A Self Adaptive Penalty Function Based Genetic Algorithm for value-Bilevel Programming Problem

Danping Wang
Tianjin University
College of Management and Economics
Tianjin, China
danping@tju.edu.cn

Gang Du
Tianjin University
College of Management and Economics
Tianjin, China

Abstract—This paper propose a self adaptive penalty function for solving constrained value-bilevel programming problem using genetic algorithm. This self adaptive penalty function based genetic algorithm both used in the higher level and the lower level problem's solving process. In the constraint handing method, a new fitness value called distance value, in the normalized fitness-constraint violation space, and two penalty values are applied to infeasible individuals so that the genetic algorithm would be able to identify the best infeasible individuals in the current population. The method aims to encourage infeasible individuals with low objective function value and low constraint violation. The number of feasible individuals in the population is used to guide the search process either toward finding more feasible solutions or toward finding the optimum solution. The proposed method is simple to implement and does not need parameter tuning. In this paper using self adaptive penalty function based Genetic Algorithm to solve the leader and the follower optimization module in the value-bilevel optimization problem. The performance of the algorithm is tested on an value-bilevel optimization problem. The results show that the approach is able to find very good solution.

***Keywords*-value-bilevel programming problem; genetic algorithm; constraint handing**

I. INTRODUCTION

Decision-making in most real life problems such as rolling system design can be made in hierarchical order and in most cases the search space is unknown. The bilevel programming problem (BLP) is a nested optimization problem with two levels in a hierarchy, the higher and lower level decision-makers,also termed as the leader and the follower.

The bilevel programming problem is a nested optimization problem characterized by the decision maker at one level influencing the behaviour of the decision maker at another level. Thus an important feature of the bilevel programming problem is that the objective functions of each unit may be partially determined by variables controlled by other units operating at other level. While the decisions made at lower level are not dictated by their superiors, their reactions have influence at the upper levels and have the tendency to improve their own objectives. The common characteristics of the bilevel programming are stated as follows:

• The decision-making units are interactive and exist within a predominantly hierarchical structure.

• Decision-making is sequential from leader to follower. The lower level decision-maker executes its policies after decisions are made at the higher level.

• Each unit independently optimizes its own objective functions but is influenced by actions taken by other units

The BLPs are generally difficult to solve due to the non-convex nature of the search space resulting from the complex interaction of the leader and follower problem. Genetic algorithms(GAs) are a stochastic search technique inspired by natural selection and natural genetics [1,2]. In the past decades, GAs have been extensively used for solving a wide variety of problems including BLPs, in which traditional approaches may not work adequately. Compared with traditional approaches, GAs have the following advantages.

(1) GAs do not require the objective function to be continuous or differentiable.

(2) GAs have good robustness for many applications.

(3) GAs have outstanding global search capabilities for convex and non -convex problems.

(4) GAs have inherent parallel processing capabilities.

(5) GAs are easy to implement.

However, on the other hand, GAs also have some limitations. GAs are essentially an unconstrained optimization technique. Although GAs perform well for unconstrained or simple constrained optimization problems, they may encounter some difficulties when applied to highly constrained problems. Generally, constrained optimization problems are difficult to solve. Due to the presence of constraints, the feasible space might be reduced to some portion of the total search space, and finding feasible solutions itself could be a daunting challenge.

One of the major issues of constrained optimization is how to deal with the infeasible individuals throughout the search process. One way to handle infeasible individuals is to completely disregard them and continue the search process with feasible individuals only. But this has a drawback because GAs are probabilistic search methods and some of the information contained in the infeasible individuals could be unutilized. If the search space is discontinuous, then the GA can also be trapped in local minimax. Therefore, different techniques have been developed to exploit the information contained in infeasible individuals.

The simplest and earliest method of involving infeasible individuals in the search process is the static penalty function method. In this method, some penalty values are added to the fitness value of each infeasible individual so that it will be penalized for violating the constraints. Static penalty functions are popular due to their simplicity but they usually require different parameters to be defined by the user to control the amount of penalty added when multiple constraints are violated. These parameters are usually problem dependent. To address this concerning issue, adaptive penalty functions are suggested recently where information gathered from the search process will be used to control the amount of penalty added to infeasible individuals. Adaptive penalty functions are easy to implement and they do not require users to define parameters.

This paper extends the single objective constrained optimization algorithm proposed by Tessema and Yen [3] to value-bilevel programming problem (BLP). The proposed algorithm basically modifies the objective function of an individual using its distance measure and penalty value. These modified objective function values are ranked through the non-dominance sorting of the multi-objective optimization. Distance measures are found for each dimension of the objective space by incorporating the effect of an individual’s constraint violation into its objective function. The penalty function, on the other hand, introduces additional penalty for infeasible individuals based on their objective values and constraint violations. The balance between two components, one based on objective function and the other on constraint violation, is controlled by the number of feasible individuals currently present in the population. If few feasible individuals are present, then those infeasible individuals with higher constraint violations are penalized more than those with lower constraint violations. On the other hand, if a sufficient number of feasible individuals exists, then those infeasible individuals with worse objective values are penalized more than those with better objective values. However, if the number of feasible individuals is in the middle of the two extremes, then the individual with lower constraint violation and better objective function is less penalized. The two components of the penalty function allow the algorithm to switch between feasibility and optimality at anytime during the evolutionary process. Furthermore, since priority is initially given to finding feasible individuals before searching for optimal solutions, the algorithm is capable of finding feasible solutions when the feasible space is very small compared to the search space.

This paper is structured as follows. Section II provides a brief overview of the genetic algorithms developed for BLPs. Then, in Section III, the proposed BLPs genetic algorithm is presented and analyzed in detail. Next, in Section IV, we discuss various BLPs test problem . Finally, we present the result of the experiment and conclude with a summary of this paper.

II. LITERATURE SURVEY

In bilevel programming problems, after the higher level decision maker --the leader give a control vector , the lower level decision maker--the follower make the decision , and then the follower take the lower level optimal\value to the leader , the leader change the decision x according the follower decision, finally through many times iteration the leader and the follower get the whole equilibrium optimal solutions. Usually the optimal solutions of the follower are not unique, however, the optimal value of the follower is unique. So in this paper, we just only concentrate on value-BLPs , in which the leader constraints just contain the upper level decision variable x, the model can be represented as the following expression:

$$\begin{aligned}
&\min_{X} F(X, v(X)) \\
&s.t.\ \ G_{i_1}(X) \le 0 \qquad i_1 = 1,\dots,k_1 \\
&\quad H_{i_1}(X) = 0 \qquad i_1 = k_1+1,\dots,m_1 \\
&\quad lb_x \le X \le ub_x \\
&\quad \mathrm{v}(X) = \min_{Y} f(X,Y) \\
&\quad s.t.\ \ g_{i_2}(X,Y) \le 0 \qquad i_2 = 1,\dots,k_2 \\
&\quad\quad h_{i_2}(X,Y) = 0 \qquad i_2 = k_2,\dots,m_2 \\
&\quad\quad lb_y \le \mathrm{Y} \le ub_y
\end{aligned}$$

where $X = \left(x_1, x_2, \dots, x_{n_1}\right), Y = \left(y_1, y_2, \dots, y_{n_2}\right)$.

Some important BLP definitions and notations are itemized below:

The relaxed feasible region (or constraint region) is

$$\Omega = \left\{(x,y) : G_{i_1}(x) \le 0, H_{i_1} = 0, g_{i_2}(x,y) \le 0, h_{i_2}(x,y) = 0\right\};$$

For a given (fixed) vector , the lower-level feasible set is defined by

$$\Omega(x) = \left\{y : g_{i_2}(x,y) \le 0, h_{i_2}(x,y) = 0\right\};$$

For a given (fixed) vector , the lower-level reaction set (or rational reaction set) is

$$M(x) = \left\{y : y \in \arg\min f(x,y) : y \in \Omega(x)\right\};$$

For a given (fixed) vector and any value of , the lower level optimal value is

$$v(x) = \arg\min f(x,y);$$

The induced region (or inducible region): $IR = \left\{(x,y) : (x,y) \in \Omega, y \in M(x)\right\}$.

Definition 1. If $(x,y) \in IR$,then (x,y) is the feasibility solution of BLP.

Definition 2. If $\left(x^*, y^*\right) \in IR$ and $F\left(x^*, y^*\right) \le F(x,y), \forall (x,y) \in IR$,then $\left(x^*, y^*\right)$ is the optimal solution of BLP.

This section presents a brief review of approaches developed for BLPs. Over the last decade, a number of various algorithms have been developed based on classical optimization approaches such as variable elimination method based on Kuhn Tucker approach [4-6] and algorithms based on penalty function approach [7-9]. Literature suggests that most of the traditional solutions are problem dependent relying on knowledge of the search space and are not sufficiently robust methods to solve real life problems [10]. Heuristic approaches are now generating interest in the research community as an alternative for solving the BLPs. Anadalingam et al [11] developed a Simulated Annealing Based Bi-level programming Algorithm (SABBA), Mathieu et al [10] developed a Genetic Algorithm Based Bi-level programming Algorithm (GABBA), Gendreau et al [12] proposed a hybrid Tabu Search ascent algorithm and recently Yin [13] proposed an approach (GAB) based on genetic algorithm. Most of the computational results of the heuristic approaches reported in the literature are still not as satisfying when compared to exact methods such as the Kuhn Tucker approach or the penalty method and also they are reported to be computationally expensive [14].

On the other hand, constraint handling for single objective optimization problems has also been actively researched over the past two decades [15]-[16]. Penalty functions are the simplest and most commonly used methods for handling constraints using EAs. In death penalty function methods such as [17], individuals that violate any one of the constraints are rejected and no information is extracted from infeasible individuals. If the added penalties do not depend on the current generation number and remain constant during the entire evolutionary process, then the penalty function is called static penalty function. In static penalty function methods [18], the penalties are the weight sum of the constraint violations. If, alternatively, the current generation number is considered in determining the penalties, then the method is called dynamic penalty function method [19]. In adaptive penalty function methods [20],-[21], information gathered from the search process will be used to control the amount of penalty added to infeasible individuals.

In [19] and [22], methods based on preference of feasible solutions over infeasible solutions are employed. In these types of techniques, feasible solutions are always considered better than infeasible ones. Therefore, when population fitness ranking is performed, feasible individuals will come first followed by infeasible individuals with low constraint violation. In [13] and [24], Runarsson and Yao introduce the stochastic ranking method to achieve a balance between objective and penalty functions stochastically. A probability factor is used to determine whether the objective function value or the constraint violation value determines the rank of each individual. In [25] and [26], similar algorithms are proposed where constraint violation and objective function are optimized separately.

More recently, multi-objective optimization techniques have been used to solve constrained optimization problems. In [27], a multi-objective optimization technique that uses population based algorithm generator and infeasible solutions archiving and replacement mechanism is introduced. In [28], a two-phase algorithm that is based on multi-objective optimization technique is proposed. In the first phase of the algorithm, the objective function is completely disregarded and the constraint optimization problem is treated as a constraint satisfaction problem. In the second phase, both constraint satisfaction and objective optimization are treated as a bi-objective optimization problem. An algorithm that combines penalty function approach and multiobjective optimization technique is also suggested in [29]. The algorithm has a similar structure as the penalty-based approach but borrows the ranking scheme from multi-objective optimization techniques.

Although genetic algorithm and constraint handling have received a lot of attention individually, very little effort has been devoted in solving constrainted bilevel optimization problems. In light of the superior performance achieved in [3] for the single objective constraint optimization, a similar idea is extended in this paper into the use of constrainted value-bilevel optimization problems.

When dealing with constrained value-bilevel optimization problems, in the lower level optimization problem, the the lower level individuals that satisfy all of the lower level constraints are called feasible individuals while individuals that do not satisfy at least one of the lower level constraints are called infeasible individuals; in the upper level optimization problem, the individuals that satisfy all of the constraints are called feasible individuals while individuals that do not satisfy at least one of the constraints are called infeasible individuals.

III. PROPOSED ALGORITHM

The proposed algorithm extends the single-objective constrained evolutionary algorithm proposed by Tessema and Yen [3] into value-bilevel programming framework. The major difference in various constraint handling techniques used in value-bilevel optimization problems arises from the variations in the involvement of two group infeasible individuals (the lower level and the whole) in the evolutionary process . The main purpose of involving infeasible individuals in the whole search process is to exploit the information they carry. Since GA is stochastic search technique, discarding infeasible individuals might lead to the GA being stuck in local optima, especially in problems with disjoint search space. In addition, in some highly constrained optimization problems, finding a single feasible individual by itself might be a daunting challenge when the algorithm must be able to extract information from the previous infeasible individuals.

The proposed algorithm use modified objective function values for checking dominance in the population. The modification is based on the constraint violation of the individual and its objective performance. The modified objective value has two components: distance measure and adaptive penalty. After that, using GA solves two optimization problems iteratively, one for the leader in all the x variables and a subset of the y variables associated with the optimal basis follower's problem, and the other for the follower problem with all the x variables fixed. The optimal basis of the follower problem is explored with x fixed and then returned to the leader problem with the corresponding v(x) variables. The algorithm uses the full operators in the basic genetic algorithm in a dual population environment to solve both leader and the follower problem, and uses the modified objective value to get the fitness value. The modified objective value's two components are discussed next in detail.

A. The Follower

For the follower, when the leader give a (fixed) vector x, there is a group population of y. When dealing with the follower optimization problem, individuals (x,y) that satisfy all of the follower constraints are called feasible individuals while individuals that do not satisfy at least one of the follower constraints are called infeasible individuals.

A1. Follower Distance Values

Follower distance measure is found for the follower's objective space by including the effect of a follower individual' s constraint violation into its objective function. The major steps in calculating the follower distance measure starts with obtaining the minimum and maximum values of each follower objective function in the population of the follower

$$f_{\min} = \min_{y} f(x,y)$$

and

$$f_{\max} = \max_{y} f(x,y).$$

Using these values, normalize the follower objective function for every follower individual (x,y),

$$\tilde{f}(x,y) = \frac{f(x,y) - f_{\min}}{f_{\max} - f_{\min}}$$

where $\widetilde{f(x,y)}$ is the normalized follower objective value of individual (x,y).

Follower constraint violation $v_f(x,y)$ of individual (x,y) is then calculated as the summation of the follower normalized violations of each follower constraint divided by the total number of follower constraints:

$$v_f(x,y) = \frac{1}{m_2} \sum_{j=1}^{m_2} \frac{c(x,y)}{c_{\max}}$$

where

$$c(x,y) = \begin{cases} \max\left(0, g_j(x,y)\right) & j = 1,\ldots,k_2 \\ \left|h_j(x,y)\right| & j = k_2+1,\ldots,m_2 \end{cases}$$

.

Then the “follower distance” value of individual in the follower objective function is formulated as follows:

$$d_f(x,y) = \begin{cases} v(x,y) & r_f = 0 \\ \sqrt{\widetilde{f(x,y)}^2 + v(x,y)^2} & otherwise \end{cases}$$

where $r_f = \dfrac{\text{number of feasible individual in current population } (x,y)}{\text{population size}}$.

we observe that if there is no feasible individual in the follower current population , then the follower distance values are equal to the follower constraint violation of the individual (x,y). In this case, according to the follower distance values, an infeasible follower individual with smaller follower constraint violation will dominate another infeasible follower individual with higher follower constraint violation regardless of their follower objective function values. This is the best way to compare infeasible follower individuals in the absence of feasible follower individuals [18], [21] since it gives priority to finding feasible follower individuals over finding follower optimal solutions. On the other hand, if there is more than one feasible solution in the follower population, then the follower distance values will have the properties summarized below.

1) For a follower feasible individual , the distance value in the follower is equal to $\widetilde{f(x,y)}$. Hence, those follower feasible individuals with smaller follower objective function values will have smaller follower distance values.

2) For follower infeasible individuals, the follower distance value has two components: the follower objective function value and the follower constraint violation. Hence, follower individuals closer to the origin in the follower space would have lower follower distance value than those farther away from the origin.

3) If we compare the follower distance values of follower infeasible and feasible individuals, then either one may have a smaller value. However, if the two follower individuals have similar follower objective function values, then the follower feasible individual will have a smaller follower distance value .

A2. Follower two Penalties

In addition to the follower distance measure, two penalty functions are added to the follower fitness value of infeasible individuals in the follower. These follower functions penalize infeasible individuals based on their corresponding follower objective value and follower constraint violation. The first follower penalty function is based on the follower objective functions, and the second is based on the follower constraint violation. The balance between the two components is controlled by the number of follower feasible individuals currently present in the population.

These follower penalties have two major purposes:

1) To further reduce the fitness of follower infeasible individuals as the penalty imposed by the distance formulation alone is small.

2) To identify the best follower infeasible individuals in the population by adding different amount of penalty to each follower infeasible individual’s fitness.

The two penalties are formulated for follower individual y in the follower as follows:

$$p_f(x,y) = (1-r_f)X_f(x,y) + Y_f(x,y)$$

where

$$X_f = \begin{cases} 0 & r_f = 0 \\ v_f(x,y) & otherwise \end{cases}$$

and

$$Y_f = \begin{cases} 0 & \text{if (x,y) is a feasible individual in the follower} \\ \widetilde{f(x,y)} & \text{if (x,y) is an infeasible individual in the follower} \end{cases}.$$

The two components of the follower penalty function allow the algorithm to switch between finding more follower feasible solutions and finding better follower solutions at anytime during the evolutionary process. Furthermore, because priority is initially given to the search for follower feasible individuals, the algorithm is capable of finding follower feasible solutions in cases where the follower feasible space is small or disjoint compared to the search space.

A3. Final Modified Follower Objective Value Formulation

The final modified follower objective value of follower individual (x,y), using which non-dominance sorting is performed, is formulated as the sum of the follower distance measure and follower penalty function:

$$F_f(x,y) = d_f(x,y) + p_f(x,y).$$

This modified follower objective value formulation is flexible and will allow us to utilize follower infeasible individuals efficiently and effectively. Most constraint optimization algorithms in literature are “rigid” in a sense that they always prefer certain types of infeasible individuals throughout the entire evolutionary process. For example, they might always give priority to those individuals with small constraint violation only or those individuals with low objective value only. According to our new follower fitness formulation, the follower infeasible individuals that are considered valuable are not always similar. Here are some of the interesting properties of this modified follower objective value formulation.

1) If there is no follower feasible individual in the follower current population, each $d_f(x,y)$ will be equal to the follower constraint violation $v_f(x,y)$, and each $p_f(x,y)$ term will be zero. In this case, the objective values of the follower individuals will be totally disregarded, and all individuals will be compared based on their constraint violation only. This will help us find follower feasible individuals before looking for follower optimal solutions.

2) If there are follower feasible individuals in the follower population, then follower individuals with both low follower objective function values and low follower constraint violation values will dominate individuals with high follower objective function values or high follower constraint violation or both.

3) If two follower individuals have equal or very close follower distance values, then the follower penalty term $\left(p_f\left(x,y\right)\right)$ determines the dominant individual. According to our penalty formulation, if the feasibility ratio (r_f) in the follower population is small, then the follower individual closer to the follower feasible space will be dominant. On the other hand, the follower individual with smaller follower objective function values will be dominant. Otherwise, the two follower individuals will be nondominant solutions.

4) If there is no follower infeasible individual in the follower population ($r_f = 1$), then individuals will be compared based on their follower objective function values alone.

Through above constraints handling method used in GAs to solved the follower optimization problem, can get the approximate optimal value of the follower.

B. The Leader

When dealing with the leader optimization problems, individuals that satisfy all of the leader constraints are called feasible individuals while individuals x that do not satisfy at least one of the leader constraints are called infeasible individuals.

The leader distance value and two leader penalty functions are similarity to the follower. The details are as follow. The leader distance value of individual in the leader objective function is formulated as follows:

$$d_F\left(x,v(x)\right)=\begin{cases} v_F\left(x,v(x)\right) & r_F=0 \\ \sqrt{\widetilde{F}\left(x,v(x)\right)^2+v_F\left(x,v(x)\right)^2} & otherwise \end{cases}$$

where

$$r_F=\frac{\text{number of feasible individual in current population x}}{\text{population size}},$$

$$v_F\left(x,v(x)\right)=\frac{1}{m_1}\sum_{j=1}^{m_1}\frac{C\left(x,v(x)\right)}{C_{\max}},$$

$$C\left(x,v(x)\right)=\begin{cases} \max\left(0,G_j(x)\right) & j=1,\ldots,k_1 \\ \left|H_j(x)\right| & j=k_1+1,\ldots,m_1 \end{cases},$$

$$C_{\max}=\max_x C\left(x,v(x)\right),$$

$$\tilde{F}\left(x,v(x)\right)=\frac{F\left(x,v(x)\right)-F_{\min}}{F_{\max}-F_{\min}},$$

$$F_{\min}=\min_x F\left(x,v(x)\right)$$

and

$$F_{\max}=\max_x F\left(x,v(x)\right).$$

The two leader penalties are formulated for leader individual x in the leader as follows:

$$p_F\left(x,v(x)\right)=\left(1-r_F\right)X_F\left(x,v(x)\right)+Y_F\left(x,v(x)\right)$$

where

$$X_F = \begin{cases} 0 & r_F = 0 \\ v_F\left(x, v(x)\right) & otherwise \end{cases}$$

and

$$Y_F = \begin{cases} 0 & \text{if x is a feasible individual in the leader} \\ \widetilde{F\left(x, v(x)\right)} & \text{if x is an infeasible individual in the leader} \end{cases}.$$

The final modified leader objective value of individual , using which nondominance sorting is performed, is formulated as the sum of the leader distance measure and leader penalty function :

$$F_F\left(x, v(x)\right) = d_F\left(x, v(x)\right) + p_F\left(x, v(x)\right).$$

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. The GA Procedure for BLP

Genetic algorithm (GA) is a stochastic search and optimization algorithm that mimics biological evolution. The idea behind GA is to use this power of evolution to solve optimization problems. GA works on the composition of genetic traits called chromosomes, in which successive operations through crossover or mutation give rise to better performing off-springs due to successive refinement of these hereditary traits. GA works with a population of design solutions and tries to find the best solution. A design solution, composed of the design variables, is represented as a single chromosome.

Using Genetic algorithm to directly solve the bilevel programming equal to solving two optimization problems iteratively,one for the leader in all the x variables and a subset of the y variables associated with the optimal basis follower’ s problem, and the other for the follower problem with all the x variables fixed. The solution strategy is encoded in Matlab using Genetic algorithm shown in figure 1.

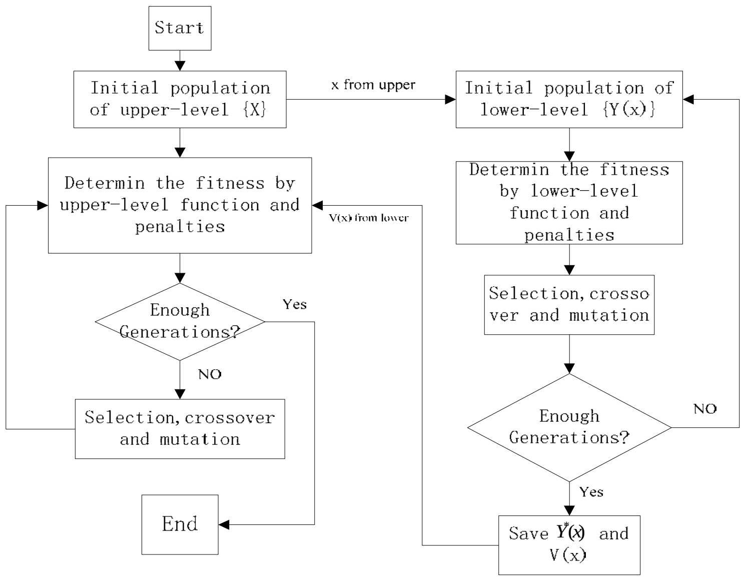
Start
Initial population of upper-level {X}
x from upper
Initial population of lower-level {Y(x)}
Determin the fitness by upper-level function and penalties
V(x) from lower
Determin the fitness by lower-level function and penalties
Selection, crossover and mutation
Enough Generations?
Yes
NO
Selection, crossover and mutation
Enough Generations?
NO
Yes
End
Save Y(x) and V(x)

Figure 1.Genetic algorithm process flow

After the initialization of the leader population (PL), the follower members represented in the leader problem are copied from the follower population (PF) into the leader population. First given a fixed ,using GA to get the optimal value of the follower problem;then solving the leader problem through using optimal value of the follower problem. Selection, crossover and mutation operations are similar to conventional simple GA technique. Through used the self adaptive penalty function based Genetic Algorithm in the follower, got the approximate optimal value of the follower. Then took the follower approximate optimal value back to the leader optimization problem, using the self adaptive penalty function based GA to get the optimal value of the whole problem--BLP, at the same time,Selection, crossover and mutation operations are similar to conventional simple GA technique.

The self adaptive penalty function based genetic algorithm will terminate after a given number of cyclic repetitions of the above steps.We can summarize the genetic algorithm for solving the value-bilevel programming models as follows.

```
Step 0. Input parameters population size, crossover probability and mutation probability.
Step 1. Initialize population size chromosomes x for the leader.
Step2. Take the chromosomes to the follower programming model:
    Step2.1 Initialize population size chromosomes y for the follower corresponding one chromosome x .
    Step2.2 Calculate the follower fitness objective values of the follower for all follower chromosomes
            according to the follower self adaptive  penalty function.
    Step2.3 Select the follower chromosomes y by spinning the roulette wheel.
    Step2.4 Update the follower chromosomes y by crossover and mutation operations.
    Step2.5 Repeat the 2.2  to 2.4 steps a given number of cycles.
    Step2.6 Report the best chromosome as the optimal solution,and get  the optimal value of the follower.
Step3. Calculate the leader fitness objective values of all leader chromosomes according to the leader self
       adaptive penalty function.
Step4.Select the leader chromosomes by spinning the roulette wheel.
Step5.Update the leader chromosomes y by crossover and mutation operations.
Step6. Repeat the 2 to 5 steps a given number of cycles.
Step7. Report the best chromosome x as the optimal solution.
```

B. Experimental Setup

The proposed genetic algorithm is tested on an value-bilevel programming problems (BLPs) . The simulations are conducted with a population size of 100, crossover rate of 0.8, mutation rate of 0.2, and maximum generation number of 200 for all implementations.

In recent reach. although the value-bilevel programming model used in the engineering models,there are little algorithm reach on the numerical value-bilevel programming model, so we just find one existing simplest examples of bilevel programming to test the proposed genetic algorithm. Assume that in a value-bilevel programming model there exist one leader and one follower. Suppose that the leader has a control vector $x=\left(x_1, x_2\right)$ and the follower has a control vector $y=\left(y_1, y_2\right)$.The bilevel programming is formulated as follows,

$$\begin{aligned}
&\max F\left(x, y\right)=\frac{\left(x_1+y_1\right)\left(x_2+y_2\right)}{1+x_1 y_1+x_2 y_2} \\
&s.t \quad x_1^2+x_2^2 \leq 100 \\
&\qquad 0 \leq x_1 \leq 10 \\
&\qquad 0 \leq x_2 \leq 10 \\
&\qquad \max f\left(x, y\right)=-F(x, y) \\
&\qquad s.t\, 0 \leq y_1 \leq 10 \\
&\qquad\quad 0 \leq y_2 \leq 10
\end{aligned}$$

which is equivalent to the following problem:

$$
\begin{aligned}
&\min F\left(x,v\left(x\right)\right)=-v\left(x\right)\\
&s.t \quad x_1^2+x_2^2\le 100\\
&\qquad 0\le x_1\le 10\\
&\qquad 0\le x_2\le 10\\
&\qquad v\left(x\right)=\min\ f(x,y)=-\frac{\left(x_1+y_1\right)\left(x_2+y_2\right)}{1+x_1y_1+x_2y_2}\\
&\qquad s.t\, 0\le y_1\le 10\\
&\qquad\quad 0\le y_2\le 10
\end{aligned}
$$

This is one of the simplest examples of bilevel programming. A run of genetic algorithm with 200 generations shoes that the solution is x^*= (9.5894, 9.5503), y^*= (0.4526, 9.5513), F^*=1.9866.

From the figure 2, we can see after 20 iteratives the best individual function value of each generation has no change, this is to say,in very small iterative the algorithm find the best solution of the bilevel programming. The average fitness value of each generation show that the algorithm have good convergence.

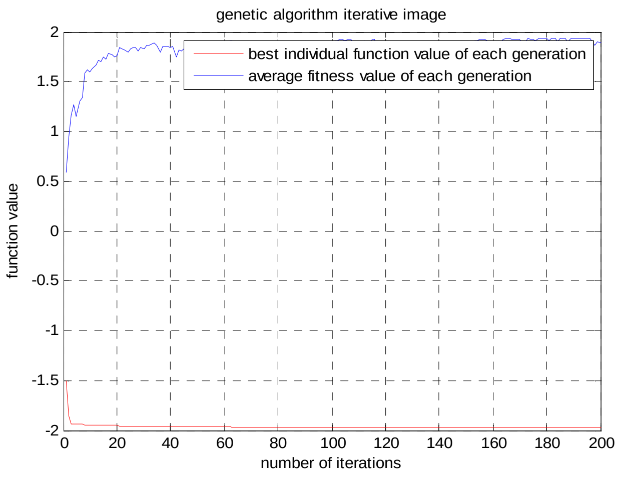

Figure 2. Genetic algorithm interation result

The test performed with the GA lead to the following conclusions.

• The proposed algorithm is able to find solutions close to the optimal region similar to traditional approaches and it can deal with non-differentiable complex the BLP problems.

• The GA is robust to address different classes of the BLP problem and can be used to solve problems independent of the nature of the search space. This is an important requirement for solving real-life problems where the nature search space is not always known.

V. CONCLUSION AND DISCUSSIONS

In this paper, we propose an adaptive constraint handling technique in GA for solving constraint value-BLPs. Besides the search for optimal solutions in the feasible region, the genetic algorithm also exploits the information hidden in infeasible individuals with better objectives and lower constraint violation. The modified objective values in the leader and the follower are composed of distance measures and penalty functions.These values are associated with how well an individual performs and how much it violates the constraints in module. The number of feasible individuals in the population adaptively controls the emphasis given to objective values or constraint violation in the modified objective function formulation. If there is no feasible individual in the population, the genetic algorithm uses the constraint violations as the primary means to rank the individuals. Involving infeasible individuals in the evolutionary process helps the genetic algorithm to find additional feasible individuals, even in cases where the leader and the follower feasible space is very small and disjoint. Furthermore, since there is no parameter tuning, this makes the genetic algorithm easy to implement. Moreover, the additional evaluations are simple arithmetic operations and do not impose any significant increase in the computational cost.

REFERENCES

[1] Goldberg DE.Genetic algorithms in search,optimization,andmachine learning. AddisonWesley;1989.

[2] ThomasB.Evolutionary algorithms in the oryand practice:evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press;1996.

[3] B. Tessema and G. G. Yen, “A self-adaptive constrained evolutionary algorithm,” in Proc. IEEE Cong. Evol. Comput., Vancouver, Canada, 2006, pp. 246–253

[4] Hansen, P., B. Jaumard, and G. Savard, New Branch-and- Bound Rules for Linear Bilevel Programming. SIAM Journal of Scientific and Statistical Computing, 1992. 13(5): pp. 1194-1217.

[5] Judice, J.J. and A.M. Faustino, A Sequential LCP Method for Bilevel Linear Programming. Annals of Operations Research, 1992. 34(1-4): p. 89-106.

[6] Bard, J.F. and J.T. Moore, A Branch and Bound Algorithm for the Bilevel Programming Problem. SIAM Journal of Scientific and Statistical Computing, 1990. 11(2): pp. 281- 292.

[7] Wei, Z., Penalty function method for bi-level multiobjective programming. Acta Automatica Sinica, 1998. 24(3): p. 331- 337.

[8] Shimizu, K. and M. Lu, A Global Optimization Method fro the Stakelberg Problem with Convex Functions via Problem Transformations and Concave Programming. IEEE Transactions System, Man and Cybernatics, 1995. SMC- 25(12): pp. 1635-1640.

[9] White, D.J. and G. Anandalingam, A Penalty Function for Solving Bi-Level Linear Programs. Journal of Global Optimization, 1993. 3: pp. 397-419.

[10] Mathieu, R., L. Pittard, and G. Anandalingam, Genetic Algorithm Based Approach to Bi-level Linear Programming. Operations Research, 1994. 28(1): pp. 1-21.

[11] Anandalingam, G., et al., Artificial Intelligence Based Approaches for Hierarchical Optimization, in Impact of Recent Computer Advances in Operations Research, R. Sharda, Editor, North-Holland: New York,1989.

[12] Gendreau, M., P. Marcotte, and G. Savard, A Hybrid Tabu- Ascent Algorithm for the Linear Bilevel Programming Problem. Journal of Global Optimization, 1996. 8(3): pp. 217- 233.

[13] Yin, Y., Genetic Algorithm based approach for bilevel programming models. Journal of Transportation Engineering, 2000. 126(2): pp. 115-120.

[14] Bard, J., Practical Bilevel Optimization Algorithm and Applications. Nonconvex Optmization and Its Applications. Vol. 30. Kluwer Academics, 1998.

[15] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, “An adaptive tradeoff model for constrained evolutionary optimization,” IEEE Trans. Evol. Comput., vol. 12, pp. 80–92, 2008.

[16] R. C. Purshouse and P. J. Fleming, “On the evolutionary optimization of many conflicting objectives,” IEEE Trans. Evol. Comput., vol. 11, pp. 770–784, 2007.

[17] T. Bäck, F. Hoffmeister, and H. Schwefel, “A survey of evolution strategies,” in Proc. Int. Conf. Genetic Algorithms, San Diego, CA, 1991, pp. 2–9.

[18] A. Homaifar, S. H. Y. Lai, and X. Qi, “Constrained optimization via genetic algorithms,” Simulation, vol. 62, no. 4, pp. 242–254, 1994.

[19] J. Joines and C. Houck, “On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs,” in Proc. Congr. Evol. Comput., Orlando, FL, 1994, pp. 579–584.

[20] J. C. Bean and A. B. Alouane, A dual genetic algorithm for bounded integer programs Univ. Michigan, Dept. Industrial and Operations Engineering, Ann Arbor, MI, Tech. Rep. TR 92-5, 1992, .

[21] R. Farmani and J. Wright, “Self-adaptive fitness formulation for constrained optimization,” IEEE Trans. Evol. Comput., vol. 7, pp. 445–455, 2003.

[22] K. Deb, “An efficient constraint handling methods for genetic algorithms,” Comput. Methods in Appl. Mech. Eng., vol. 186, pp. 311–338, 2000.

[23] T. P. Runarsson and X. Yao, “Stochastic ranking for constraint evolutionary optimization,” IEEE Trans. Evol. Comput., vol. 4, pp. 284–294, 2000.

[24] T. P. Runarsson and X. Yao, “Search bias in constrained evolutionary optimization,” IEEE Trans. Syst., Man, Cybern., vol. 35, pt. C, pp. 233–243, 2005.

[25] T. Takahama and S. Sakai, “Constrained optimization by applying the -constrained method to the nonlinear simplex method with mutations,” IEEE Trans. Evol. Comput., vol. 9, pp. 437–451, 2005.

[26] T. Takahama and S. Sakai, “Constrained optimization by the ┐ constrained differential evolution with gradient-based mutation and feasible elite,” in Proc. IEEE Cong. Evol. Comput., Vancouver, Canada, 2006, pp. 308–315.

[27] Y. Wang and Z. Cai, “A multi-objective optimization based evolutionary algorithm for constrained optimization,” in Proc. IEEE Cong. Evol. Comput., Edinburgh, U.K., 2005, pp. 1081–1087.

[28] S. Venkatraman and G. G. Yen, “A generic framework for constrained optimization using genetic algorithms,” IEEE Trans. Evol. Comput., vol. 9, pp. 424–435, 2005.

[29] J. Aidanpaa, J. Anderson, and A. Angantyr, “Constrained optimization based on a multi-objective evolutionary algorithm,” in Proc. Cong. Evol. Comput., Canberra, Australia, 2003, pp. 1560–1567.