Enhancing the Solution Method of Linear Bi-Level Programming Problem Based on Combining PSO Algorithm with a Modified Genetic Algorithm

Eghbal Hosseini [a1], Isa Nakhai Kamalabadi [b] , Mohammad Fathi [c]

[a] *University of Payamnoor of Tehran, Department of Mathematics, Tehran, Iran*
[b] *University of Kurdistan, Department of Industrial Engineering, Sanandaj, Iran*
[c] *University of Kurdistan, Department of Electronic, Sanandaj, Iran*

ABSTRACT

The multi-level programming problems are attractive for many researchers because of their application in several areas such as economic, traffic, finance, management, transportation and so on. Among these, the bi-level programming problem (BLPP) is an appropriate tool to model these real problems. It has been proven that the general BLPP is an NP-hard problem, so it is a practical and complicated problem therefore solving this problem would be significant. However the literature shows several algorithms to solve different forms of the bi-level programming problems (BLPP), but there is no any hybrid approach of combining of two meta-heuristic algorithms. The most important part of this paper is combining particle swarm optimization (PSO), which is a continuous approach, with a proposed modified genetic algorithm (MGA), which is a discrete algorithm, using a heuristic function and constructing an effective hybrid approaches (PSOMGA).Using the Karush-Kuhn-Tucker conditions the BLPP is converted to a non-smooth single level problem, and then it is smoothed by a new heuristic method for using PSOMGA. The smoothed problem is solved using PSOMGA which is a fast approximate method for solving the non-linear BLPP. The presented approach achieves an efficient and feasible solution in an appropriate time which has been evaluated by solving test problems.

Keywords: Particle Swarm Optimization, Genetic Algorithm, Linear bi-level programming problem, Karush-Kuhn-Tucker conditions.

1. Introduction

[1] Corresponding author
E-mail addresses: eghbal_math@yahoo.com

It has been proven that the bi-level programming problem (BLPP) is an NP-Hard problem [1, 2]. Several algorithms have been proposed to solve BLPP [11, 12, 13, 21, 25, 26, 27, 28, 29, 31, 32, 33]. These algorithms are divided into the following classes: global techniques, enumeration methods, transformation methods [3, 4, 22, 23], meta heuristic approaches, fuzzy methods [5, 6, 7, 8, 24], primal-dual interior methods [13]. In the following, these techniques are shortly introduced.

1.1. Global techniques

All optimization methods can be divided into two distinctive classes: local and global algorithms. Local ones depend on initial point and characteristics such as continuity and differentiability of the objective function. These algorithms search only a local solution, a point at which the objective function is smaller than at all other feasible points in vicinity. They do not always find the best minima, that is, the global solution. On the other hand, global methods can achieve global optimal solution. These methods are independent of initial point as well as continuity and differentiability of the objective function [9, 10, 11, 12, 34, 35].

1.2. Enumeration methods

Branch and bound is an optimization algorithm that uses the basic enumeration. But in these methods we employ clever techniques for calculating upper bounds and lower bounds on the objective function by reducing the number of search steps. In these methods, the main idea is that the vertex points of achievable domain for BLPP are basic feasible solutions of the problem and the optimal solution is among them [14].

1.3. Meta heuristic approaches

Meta heuristic approaches are proposed by many researchers to solve complex combinatorial optimization. Whereas these methods are too fast and known as suitable techniques for solving optimization problems, however, they can only propose a solution near to optimal. These approaches are generally appropriate to search global optimal solutions in very large space whenever convex or non-convex feasible domain is allowed. In these approaches, BLPP is transformed to a single level problem by using transformation methods and then meta- heuristic methods are utilized to find out the optimal solution [15, 16, 17, 18, 19, 25, 36-40].

However there are meta– heuristic approaches and their combinations to solve optimization problems [25, 37], but there is no any approach which combines a continuous algorithm and a discrete one in these approaches. In this paper, the authors have tried to combine particle swarm optimization, a continuous approach, and proposed modified genetic algorithm, a discrete algorithm, to solve linear BLPP.

The remainder of the pages is structured as follows: problem formulation and smooth method to the BLPP are introduced in Section 2. The algorithm based on combining modified genetic algorithm and particle swarm optimization is proposed in Section 3. Computational results are presented for our approaches in Section 4. As result, the paper is finished in Section 5 by presenting the concluding remarks.

2. Problem formulation and Smoothing Method

It has been proven that the bi-level programming problem (BLPP) is an NP-Hard problem [1, 2]. Several algorithms have been proposed to solve BLPP [11, 12, 13, 21, 25, 26, 27, 28, 29, 31, 32, 33]. These algorithms are divided into the following classes: global techniques [9, 10, 11, 12, 34, 35], enumeration methods [14], transformation methods, meta heuristic

approaches, fuzzy methods [5, 6, 7, 8, 24], primal-dual interior methods [13]. In the following, these techniques are shortly introduced.

The BLPP is used frequently by problems with decentralized planning structure. It is defined as [20]:

$$\begin{aligned} &\min_{x} F(x,y) = a^{T}x + b^{T}y \\ &\text{s.t} \\ &\min_{y} F(x,y) = c^{T}x + d^{T}y \\ &\text{s.t} \\ &\quad Ax + By \le r, \\ &\quad x,y \ge 0. \end{aligned} \tag{1}$$

The feasible region of the BLP problem is

$$S = \{(x,y) | Ax + By \le b\ , x,y \ge 0\}. \tag{2}$$

On the other hand, if x be fixed, the feasible region of the follower can be explained as

$$S = \{y | By \le b - Ax\ , x,y \ge 0\}. \tag{3}$$

Based on the above assumptions, the follower rational reaction set is

$$P(x) = \{y \in \text{argming}(x,y), y \in S(x)\}. \tag{4}$$

where the inducible region is as follows

$$IR = \{(x,y) \in S, y \in P(x)\}. \tag{5}$$

Finally, the bi-level programming problem can be written as

$$\min\{F(x,y) | (x,y) \in IR\}. \tag{6}$$

If there is a finite solution for the BLP problem, we define feasibility and optimality for the BLP problem as

$$S = \{(x,y) | Ax + By \le b, x,y \ge 0\}. \tag{7}$$

Definition 2.1:

Every point such as $(x,y)$is a feasible solution to bi-level problem if $(x,y) \in IR$

Definition 2.2:

Every point such as (x^{*},y^{*}) is an optimal solution to the bi-level problem if

$$F(x^{*}.y^{*}) \le F(x,y)\ \forall\ (x,y) \in IR. \tag{8}$$

A summary of important properties for convex problem are as follows, which $F: S \dot{\rightarrow} R^n$ and S is a nonempty convex set in R^n:

(1) The convex function f is continuous on the interior of S.

(2) Every local optimal solution of F over a convex set $X \subseteq S$ is the unique global optimal solution.

(3) If $\nabla F(x^*)=0$, then x^* is the unique global optimal solution of F over S.

Using KKT conditions problem (1) can be converted into the following problem:

$$\min_{x} F(x,y) = a^T x + b^T y$$

$$\begin{aligned} s.t \quad & \mu B = -d, \\ & \mu(Ax + By - r) = 0, \\ & Ax + By - r \leq 0, \\ & x, y, \mu \geq 0. \end{aligned} \tag{9}$$

To convert the inequality constraint to an equality constraint, the positive slack variable v is added:

$$\min_{x} F(x,y) = a^T x + b^T y$$

$$\begin{aligned} s.t \quad & \mu B = -d, \\ & \mu(Ax + By - r) = 0, \\ & Ax + By - r + v = 0, \\ & x, y, \mu, v \geq 0. \end{aligned} \tag{10}$$

Let $\alpha = (r - Ax - By)$ then the problem can be written as follows:

$$\min_{x} F(x,y) = a^T x + b^T y$$

$$\begin{aligned} s.t \quad & \mu B = -d, \\ & \mu\alpha = 0, \\ & v - \alpha = 0, \\ & x, y, \mu, v, \alpha \geq 0. \end{aligned} \tag{11}$$

3.1 Modified Genetic algorithm (MGA)

In this section, a modified genetic algorithm is proposed then basic and general concepts related to particle swarm optimization algorithms are discussed. Finally the hybrid method of both algorithms is proposed.

Genetic algorithms are global methods that are used for global searches. As the previous researchers indicate [11, 15, 16] the basic characteristics of these algorithms consist of:

1. Initial population of solution is produced randomly. Some of the genetic algorithms use other Meta heuristic method to produce the initial population.

2. Genetic algorithms use a lot of feasible solutions. Therefore they usually avoid local optimal solutions.
3. Genetic algorithms used to solve very large problems with many variables.
4. These algorithms are simple and do not need extra conditions such as continuity and differentiability of objective functions.
5. Genetic algorithms usually gain several optimal solutions instead unique optimal solution. This property is useful for multi objective function and multi- level programming.
6. These algorithms are inherently discrete.

In the proposed genetic algorithm, each feasible solution of BLPP usually is transformed by string of characters from the binary alphabet that is called chromosome. The genetic algorithm works as follows:

Initial generation, that is generated randomly, is divided in overall the feasible space similarly. Then chromosomes are composed together to construct new generation. This process continues till to get appropriate optimal solution. The general genetic algorithm process as follows:

$A\lg orithm\ 1: GA\ to\ solve\ BLPP$

$1: t = 0$
$2:$ initialize $P(t)$
$3:$ evaluate $P(t)$
$4:$ While not *terminate* do
$5:\ P'(t) = \text{recombine}\ P(t)$
$6:\ P''(t) = \text{mutate}\ P'(t)$
$7:$ evaluate $P''(t)$
$8:\ P(t+1) = \text{select}\ (P''(t) \bigcup Q)$
$9:\ t = t+1$
$10:$ End of While
$11:$ End.

Where *P(t)* is a population of chromosomes in *t-th* generation and *Q* is a set of chromosomes in the current generation which are selected.

In the suggested method, every chromosome is demonstrated by a string. This string consists of $k + l + 2p$ corresponding variables x, y, μ, v, also these chromosomes are applied in problem (15) that it is created by using Karush -Kuhn –Tucker (KKT) conditions and proposed smoothed method for TLPP. Using slack variables, such as w, v, u, problem (5) is prepared for using genetic algorithm:

Now the chromosomes are applied according the following rules [15]:

If the *i-th* component of the chromosome is equal to zero, then $\alpha_i = 0, v_i \geq 0$ Else $\alpha_i \geq 0, v_i = 0$.
If the *j-th* component of the chromosome is equal to zero, then $\alpha_j = 0, \mu_j \geq 0$ Else $\alpha_j \geq 0, \mu_j = 0$.

Theorem 3.1:

(x^*, y^*) is the optimal solution to the problem (1) if and only if there exists such that $(x^*, y^*, \alpha^*, \mu^*, v^*)$ is the solution of the problem (5).

Proof :
The proof of this theorem was given by [17].

The MGA steps are proposed as follows:

Step 1: Generating the initial population

The initial population includes solutions in the feasible region that are called achievable chromosomes. These chromosomes are generated by solving the following problem:

$$\min_{y} F(x,y) = c^T x + d^T y$$

$$s.t$$

$$Ax + By \leq r,$$

$$x, y \geq 0. \qquad (12)$$

Step 2: Keeping the present best chromosome in an array

The best chromosome is kept in the array at the each iteration. This process continues till the algorithm is finished, then the best chromosome is found in the array as the optimal solution.

Step 3: Crossover operation

Crossover is a major operation to compose a new generation. In this stage two chromosomes are selected randomly and they are combined to generate a new chromosome. In the new generation components are created by the following rules:

1. The i-th component of the first child is replaced by the sum of the i-th components of parents (i=1,2,…,k+l). The operation sum is defined as follows:

$$X_i + Y_i = \left[\frac{x_i + y_i}{2}\right]$$

That X_i, Y_i is th i-th component of chromosomesX, Y. The other components are remained the same as the first parent.

2. The (k+l+i)-th component of the second child is replaced by the sum of the (k+l+i)-th components of parents i=1,2,…,2p. The operation sum is defined as above. The other components are remained the same as the second parent.

For example, by applying the present method to the following parents, and k= 5, l=4, p=3we generate the following children:

Parents:	Children:
92745 1036 786 123	54565 0057 786 123
27386 0178 193 321	27386 0178 484 222

Step 4: Mutation

The main goal of mutation in GA is to avoid trapping in local optimal solutions. In this algorithm each chosen gene of every chromosome, mutates according following function:

$$M(i) = \begin{cases} i+1 & if\ i \neq 9 \\ 0 & if\ i = 9 \end{cases} \qquad (13)$$

In fact if the value of the chosen gene be i, it will be changed to i+1 which $i \neq 9$ and if the value of the chosen gene be 9, it will be changed to 0.

For example, by applying the present mutation operation to the following chromosome, and k= 5, l=4, p=3, we have:

Before Mutation:	After Mutation:
17834 9267 193 052	28945 0378 204 163

Step 5: Selection

The chromosomes of the current population are arranged in descending order of fitness values. Then we select a new population similar to the size of the first generation. If the number of the generations is sufficient we go to the next step, otherwise the algorithm is continued by the step3.

Step 6: Termination

The algorithm is terminated after a maximum generation number. The best produced solution that has been recorded in the algorithm is reported as the best solution to TLPP by proposed GA algorithm.

3.2 Particle Swarm Optimization (PSO)

Swarm intelligence system is an artificial intelligence technique which is usually made up of a population of simple particles cooperation locally with one another and with their environment.

PSO is an evolutionary method based on swarm intelligence in animals such as birds, fishes, buffalos and so on. Swarm motion is better than single movement in birds for example, it has been proven that parks in affluent neighborhoods can be attracted more birds than parks in poor neighborhoods. In fact the animals want to gain more food which is optimal solution for them. In an optimization programming problem, for examples in the BLPP, each solution in feasible region is correspond with a particle in PSO which this algorithm try to find the optimal solution same as the animals to find more food. In fact a particle swarm is a group of particles, where each particle is a moving object that is through the search space and is attracted to previously visited locations with high fitness. In comparison to the evolutionary approaches, in the particle swarm optimization particles neither reproduce nor get replaced by other particles.

Because PSO simply solves discontinuous and non-convex problems, therefore it is suitable tool for solving BLPP. It is a method based on population search. In each iteration, PSO moves from a set of particle positions to the better one set with improvement optimal solution. It is inherently continuous.

PSO has the following steps:

1. The initial population of particles and their velocity are produced randomly in the following feasible region.
2. The best objective function for each particle in each iteration is kept. Also the global best objective function is defined.
3. The global best and the best objective function for each particle are updated as follows:

$$v^i[t+1] = wv^i[t] + c_1 r_1 (x^{ibest}[t] - x^i[t]) + c_2 r_2 (x^{gbest}[t] - x^i[t]) \quad (14)$$

$$x^i[t+1] = x^i[t] + v^i[t+1]$$

4. If termination conditions are not satisfied, the above steps will be continued from step two. Otherwise the algorithm will be finished.

3.3 Hybrid Algorithm by Combining PSO and MGA (PSOMGA)

As mention previously, the genetic algorithm searches in the discrete space but the particle swarm optimization searches in the continuous one. Therefore to hybrid these two approaches it is necessary that we round the positions of particles, which are continuous, to use genetic algorithm.

In this method initial population is produced using PSO algorithm. Then position and their velocity are updated by the step 3 in PSO and the new population with better objective functions will be made. After that, the position of particles will be rounded then MGA is applied to the new population by these steps: crossover operation, mutation and selection. In the next iteration PSO algorithm is applied to the obtained population by MGA. The algorithm is continued while termination condition is satisfied. In fact in this proposed hybrid method by combining PSO and MGA the genetic algorithm is used after the particle swarm optimization algorithm. Therefore the convertor function should be defined to convert particles to the acceptable points for MGA (chromosome).

Definition 3.1:

We define the convertor function to round an arbitrary number x as follows:

$$INT^*(x) = \begin{cases} n & if \quad x < n + \frac{1}{2} \\ n+1 & if \quad x \geq n + \frac{1}{2} \end{cases}, \tag{15}$$

For $x \in R$. Also we define for $x \in R^K$, $INT^*(x) = \left(INT^*(x_1), INT^*(x_2), \dots, INT^*(x_k)\right)$.

Example 1:

To more illustrate the proposed function that converts the particles to the point using GA, consider $X_1 = (0.4,0.44), X_2 = (0.7,0.5), X_3 = (0.3,0.6), X_4 = (0.9,0.2), X_5 = (1.6,1.3)$.

$$INT^*(X_1) = (0,0)$$
$$INT^*(X_2) = (1,1)$$
$$INT^*(X_3) = (0,1)$$
$$INT^*(X_4) = (1,0)$$
$$INT^*(X_5) = (2,1)$$

These points have been shown in figure 1. The green points are the produced particles by PSO and the red points are the acceptable point for GA (chromosome). It is easy to see that each green point is attracted to the nearest red point by the proposed convertor function.

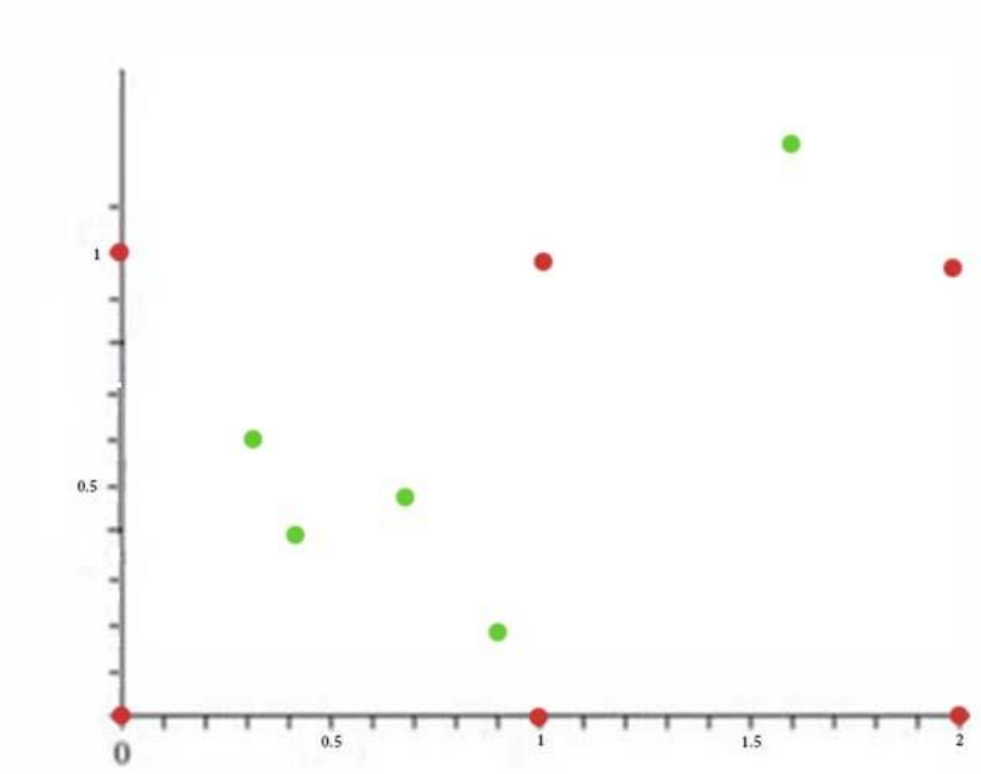


Figure 1 – converting particles to chromosomes for Example1

Now everything is prepared to propose the new hybrid approach. The algorithm steps as follows:

Step 1: initialization

The initial population of particles and their velocity are produced randomly in the problem (12).

Step 2: Keeping the present best particles in an array

The best objective function for each particle is kept in the array at the each iteration. Also the global best objective function is saved. This process continues till the algorithm is finished.

Step 3: Updating

The global best and the best position for each particle are updated according to (14).

Step 4: Checking the optimal solution

If $\mathrm{d}(\mathrm{F}(x^{gbest}[t+1]), \mathrm{F}(x^{gbest}[t])) < \varepsilon_1$ then go to step 8. Otherwise go to the next step.

Step 5: Converting particles to chromosome

In this step all particles convert to chromosome using proposed convertor function.

Step 6: Crossover operation

In this step each particle is correspond with a chromosome. The chromosomes of the current population are arranged in descending order of objective function values. Then numbers of the best chromosomes are selected to use crossover operation for positions by the proposed rules in step 3 of section 3.1. The velocities are changed too as follows:

$$Minus = c_1 r_1 v_1^i[t+1] - c_2 r_2 v_2^i[t+1]$$
$$Sum = c_1 r_1 v_1^i[t+1] + c_2 r_2 v_2^i[t+1]$$
$$v_1^i[t+1] = Minus$$
$$v_2^i[t+1] = Sum$$

Which v_1^i, v_2^i are respectively the velocities of first and second particles in the *i-th* iteration.

Step 7: Mutation

In this step each chosen gene of every chromosome, mutates like step 4 of section 3.1 as follows:

If the value of the chosen gene be 0, it will be changed to 1 and if the value of the chosen gene be 1, it will be changed to 0. Also the velocities are changed according to the following rule:

$$v^i[t+1] = v^i[t+1] + \frac{\pi}{2}$$

Step 8: Selection

We select a new population similar to the size of the first generation by combing the obtained particles in two above steps and number of the particles in current population before applying MGA.

Step 9: Termination

If $\mathrm{d}(\mathrm{F}(x^{gbest}[t+1]), \mathrm{F}(x^{gbest}[t])) < \varepsilon_1$ then the algorithm is finished and $x^{gbest}[t+1]$ is the best solution by the proposed algorithm. Otherwise, let k=k+1 and go to the step 2. That d is the following metric:

$$\mathrm{d}\big(\mathrm{F}(x^{gbest}[t+1]), \mathrm{F}(x^{gbest}[t])\big) = (\sum_{i=1}^{k+l+2p} (\mathrm{F}({x_i}^{gbest}[t+1]), \mathrm{F}({x_i}^{gbest}[t]))^2)^{\frac{1}{2}}.$$

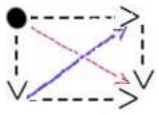
Because the authors going to gain the best optimal solution, the Termination condition, $\mathrm{d}(\mathrm{F}(x^{gbest}[t+1]), \mathrm{F}(x^{gbest}[t])) < \varepsilon_1$, will be checked in both steps 4 and 9. In fact this condition will be checked before and after MGA which rounds the solutions. Also both of steps crossover and mutation in the MGA are applied to the velocities of particles. To illustrate these two operations following example is proposed.

Example 2:

Consider following particles and their velocities that are the parents in MGA:

Using step 6 in the proposed hybrid algorithm after mutates the velocities are changed as follows:

Also after using step 5 velocities of children are changed that the red direction is resultant of two directions of parents and the blue one is difference of them.



Definition 3.2: A metric space is pair (X,d) where X is a set and d is a metric on X and:

(i) $d \geq 0$,

(ii) $d(x,y) = 0 \Leftrightarrow x = y$,

(iii) $d(x,y) = d(y,x)$,

(iv) $d(x,y) \leq d(x,z) + d(z,y)$.

Definition 3.3: A sequence $\{x_n\}$ is said to Cauchy if for every $\varepsilon > 0$ there is an N such that

$$\forall_{m>r>N} \qquad |x_m - x_r| < \varepsilon.$$

Following theorems show that the proposed algorithm is convergent.

Theorem 3.2: Every Cauchy sequence in real line and complex plan is convergent.

Proof:

Proof of this theorem is given in [30].

Theorem 3.3: Sequence $\{F_k\}$ which was proposed in above algorithm is convergent to the optimal solution, so that the algorithm is convergent.

Proof:

Let $(F_l) = (F(t^l)) = (F(t_1^l), F(t_2^l), \dots, F(t_{n+2m}^l))$= $(F_1^{(l)}, F_2^{(l)}, \dots, F_{n+2m}^{(l)})$.

According to step 4

$d(F_{k+1}, F_k) = \mathrm{d}\left(\mathrm{F}(\mathrm{t}^{\mathrm{k+1}}), \mathrm{F}(\mathrm{t}^{\mathrm{k}})\right) = (\sum_{i=1}^{n+2m}(F(t_i^{k+1}) - F(t_i^k))^2)^{\frac{1}{2}} < \varepsilon_1$

Therefore $(\sum_{i=1}^{n+2m}\left(F(t_i^{k+1}) - F(t_i^k)\right)^2) < {\varepsilon_1}^2$. There is large number such as N which k+1>k>N and j=1,2,…,2m+n we have:

$$(F_j^{(k+1)} - F_j^{(k)})^2 < {\varepsilon_1}^2 \text{ , therefore } \left|F_j^{(k+1)} - F_j^{(k)}\right| < \varepsilon_1 \qquad (16)$$

Now let $m = k+1, r = k$ *then we have*

$$\forall_{m>r>N} \qquad \left|F_j^{(m)} - F_j^{(r)}\right| < \varepsilon_1.$$

This shows that for each fixed j, $(1 \leq \mathrm{j} \leq 2\mathrm{m} + \mathrm{n})$, the sequence $(\mathrm{F_j^{(1)}}, \mathrm{F_j^{(2)}}, \dots)$ is Cauchy of real numbers, then it converges by theorem 3.2.

Say, $F_j^{(m)} \to F_j$ as $m \to \infty$. Using these 2m+n limits, we define $F = (F_1, F_2, \dots, F_{2m+n})$. From (16) and m=k+1, r=k,

$$d(F_m, F_r) < \varepsilon_1$$

Now if $r \to \infty$, by $F_r \to F$ we have $d(F_m, F) \leq \varepsilon_1$.

This shows that F is the limit of (F_m) and the sequence is convergent by definition 3.2 therefore proof of theorem is finished.

4. Computational results

To illustrate PSOMGA algorithm, two standard examples will be solved in this section.

Example 3 [17]

Consider the following linear bi-level programming problem:

$$\min x - 4y$$

$$\begin{aligned} s.t \min\ & y \\ s.t\ & x + y \geq 3, \\ & -2x + y \leq 0, \\ & 2x + y \leq 12, \\ & 3x - 2y \leq 4, \\ & x, y \geq 0. \end{aligned}$$

Using KKT conditions, the following problem is obtained:

$$\min x - 4y$$

$$\begin{aligned} s.t\ & -\mu_1 + \mu_2 + \mu_3 - 2\mu_4 = -1 \\ & \mu_1(-x - y + 3) = 0, \\ & \mu_2(-2x + y) = 0, \\ & \mu_3(2x + y - 12) = 0, \\ & \mu_4(3x - 2y - 4) = 0, \\ & -x - y + 3 \leq 0, \\ & -2x + y \leq 0, \\ & 2x + y - 12 \leq 0, \\ & 3x - 2y - 4 \leq 0, \\ & x, y, \mu_1, \mu_2, \mu_3, \mu_4 \geq 0. \end{aligned}$$

This problem have been solved using PSOMGA, Optimal solution have been presented according to Table 1. Behavior of the variables in Example 3 has been shown in figure 2. We have present behavior of the x and optimal solution (OS) with different value of α, μ, v in figure 3.

Example 4 [17]

Consider the following linear bi-level programming problem.

$$\min 4x + y_1 + y_2$$

$$\text{s.t} \min x + 3y_1$$

$$\text{s.t} \quad x + y_1 + y_2 \geq \frac{25}{9},$$

$$x + y_1 \leq 2,$$

$$y_1 + y_2 \leq \frac{8}{9},$$

$$x, y_1, y_2 \geq 0.$$

After applying KKT conditions and smoothing method, the above problem will be transformed into the problem which will be solved using PSOMGA. Optimal solution for this example is presented according to Table 2. Behavior of the variables has been show in figure 4 and we have present behavior of the x and optimal solution (OS) with different value of α, μ, v in figure 5.

5. Conclusion and future work

The main difficulty of the multi-level programming problem is that after using the KKT conditions the non-linear constraints are appeared. In this paper was attempted to remove these constraints by the proposed theorem, slack variables and proposed PSOMGA algorithm. As mentioned previously the authors have been combined two continuous and discrete effective approaches to the linear BLPP which this form of combining has not been studied by any researchers. According to the Tables the proposed method presents optimal solution in appropriate time and iterations. In the future works, the following should be researched:

(1) Examples in the larger sizes can be supplied to illustrate the efficiency of the proposed algorithm.
(2) Showing the efficiency of the proposed algorithm for solving other kinds of BLPP such as quadratic and non-linear.
(3) Solving other kinds of multi-level programming problem such as tri-level programming problem.

Nomenclature

$F_1(x, y, z)$	Objective function of the first level in the TLPP
$F_2(x, y, z)$	Objective function of the second level in the TLPP
$F_3(x, y, z)$	Objective function of the third level in the TLPP
$g(x, y, z)$	Constraints in the TLPP
w	Slack variable
v	Slack variable
$F(x, y)$	Objective function of the first level in the BLPP
$f(x, y)$	Objective function of the first level in the BLPP
$g(x, y)$	Constraints in the BLPP
S	A nonempty convex set
L	Lagrange function
α	Lagrange Coefficient
β	Lagrange Coefficient
μ	Lagrange Coefficient

P	Initial population
P'	Crossover population
P''	Mutation population
Q	Set of chromosomes in the current generation
(x^*, y^*, z^*)	Optimal solution for the TLPP
(x^*, y^*)	Optimal solution for the BLPP

References

[1] J.F. Bard, Some properties of the bi-level linear programming, Journal of Optimization Theory and Applications (1991) 68 371–378.

[2] L. Vicente, G. Savard, J. Judice, Descent approaches for quadratic bi-level programming, Journal of Optimization Theory and Applications (1994) 81 379–399.

[3] Lv. Yibing, Hu. Tiesong, Wang. Guangmin , A penalty function method Based on Kuhn–Tucker condition for solving linear bilevel programming, Applied Mathematics and Computation (2007) 1 88 808–813.

[4] G. B. Allende, G. Still, Solving bi-level programs with the KKT-approach, Springer and Mathematical Programming Society (2012) 1 31:37 – 48.

[5] M. Sakava, I. Nishizaki, Y. Uemura, Interactive fuzzy programming for multilevel linear programming problem, Computers & Mathematics with Applications (1997) 36 71–86.

[6] S Sinha, Fuzzy programming approach to multi-level programming problems, Fuzzy Sets And Systems (2003) 136 189–202.

[7] S. Pramanik, T.K. Ro, Fuzzy goal programming approach to multilevel programming problems, European Journal of Operational Research (2009) 194 368–376.

[8] S.R. Arora, R. Gupta, Interactive fuzzy goal programming approach for bi-level programming problem, European Journal of Operational Research (2007) 176 1151–1166.

[9] J. Nocedal, S.J. Wright, 2005 Numerical Optimization, Springer-Verlag, , New York.

[10] A.AL Khayyal, Minimizing a Quasi-concave Function Over a Convex Set: A Case Solvable by Lagrangian Duality, proceedings, I.E.E.E. International Conference on Systems, Man,and Cybemeties, Tucson AZ (1985) 661-663.

[11] R. Mathieu, L. Pittard, G. Anandalingam, Genetic algorithm based approach to bi-level Linear Programming, Operations Research (1994) 28 1–21.

[12] G. Wang, B. Jiang, K. Zhu, (2010) Global convergent algorithm for the bi-level linear fractional-linear programming based on modified convex simplex method, Journal of Systems Engineering and Electronics 239–243.

[13] W. T. Wend, U. P. Wen, (2000) A primal-dual interior point algorithm for solving bi-level programming problems, Asia-Pacific J. of Operational Research, 17.

[14] N. V. Thoai, Y. Yamamoto, A. Yoshise, (2002) Global optimization method for solving mathematical programs with linear complementary constraints, Institute of Policy and Planning Sciences, University of Tsukuba, Japan 978.

[15] S.R. Hejazi, A. Memariani, G. Jahanshahloo, (2002) Linear bi-level programming solution by genetic algorithm, Computers & Operations Research 29 1913–1925.

[16] G. Z. Wang, Wan, X. Wang, Y.Lv, Genetic algorithm based on simplex method for solving Linear-quadratic bi-level programming problem, Computers and Mathematics with Applications (2008) 56 2550–2555.

[17] T. X. Hu, Guo, X. Fu, Y. Lv, (2010) A neural network approach for solving linear bi-level programming problem, Knowledge-Based Systems 23 239–242.

[18] B. Baran Pal, D .Chakraborti , P. Biswas, (2010) A Genetic Algorithm Approach to Fuzzy Quadratic Bi-level Programming, Second International Conference on Computing, Communication and Networking Technologies.

[19] Z. G.Wan, Wang, B. Sun, (2012) A hybrid intelligent algorithm by combining particle Swarm optimization with chaos searching technique for solving nonlinear bi-level programming Problems, Swarm and Evolutionary Computation.

[20] J.F. Bard, Practical bi-level optimization: Algorithms and applications, Kluwer Academic Publishers, Dordrecht, 1998.

[21] J.F. Bard, Some properties of the bi-level linear programming, Journal of Optimization Theory and Applications 68 (1991) 371–378.

[22] S Hosseini, E & I.Nakhai Kamalabadi. Line Search and Genetic Approaches for Solving Linear Tri-level Programming Problem International Journal of Management, Accounting and Economics Vol. 1, No. 4, 2014.

[23] S Hosseini, E & I.Nakhai Kamalabadi. aylor Approach for Solving Non-Linear Bi-level Programming Problem ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 5, No.11 , September. 5.

[24] S Hosseini, E & I.Nakhai Kamalabadi. Two Approaches for Solving Non-linear Bi-level Programming Problem , Advances in Research Vol. 4, No.3 , 2015. ISSN: 2348-0394

[25] J. Yan, Xuyong.L, Chongchao.H, Xianing.W, Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bi-level programming problem, Applied Mathematics and Computation 219 (2013) 4332–4339.

[26] Xu, P, & L. Wang. An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions, Computers & Operations Research, Volume 41, January, Pages 309-318 (2014).

[27] Wan, Z, L. Mao, & G. Wang. Estimation of distribution algorithm for a class of nonlinear bilevel programming problems, Information Sciences, Volume 256, 20 January, Pages 184-196(2014).

[28] Zheng, Y, J. Liu, & Z. Wan. Interactive fuzzy decision making method for solving bi-level programming problem, Applied Mathematical Modelling, Volume 38, Issue 13, 1 July, Pages 3136-3141(2014).

[29] Zhang , G, J. Lu , J. Montero , & Y. Zeng , Model. solution concept, and Kth-best algorithm for linear tri-level, programming Information Sciences 180 481–492 (2010) .

[30] A. Silverman. Richard, Calculus with analytic geometry, ISBN:978-964-311-008-6, 2000.

[31] Y. Zheng, J. Liu, Z. Wan, Interactive fuzzy decision making method for solving bi-level programming problem, Applied Mathematical Modelling, Volume 38, Issue 13, 1 July 2014, Pages 3136-3141.

[32] Y. Jiang, X. Li, C. Huang, X. Wu, An augmented Lagrangian multiplier method based on a CHKS smoothing function for solving nonlinear bi-level programming problems, Knowledge-Based Systems, Volume 55, January 2014, Pages 9-14.

[33] X. He, C. Li, T. Huang, C. Li, Neural network for solving convex quadratic bilevel programming problems, Neural Networks, Volume 51, March 2014, Pages 17-25.

[34] Z. Wan, L. Mao, G. Wang, Estimation of distribution algorithm for a class of nonlinear bilevel programming problems, Information Sciences, Volume 256, 20 January 2014, Pages 184-196.

[35] P. Xu, L. Wang, An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions, Computers & Operations Research, Volume 41, January 2014, Pages 309-318.

[36] E. Hosseini, I.Nakhai Kamalabadi, A Genetic Approach for Solving Bi-Level Programming Problems, Advanced Modeling and Optimization, Volume 15, Number 3, 2013.

[37] E. Hosseini, I.Nakhai Kamalabadi, Solving Linear-Quadratic Bi-Level Programming and Linear-Fractional Bi-Level Programming Problems Using Genetic Based Algorithm, Applied Mathematics and Computational Intelligence, Volume 2, 2013.

[38] E. Hosseini, I.Nakhai Kamalabadi, Taylor Approach for Solving Non-Linear Bi-level Programming

Problem ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 5, No.11, September 2014.

[39] E. Hosseini, I.Nakhai Kamalabadi, Solving Linear Bi-level Programming Problem Using Two New Approaches Based on Line Search International Journal of Management sciences and Education, Vol. 2, Issue 6, 2014, 243-252.

[40] E. Hosseini, I.Nakhai Kamalabadi, Two Approaches for Solving Non-linear Bi-level Programming Problem, Advances in Research, 3(5), 2015.

Table 1 comparison optimal solutions in PSOMGA- Example 3

Best solution by PSOMGA	Best solution according to reference [17]	Optimal solution
(x^*, y^*)	(x^*, y^*)	(x^*, y^*)
(4.00,4.00)	(4.00,4.01)	(4.00,4.00)

Table 2 comparison optimal solutions in PSOMGA- Example 4

Best solution by PSOMGA	Best solution according to reference [17]	Optimal solution
(x^*, y^*)	(x^*, y^*)	(x^*, y^*)
(1.834,0.892,0.004)	(1.833,0.891,0.000)	(1.834,0.892,0.004)

Table 3-Comparison of optimal solutions with different examples 5-8 of BLPP by PSOMGA

	Best solution by our method	Best solution according to reference [3, 7, 26, 27]
Example 5	(1.44,1.33,0,0.24,1.4,0.88) $F^* = -52.14$	(1.32,1.28,0,0.33,1.25,0.92) $F^* = -51.31$
Example 6	(2,0,0,0)$F^* = 10$	(2,0,0,0)$F^* = 10$
Example 7	(0.05,0.95,0,0.6,0.4) $F^* = 29.8$	(0,0.9,0,0.6,0.4) $F^* = 29.2$
Example 8	(17.43,11) $F^* = 86.22$	(17.45,10.90) $F^* = 85.08$

Table 4 comparison optimal solutions Improvement by PSOMGA

	PSOMGA Algorithm				
	Gap of Optimal Solution	Improvement Rathar Than [3, 7,17,26,27,30]	Improvement Rathar Than [39]	Iterations	Time
Example 3	0	0.01%	0 %	3000	0.41 s
Example 4	0	0.003%	0.0001%	1000	0.53 s
Example 5	0	0.003%	0.001%	4300	1.52 s
Example 6	0	0 %	0%	1700	1.55 s
Example 7	0	0.2%	0%	4100	1.43 s
Example 8	0	0.001%	0.0004%	4700	2.37 s

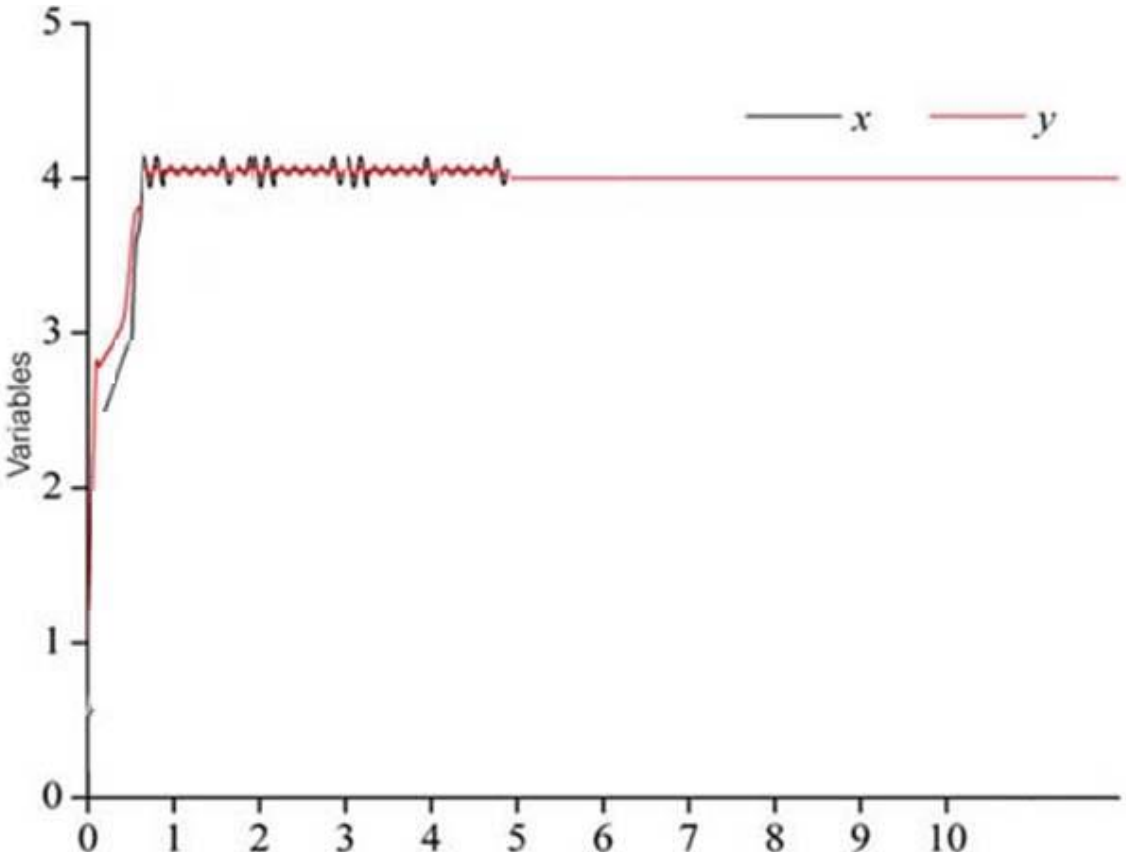


Figure 2 Behavior of the variables by PSOMGA in Example 3

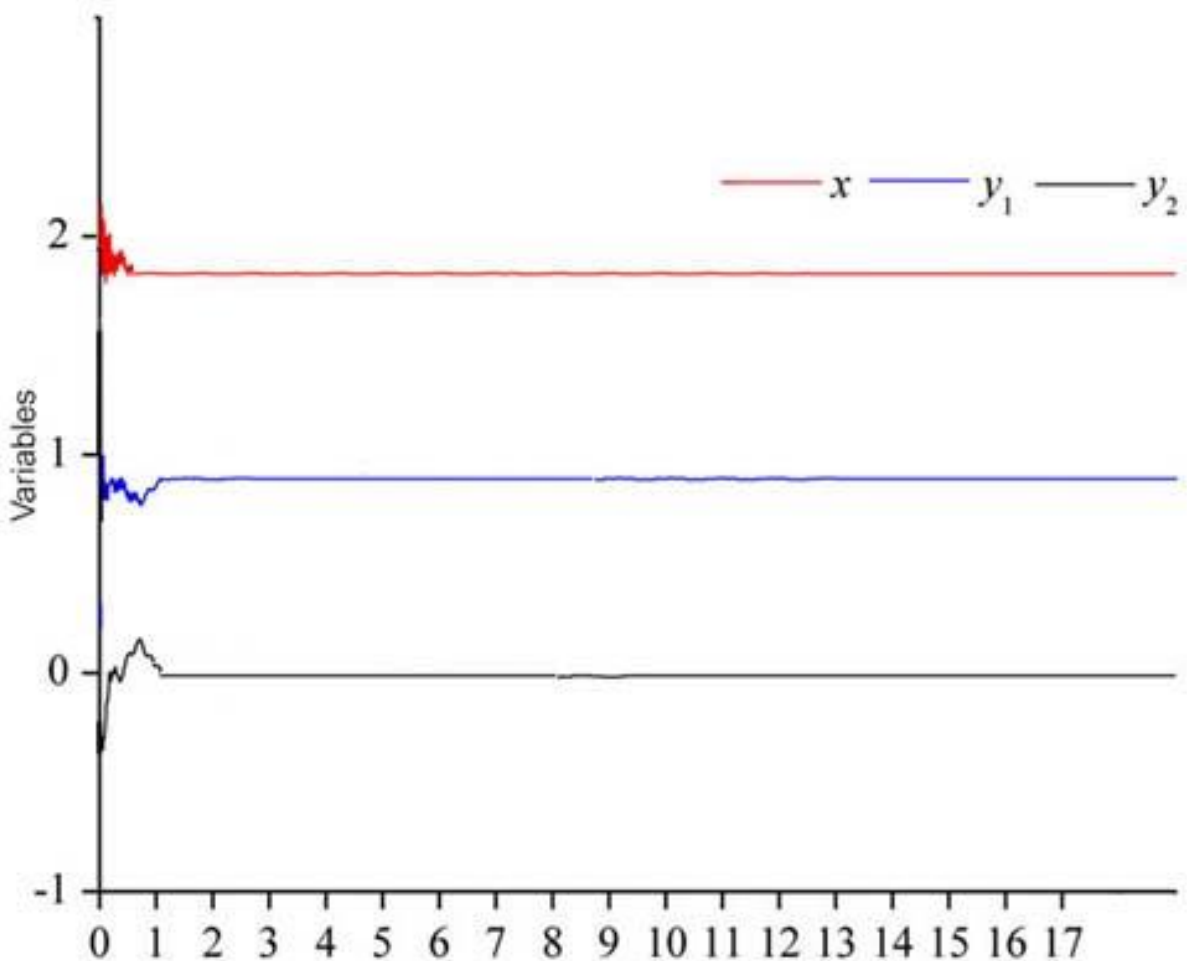


Figure 3 Behavior of the variables by PSOMGA in Example 4

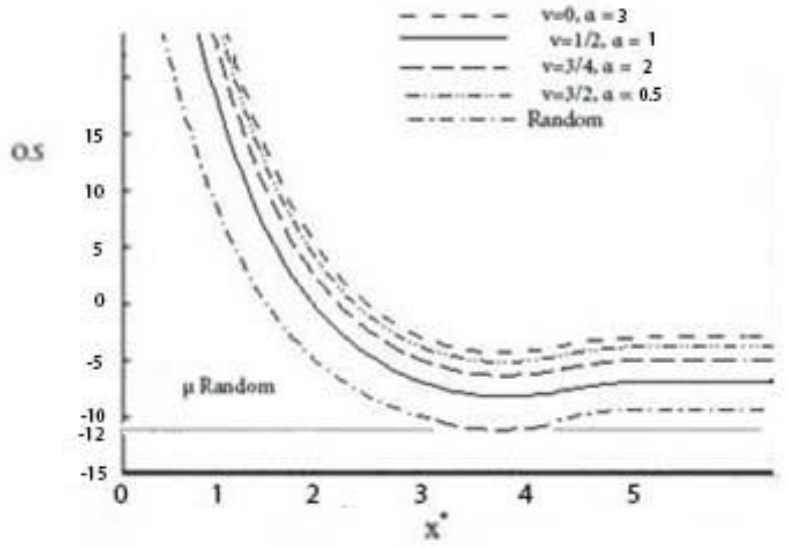


Figure 4 Behavior of the x and optimal solution (OS) with different value of α, μ, ν in Example 3

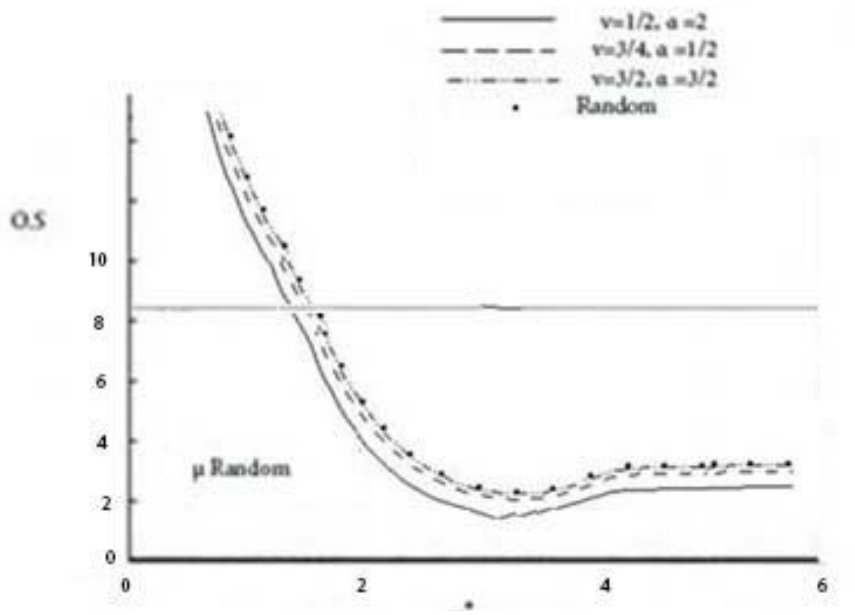


Figure 5 Behavior of the x and optimal solution (OS) with different value of α, μ, v in Example 4