organization through two levels of management," *Omega*, vol. 11, no. 5, pp. 457–468, 1983.

[2] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol. 153, pp. 235–256, 2007.

[3] S. Dempe, "Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints," *Optimization*, vol. 52, no. 3, pp. 333–359, 2003.

[4] L. N. Vicente and P. H. Calamai, "Bilevel and multilevel programming: a bibliography review," *Journal of Global Optimization*, vol. 5, no. 3, pp. 291–306, 1994.

[5] J. F. Bard, *Practical Bilevel Optimization: Algorithms and Applications*, vol. 30 of *Nonconvex Optimization and Its Applications*, Kluwer Academic, Dordrecht, The Netherlands, 1998.

[6] S. Dempe, *Foundations of Bilevel Programming*, vol. 61 of *Nonconvex Optimization and its Applications*, Kluwer Academic, Dordrecht, The Netherlands, 2002.

[7] R. Mathieu, L. Pittard, and G. Anandalingam, "Genetic algorithm based approach to bi-level linear programming," *RAIRO Recherche Opérationnelle*, vol. 28, no. 1, pp. 1–21, 1994.

[8] K.-M. Lan, U.-P. Wen, H.-S. Shih, and E. S. Lee, "A hybrid neural network approach to bilevel programming problems," *Applied Mathematics Letters*, vol. 20, no. 8, pp. 880–884, 2007.

[9] H. I. Calvete, C. Galé, and P. M. Mateo, "A new approach for solving linear bilevel problems using genetic algorithms," *European Journal of Operational Research*, vol. 188, no. 1, pp. 14–28, 2008.

[10] R. J. Kuo and C. C. Huang, "Application of particle swarm optimization algorithm for solving bi-level linear programming problem," *Computers & Mathematics with Applications*, vol. 58, no. 4, pp. 678–685, 2009.

[11] R. J. Kuo and Y. S. Han, "A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem—a case study on supply chain model," *Applied Mathematical Modelling*, vol. 35, no. 8, pp. 3905–3917, 2011.

[12] Y. P. Wang, Y. C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handing scheme," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 35, no. 2, pp. 221–232, 2005.

[13] K. Deb and S. Sinha, "An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm," *Evolutionary Computation*, vol. 18, no. 3, pp. 403–449, 2010.

[14] Y. Jiang, X. Li, C. Huang, and X. Wu, "Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4332–4339, 2013.

[15] H. Li and L. Fang, "An evolutionary algorithm for solving bilevel programming problems using duality conditions," *Mathematical Problems in Engineering*, vol. 2012, Article ID 471952, 14 pages, 2012.

[16] Z. P. Wan, G. M. Wang, and B. Sun, "A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems," *Swarm and Evolutionary Computation*, vol. 8, pp. 26–32, 2013.

[17] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic, 2002.

[18] P. Pelikan, *Hierachical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, Springer, 2005.

[19] H. Tuy, A. Migdalas, and N. T. Hoai-Phuong, "A novel approach to bilevel nonlinear programming," *Journal of Global Optimization*, vol. 38, no. 4, pp. 527–554, 2007.

[20] G. Wang, X. Wang, Z. Wan, and Y. Lv, "A globally convergent algorithm for a class of bilevel nonlinear programming problem," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 166–172, 2007.

[21] H. I. Calvete and C. Galé, "Bilevel multiplicative problems: a penalty approach to optimality and a cutting plane based algorithm," *Journal of Computational and Applied Mathematics*, vol. 218, no. 2, pp. 259–269, 2008.

[22] Z. Wan, G. Wang, and Y. Lv, "A dual-relax penalty function approach for solving nonlinear bilevel programming with linear lower level problem," *Acta Mathematica Scientia B*, vol. 31, no. 2, pp. 652–660, 2011.

[23] H. Li and Y. P. Wang, "A hybrid genetic algorithm for nonlinear bilevel programming," *Journal of XiDian University*, vol. 29, no. 6, pp. 840–843, 2002.

[24] H. C. Li and Y. P. Wang, "A mixed-encoding genetic algorithm for nonlinear bilevel programming problems," in *Proceedings of the International Joint Conference on INC, IMS and IDC*, 2009.

[25] G. S. Qiu and J. F. Wang, "Penalty function of the nonlinear bilevel programming problem," *Journal of NanChang HangKong University*, vol. 23, no. 3, pp. 61–64, 2009.

[26] H. I. Calvete and C. Galé, "A penalty method for solving bilevel linear fractional/linear programming problems," *Asia-Pacific Journal of Operational Research*, vol. 21, no. 2, pp. 207–224, 2004.

[27] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. binary parameter," *Lecture Notes in Computer Science*, vol. 1141, pp. 178–187, 1996.

[28] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[29] K. T. Fang, "Uniform design: an application of number-theoretic methods to experimental designs," *Acta Mathematicae Applicatae Sinica*, vol. 3, no. 4, pp. 363–372, 1980.

[30] Y. Wang and K. T. Fang, "A note on uniform distribution and experimental design," *Kexue Tongbao*, vol. 26, no. 6, pp. 485–489, 1981.

[31] Y. W. Leung and Y. P. Wang, "Multiobjective programming using uniform design and genetic algorithm," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 30, no. 3, pp. 293–304, 2000.