INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Layered Approach Based on Objectives to Multi-Objective Optimization

Lian-Shuan Shi[1,2*], **Yin-Mei Chen**[1]

[1] *Tianjin University of Technology and Education, Tianjin 300222, China*
[2] *State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116023, China*
[*] *Corresponding author's Email: shilianshuan@sina.com*

Abstract: A new multi-objective optimization approach based on layered objective is proposed. Two improvements on the non-dominated sorting genetic algorithm are given which are new layering method and the new selection mechanism. On the basis of the traditional non-dominated sorting genetic algorithm, a new objective layer approach is adapted and cycle copy through the first half layers. Another one is making a restriction mechanism before crossover on the classical non-dominated sorting genetic algorithm with elitist strategy. Through these operations, it can effectively improve the convergence rate of the optimal solution set. At last, the corresponding non-inferior solutions will be maintained. Several classic test functions commonly adopted in evolutionary multi-objective optimization and the network design problem are used. The results indicate that the non-dominated sorting genetic algorithm based on layered objective can effectively speed up the convergence rate.

Keywords: Genetic algorithm; Layered objective; Multi-objective programming; Network optimization.

1. Introduction

The multi-objective optimization problems widely exist in real engineering world. And such problem is usually characterized by the presence of many conflicting objectives. Because of the variety internal relations, these objectives are both interdependent and competitive. As a result, there is usually not a single solution which optimizes all objectives simultaneously and a number of techniques have been developed in order to find a "good" solution to the MOP.

For multi-objective optimization problems, there are many methods available to tackle these kinds of problems. One way to solve multi-objective problems is to transform the original problem into a single-objective one by using the weighted sum method with a weight vector. When the multi-objective optimization problem is converted to a single objective optimization problem, a few mature algorithms can be used to solve the single objective optimization problem whose solutions are taken as one of the multi-objective optimization problems. But the obtained solution depends on the weight vector used in the weighting process, so some improving multi-objective optimization approaches weighted are given [1, 2, 3].

Another way is to find the Pareto efficient solutions to a multi-objective optimization problem. The Pareto optimality was introduced in the late 1800's by the economist Vilfredo Pareto, and defined as follows: A solution is said to be Pareto optimal if there exists no other solution that is better in all attributes. This implies that in order to achieve a better value in one objective at least one of the other objectives is going to deteriorate if the solution is Pareto optimal. Thus, the outcome of a Pareto optimization is not one optimal point, but a set of Pareto optimal solutions that visualize the trade-off between the objectives [4].

Goldberg [5] introduced non-dominated sorting to

rank a search population according to Pareto optimality. Firstly, the non-dominated individuals in the population are identified. They are given the rank 1 and are removed from the population. Then the non-dominated individuals in the reduced population are identified, given the rank 2, and then they are also removed from the population. This procedure of identifying non-dominated sets of individuals is repeated until the whole population has been ranked.

The main purpose of the non-dominated sorting genetic algorithm (NAGA), which is developed from the basic genetic algorithm, is to solve the multi-objective optimization problem. The non-dominated sorting genetic algorithm [6] was proposed by Srinivas and Deb in 1994 which is also known as non-poor classification genetic algorithm. This algorithm is the same as SGA algorithm except on the selective operator. The main idea of this algorithm is to classify the populations according to the idea of non-dominated sorting firstly, then, each individual that have been classified will be given a certain shared virtual fitness in order to maintain the diversity of the population. Non-dominated individuals in the population are identified, given a high initial individual score and are then removed from the population. These individuals are considered to be of the same rank. The score is then reduced using sharing techniques between individuals with the same ranking. Thereafter, the non-dominated individuals in the remaining population are identified and scored lower than the lowest one of the previously ranked individuals. At last, next generation is produced by some selection operation with a certain probability. After that, a number of genetic algorithms is given, such as: NPGA [7] (Nich-Pareto)FFGA [8] (Fonseca and Fleming GA), SPEA [9] (Stregth Pareto GA), NSGA- [10] (A fast elitist non-dominated Pareto GA).

Although the non-dominated sorting genetic algorithm has been well used in many engineering problems, some problems still exist:

(1) Higher computation time complexity [10]. Its computation complexity is O(mN3), where, m represents the number of objective function; N represents the number of the population size. When the population size is large, the computing time cost will be great, especially to sort the populations of every generation, the time cost will be much bigger.

(2) Lack of elite strategy. The elite strategy guarantees that the optimal solution has been found under a certain extent cannot be damaged by crossover and mutation, but also can accelerate the convergence speed.

(3) That the sharing radius needs to be specified makes the calculation more complex.

Although the distribution of Pareto optimal solution set which is got from the NSGA is fairly uniform, the convergence speed is not fast because of the above three drawbacks. Aimed at the high computation time complexitywhen carrying out the algorithm, this article proposes a new hierarchical approach based on objective value to reduce the time complexity, so as to improve its convergence rate of the Pareto optimal solution. Additionally, in some algorithm, the selection operation is based on shared virtual fitness value under a certain probability, so some good individuals can not be selected, which results in the slower convergence.

In response to this phenomenon, this paper introduces a new selective mechanism, changing the defects resulted from the selective with probability in the past, so that the convergence rate of the Pareto optimal solution will be faster.

Aimed at the problem of slow convergence of optimal solutions in genetic algorithms, two improved genetic algorithms are proposed. The first one is based on the objective layer of the non-dominated sorting genetic algorithm. On the basis of the traditional non-dominated sorting genetic algorithm, it has two contributions. The first one is the adaptation of a new objective layer approach and the second one is the new selection mechanism. Its main idea is cycle copy through the first 1/2 layer according to the hierarchy. Another one is based on the cross-limited with elitist strategy of the non-dominated sorting genetic algorithm. It makes a restriction mechanism before crossover on the classical non-dominated sorting genetic algorithm with elitist strategy. Through this operation, it can effectively improve the convergence rate of the optimal solution set. Meanwhile, the algorithm also combines the objective layer approach to identify the non-inferior solution. The two multi-objective optimization functions are used to do the test. The results show that these improved algorithms speed up the convergence rate of the optimal solution. Last, the improved algorithm is used to deal with the multi-objective network optimization problem. The mathematical model includes two objectives: the path length and the path delay. In practice, each different path has different node numbers in the network, so the variant length of node index is used to code each path. Through the application of the improved algorithms, the Pareto optimal solution sets of network optimiza-

tion are obtained. All of the programs are achieved by C++ language programming and the simulation results are achieved with Matlab software.

2. Related Concepts of The Multi-objective Optimization

In most cases, the multi-objective optimization problem is defined as the combinatorial optimization problem under constrained conditions. It is mainly used to maximize (or minimize) the objective function of different objective.

A multi-objective optimization problem can be described as follows: the set of decision variables with the number of n, the set of the objective function with the number of k and the set of constraint functions with the number of m. A constrained multi-objective optimization problem (MOP) can be given as follows. Here, the optimization problem associated with each objective will be considered to be a minimization one, without loss of generality.

Minimize $y = f(x) = (f_1(x),\ f_2(x),\ldots,f_k(x))^T$

subject to $e(x) = (e_1(x),\ e_2(x),\ldots,e_m(x))^T \leq 0$

$x = (x_1,\ x_2,\ldots,x_n)^T \in \Omega \subset R^n$

$y = (y_1,\ y_2,\ldots,y_n)^T \in Y$

where, $f_1(x),\ f_2(x),\ldots,f_k(x)$ are k objective functions, $x_1,\ x_2,\ldots,x_n$ are n optimization parameters, x represents the vector of decision variables, y represents the vector of objective functions, Ω represents the decision variable space and Y represents the objective function space, the constraint $e(x) \leq 0$ defines the set of feasible solutions.

In order to better understand the decision variable space and the objective function space of the multi-objective optimization problem, some definitions are given as follows:

Definition 1

For any two objective vector b and c,
$b = (b_1,\ b_2,\ldots,b_n)^T$, and $c = (c_1,\ c_2,\ldots,c_n)^T$,
$b = c$, If and only if $b_i = c_i$ for all $i \in \{1,2,\ldots,k\}$,
$b \leq c$, If and only if $b_i \leq c_i$ for all $i \in \{1,2,\ldots,k\}$,
$b < c$, If and only if $b \leq c$ for all $b \neq c$. In the above formula, b_i represents an element of the objective vector b and c_i represents an element of the objective vector c.

Definition 2

For any two objective vector u and v, the binary relation $\prec$, $\preceq$ and $\sim$ define as follows:
$u \prec v$ (u dominates v), If and only if $f(u) < f(v)$.
$u \preceq v$ (u weakly dominates v), If and onlyif $f(u) \leq f(v)$.
$v \sim v$ (u indifferent v), If and only if $f(u) \not<= f(v)$ and $f(v) \not<= f(u)$.
where, $f(u)$ represents the objective function value of the decision vector u, $f(v)$ represents the objective function value of the decision vector v.

Definition 3

Suppose that $x^* \in \Omega$, if $f_i(x^*) \leq f_i(x)$ $(i = 1,2,\ldots,m)$ for all $x \in \Omega$, then X^* is one optimal solution of the MO (multi-objective optimization).

Definition 4

If one solution $x^* \in \Omega$ are non-dominated by any $x \in \Omega$, then x^* is called the Pareto optimal solution.

In the algorithm process, the multi-objective function optimization problems are solved by the non-dominated hierarchical genetic algorithm. Firstly, each individual in the group is given a certain virtual fitness value using the non-domain hierarchical method. Then, the traditional genetic algorithm is used to choose their mode, crossover and mutate. At last, the corresponding non-inferior solutions will be maintained.

3. Algorithm Design

3.1 The layered approach based on objective value

In this algorithm, a new layered approach based on objective value is introduced. The main idea of the approach is shown according to the following example. $A(20,\ 90)$ $B(25,\ 85)$ $C(30,\ 80)$ $D(35,\ 75)$ $E(40,\ 60)$ $F(45,\ 55)$ $H(40,\ 40)$ are the numbers of the example and the minimum number is taken as the best. After being layered, it can maintain the data of the first layer with the number of $A(20,\ 90)$ $B(25,\ 85)$ $C(30,\ 80)$ $D(35,\ 75)$ $H(40,\ 40)$ and the data of the second layer with the number of $D(35,\ 75)$ $E(40,\ 60)$.

According to the layered data, some phenomena can be found.

(1) In terms of the first objective, if the minimum of the first objectives has been found, this value will be divided into the first layer. For instance, in the number $A(20,\ 90)$, the data 20 is the minimum data of all the first objective, so $A(20,\ 90)$ is divided into the first layer.

(2)In terms of the second objective, if the minimum of the second objectives has been found, this data will be divided into the first layer. For instance, in the number $H(40,\ 40)$, the data 40 is the minimum data of all the second objective, so $H(40,\ 40)$ is divided into the first layer.

According to all the above, a result can be concluded that if there are m objectives, then the third objective can be layered to the m-th objective as the first or the second one. Here, the detail of the layered process

is given (take the objectives of path length and path delay as an example).

Layered process: specify mark $front = 1$ as the beginning.

Step 1: Start to look from the first objective:

① Find the individual with the minimum first objective values from all unmarked individuals, and mark 'Front'.

② Check out whether there exist individuals with the same minimum first objective values. If there exist the individuals, mark 'Front'.

③ Find out the individuals with the minimum second objective values among the marked individuals with mark Front and remove the mark front for the rest of the individuals which have the non-minimum second objective value.

Step 2: Start to look from the second objective:

① Find out the individuals with the minimum second objective value from the unmarked individuals or the individuals that marked 'front' and mark 'front+1'.

② Then Check out whether there exist individuals with the same minimum second objective value. If it is true, make the same mark.

③ Find out the individuals that has minimum second objective value with the mark 'front+1' and remove the mark of the individuals which have non-minimum second objective value.

④ convert the mark of the individuals from 'front+1' to 'front'.

Step 3: The third objective to the *m-th* objective can be done as the above.

Step 4: Compare all the individuals with the mark 'front'. If there exist the individuals dominated, remove its mark.

Step 5: Find the individuals which are unrelated to the marked individuals in the unmarked individuals and then mark 'front'.

Step 6: Find out the non-dominated individuals of the current population with the mark 'front'.

Step 7: Increment the mark 'front' by one.

Step 8: Loop from step 1 to Step 6 until all the individuals are identified.

3.2 Selection mechanism

Aimed at the slow convergence problem for most of the evolution algorithms, a new selective mechanism is proposed to accelerate the convergence speed of the populations that the idea is to cycle through the first 1/2 layer according to the hierarchy until the number of the individual copied is the same size as the population.

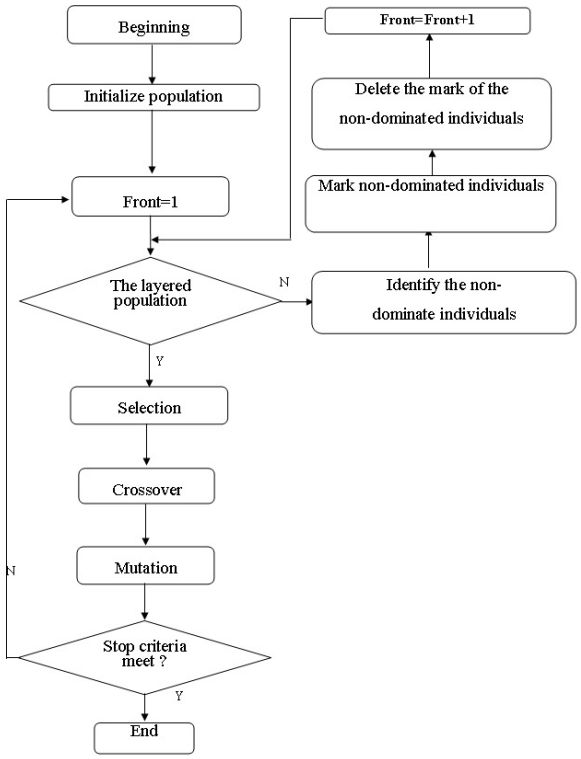

Figure 1 Flowchart of the Algorithm

For example, if the population size is 10, then they can be divided into 4 layers. Three individuals are at the first layer, two individuals at the second layer, three individuals at the third layer and two individuals in the 4th layer.

The number of the loop copy is decided by the number of the layer divided by 2, so the result is 4/2=2 and copy the first two layers. Because of the number of the population is 10, so at first, three individuals of the first layer is copied, then, copy two individuals of the second layer. Repeat it again until the number of the individual copied is the same size as the population.

4. Algorithm Descriptions

In order to better understand the operation process of the algorithm, the following steps will describe the details of this algorithm. Suppose the size of the evolutionary population P is N.

Step 1: suppose the variable t is equal to 0, the algorithm randomly generates the initial population with size n.

Step 2: Calculate the objective value of each individual in the initial population $p(0)$.

Step 3: According to the objective value, the population $p(0)$ is layered by the objective value layered method and make the variable 'front' equal to 1.

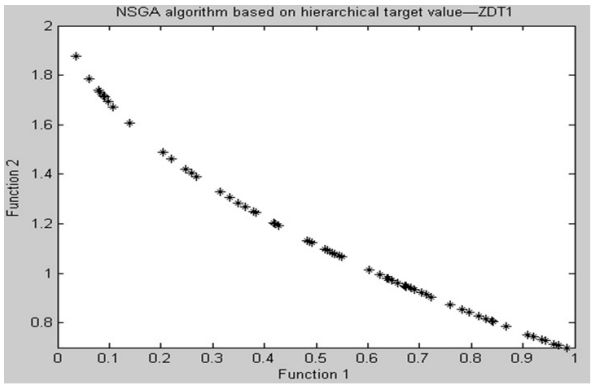

Figure 2 The 10th objective space of NSGA algorithm based on hierarchical objective value

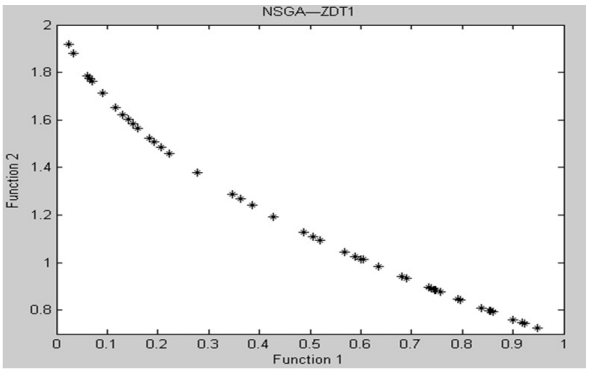

Figure 3 The 100th objective space of NSGA algorithm

① Select the first layer non-dominate individuals of each objective value and mark 'front'.

② Find out the dominated individuals having the mark 'front' and then remove its mark.

③ Make a judgment whether all the individuals have been layered. If not, increase 1 for front and implement step 1; otherwise implement step 4.

Step 4: Do selective operation according to the result of the layering operating.

Step 5: Do crossover and mutate operation on the selective individuals under a certain probability.

Step 6: If the variable t (the evolution generation) is less than the maximum evolution generation, implement step 2; otherwise, end the operation.

The flowchart of the Algorithm is given as Figure 1.

5. Experimental Result

This paper uses two common multi-objective test functions to verify the performance of the non-dominated sorting genetic algorithm objective based on layer. These two functions ZDT1 and ZDT2 were introduced by Zitzler and Deb in references [12]. The two functions are the multi-objective test functions of the non-convex and convex Pareto optimal solution set.

Example1 ZDT1The non-convex Pareto optimal solution set

$$f_1(x) = x_1 \ ,$$

$$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}] \ ,$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \ ,$$

$$x_i \in [0,1], \ i = 1,\ldots,n, \ n = 6.$$

The parameters of the algorithm for example1 are as follows: the population size N is equal to 200, the termination condition t of the algorithm is equal to 10, the crossover probability is set to 0.8, the mutation probability is set to 0.1 and the number of the individual goal m is equal to 2. The parameters of the NSGA are as follows: the population size N is equal to 200, the termination condition t of the algorithm is equal to 200, the crossover probability is set to 0.8, the mutation probability is set to 0.1 and the number of the individual goal m is equal to 2.

The simulation results are shown in Figure 2. The simulation results of NSGA-II algorithm are shown in Figure 3:

Example 2 ZDT2: The convex Pareto optimal solution set.

$$f_1(x) = x_1 \ ,$$

$$f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2] \ ,$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \ ,$$

$$x_i \in [0,1], \ i = 1,\ldots,n, \ n = 6.$$

The parameters of the algorithm for the example 2 are as follows: the population size N is equal to 200, the termination condition t of the algorithm is equal to 10, the crossover probability is set to 0.8, the mutation probability is set to 0.1 and the number of the individual goal m is equal to 2. The parameters of the NSGA are as follows: the population size N is equal to 200, the termination condition t of the algorithm is equal to 200, the crossover probability is set to 0.8, the mutation probability is set to 0.1 and the number of the individual goal m is equal to 2.

The simulation results are shown in Figure 4.

The simulation results of NSGA-II algorithm are shown in Figure 5:

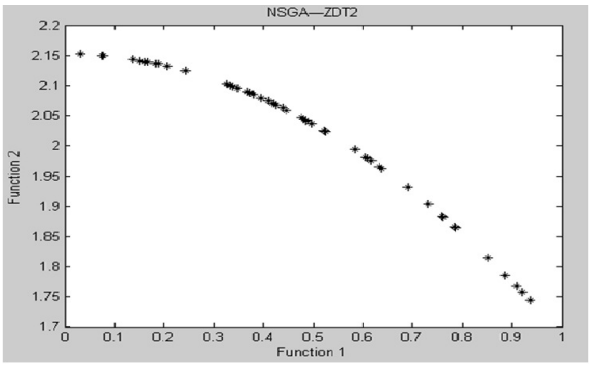

Figure 4 The 10th objective space of NSGA algorithm based on hierarchical objective value

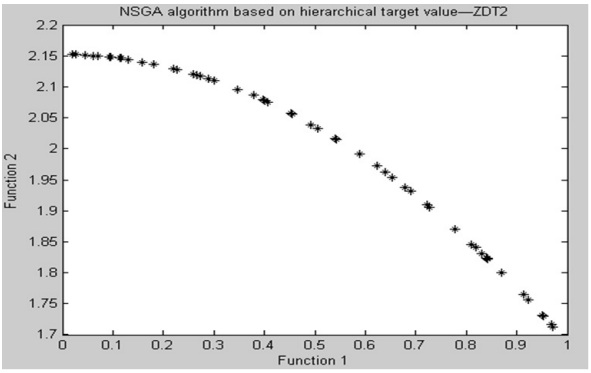

Figure 5 The 100th objective space of NSGA algorithm

From the above simulation results, it can be seen that the distribution of non-inferior solution of this algorithm is just the same as the NSGA algorithm. But the convergence speed of this algorithm is much faster than NSGA algorithm.

6. The Application of The Algorithm to Network Optimization

6.1 Problem description

In real application, the network design problem is also a multi-objective problem. A telecommunication network design is analyzed using various measurements such as communication cost, transmission delay, and routing policy.

Given locations of the nodes and other information such as cost, distance and traffic between any pair of nodes, the main problem of a multi-objective network optimization design is how to construct a network which minimizes total link costs subject to certain constraints. There are n network nodesthe netline is used to connect node s to the node t, and only a line is connected from the node s to the node t, with no looping. In this paper, the shortest path and minimum delay is considered as the objectives of multi-objective optimization problem.

These two objectives of the network topology could be described through the undirected graph $G = (V, E, C, D)$. Set V is the set with n network nodes, $V = [V_1, V_2, \ldots, V_n]$, E is the set of links between node pair, $E = \{(s_i, t_i) | i = 1, 2, \ldots, m.s_i, t_i \in V\}$. Only a directed arc exists between any two adjacent nodes. The path length between two adjacent nodes (s_i, t_i) is denoted as C_i $(C_i \geq 0)$, the delay is denoted as D_i $(D_i \geq 0)$. $C = [C_1, C_2, \ldots, C_m]$ is the vector of path length, $D = [D_1, D_2, \ldots, D_m]$ is the vector of delay. Suppose that s is the source node, t is the destination node, and then the optimal path from the node s to the node t is denoted as R_{s-t}, the optimal length of path is denoted as C_{s-t}, the optimal delay is denoted as D_{s-t}, the mathematical model of the network optimization can be written as a multi-objective 0-1 integer programming problem as follows.

Minimize $y = F(x) = (C_{s-t}, D_{s-t})$
where, C_{s-t} is the path length.

$$C_{s-t} = \sum_{i=s}^{t} \sum_{j \neq i, j=s}^{t} C_{ij} \cdot x_{ij}$$

D_{s-t} is the path delay:

$$D_{s-t} = \sum_{i=s}^{t} \sum_{j \neq i, j=s}^{t} D_{ij} \cdot x_{ij}$$

The constraint is:

$$\sum_{j=s, j \neq i}^{t} x_{ij} - \sum_{k=s, k \neq i}^{t} x_{ki} = \begin{cases} 1, & i = s; \\ -1, & i = t; \\ 0, & i \neq s, i \neq t; \end{cases}$$

$$x_{ij} = \begin{cases} 1, & (i, j) \in R_{s-t}; \\ 0, & (i, j) \notin R_{s-t}; \end{cases}$$

When $(i, j) \in R_{s-t}$, that is (i, j) the shortest path, $x_{ij} = 1$, otherwise, $x_{ij} = 0$.

Multi-objective network optimum problem is a typical combinatorial optimization problem, and it was a NP-hard, that is their possible searching number of paths is an exponential growth with the total number of network nodes growing, so it is difficult to find out the optimal solutions, only to find out relatively optimal non-dominated solutions. This paper solves the multi-objective network using an improved genetic algorithm and can get better optimal solutions of non-dominated solutions.

6.2 Applications of hierarchic objective genetic algorithm

The algorithm is used to reflect primarily the following aspects:

(1) Coding:

The most critical step in applying a genetic algorithm to a network design is to choose a way to represent a solution to the problem in a chromosome. A chromosome is the corresponding representation or genetic encoding of the individual (solution). Although the template of genetic algorithms is the same, different solution representations lead to complete different algorithms with different efficiency. In the algorithm, a chromosome represents a path from source node to destination node, which may be a feasible solution and may be not, therefore the checking of the routing and constraints during each chromosome evaluation is inevitable, so that a large amount of computation time is taken on such checking and a repair mechanism is required when any of the constraints is violated, which can result in the loss of important genetic material and, thus reduce the efficiency of the search.

In this genetic algorithm, to avoid such inefficient checking of violation of the constraints, a repair mechanism, a chromosome, no matter it is generated initially or by genetic operators, is a feasible solution. In practice, different path has different nodes numbers in the network, so the length of chromosome is different.

(2) Population generation

Assuming the number of nodes at network is m, Population size n. The chromosome is combined by the node index $1,2\ldots,m$, and the first point (source node of path) is s, and the last point node is t. The maximum coding length is m, which is each node in the network belongs to the path. The minimum length of possible path is 2, directly from the start node to the end node, no other nodes between s and t.

In the process of generating the first path, the first node is fixed to node s, the second node k ($k \neq s$ *and* $k \in [1,m]$) is produced by using random function. When $k = t$, the first path coding is ended.

Other paths can be generated by the same method.

(3) The objective layered approach

The objective values include the path length and the delay. They can be calculated as follows:

$$C_{s-t} = \sum_{i=s}^{t} \sum_{j \neq i, j=s}^{t} C_{ij} \cdot x_{ij}$$

$$D_{s-t} = \sum_{i=s}^{t} \sum_{j \neq i, j=s}^{t} D_{ij} \cdot x_{ij}$$

Layered process: Specify mark $Front = 1$ as the beginning.

Step 1: Start to look from the first objective path length:

① Find out the shortest length from all unmarked individuals, and mark 'front'.

② Check out whether there exist individuals with the same length as the shortest path length.

③ Find out the individuals with the shortest delay among the marked shortest path and remove the rest of the individuals which have non-minimum path delay.

Step 2: Start to look from the second objective path length:

① Find out the individuals with the shortest delay from the unmarked path or the path marked 'front' and mark 'front+1'.

② Check out whether there exist individuals with the same length as the shortest path length. If it is true, make the same mark.

③ Find out the individuals with the shortest path length with the mark of 'front+1' and remove the mark of the individuals which have non-minimum path length.

④ Convert the mark of the individuals from 'front+1' to 'front'

Step 3: Compare all the individuals with the mark 'front'. If there exist the individuals that are dominated, remove its mark.

Step 4: Find out the individuals which are unrelated to the marked individuals in the unmarked individuals and then mark 'front'.

Step 5: Find out the non-dominated individuals of the current population with the mark 'front'.

Step 6: Increment the mark 'front' by one.

Step 7: Loop from step 1 to Step 5 until all the individuals are identified.

(4) Crossover operation

Genetic operations here mainly include selection, crossover and mutation.

In order to speed up convergence, the chromosomes are selected from the first 1/2 layers according to the above hierarchy in selection process. Its benefit is that the fitness of the individual does not need to calculate, the individuals are choose directly to crossover and mutation operations.

There are lots of cross means. In this algorithm, because the encoded lengths of the paths may not be same, this will cause great inconvenience to crossover. In order to minimize the complexity of the crossover, here a single point crossover is used. The crossover procedure is the following:

First calculate cross paths with crossover probability, and then randomly generate cross-location, next judge the two genes of the first individuals on in this cross-location. If the two genes are not the start node or end node, cross the two genes; otherwise the two

genes do not cross.

In the crossover process, a single point crossover is used. First of all, a cross point is produced randomly. Two parent chromosomes are then partitioned into two parts. Two new chromosomes are generated by exchanging back part of two parent chromosomes. If there are duplicated nodes in the new chromosomes after the exchanging, the operation of swapping duplicated nodes between the two resulting chromosomes is used.

(5) Mutation operation

Mutation, a single point mutation is employed. Firstly, the number of mutation is calculated by mutation probability. Then the mutation location is generated randomly and a new node between 1 and m is generated randomly. If this node exists on the individual, randomly repeat generating another node until a node appears which does not exist on the individual.

6.3 The experiment result for network design

The network design problem is given by two 8*8 matrices. A matrix is the path length matrix $A = [a_{ij}]_{n\times n}$, where, a_{ij} is the path length between of the node i and the node j.

$$\begin{bmatrix} 0 & 6 & 3 & 5 & 15 & 20 & 50 & 90 \\ 6 & 0 & 8 & 17 & 20 & 30 & 12 & 16 \\ 3 & 8 & 0 & 50 & 70 & 8 & 2 & 35 \\ 5 & 17 & 50 & 0 & 60 & 30 & 24 & 10 \\ 15 & 20 & 70 & 60 & 0 & 20 & 50 & 5 \\ 20 & 30 & 8 & 30 & 20 & 0 & 3 & 50 \\ 15 & 12 & 2 & 24 & 10 & 3 & 0 & 20 \\ 90 & 16 & 35 & 10 & 5 & 50 & 20 & 0 \end{bmatrix}$$

Another is delay matrix $B = [b_{ij}]_{n\times n}$, where, b_{ij} is the delay between the node i and j.

$$\begin{bmatrix} 0 & 3 & 5 & 4 & 8 & 10 & 15 & 13 \\ 3 & 0 & 11 & 6 & 7 & 5 & 12 & 1 \\ 5 & 11 & 0 & 4 & 2 & 8 & 30 & 27 \\ 4 & 6 & 4 & 0 & 20 & 17 & 11 & 2 \\ 8 & 7 & 2 & 20 & 0 & 13 & 16 & 2 \\ 10 & 15 & 8 & 17 & 13 & 0 & 7 & 15 \\ 15 & 2 & 30 & 11 & 16 & 7 & 0 & 8 \\ 13 & 1 & 27 & 2 & 3 & 15 & 8 & 0 \end{bmatrix}$$

In the process of using the algorithm to solve this network, the parameters are set as follows:

The source node is 1 and the destination node is node 8, the objective numbers is $m = 2$, the size of population is $N = 50$, the probability of crossover is 0.8, the probability of mutation is 0.1, the termination condition is maximum $generation = 100$.

The optimum path is $(1,\ 4,\ 8)$ and $(1,\ 2,\ 8)$, the path length and postpone are $(15,\ 6)$ and $(22,\ 4)$.

7. Conclusion

Aimed at the long running time of layered method in NSGA algorithm and the slow convergence of the Pareto optimal solution set resulted from the random selection mechanism. This paper proposes a new layered objective approach to deal with the long time computation during the layering process and also proposes a new selection mechanism to solve the slow convergence problem of the Pareto optimal solution set. There is an advantage of no need to calculate the fitness value with this selection mechanism. The experimental results show that this algorithm with the new objective value and the selection mechanism can greatly accelerate the convergence speed.

The network designs are challenging problems, although the genetic algorithm proposed can achieve the basic design task and give an optimal solution, it is necessary to compare performance of this genetic algorithm with the best solution which could be very difficult to get because it is a NP-hard problem.

The improved multi-objective optimization approach based on layered objective is used to solve the network design problems. the genetic algorithm designed in such a way that it can be easily extended to a high-connectivity network design while some methods have a difficulty in doing so or are impossible to do so; the genetic algorithm is suitable for large network designs while some methods could break down when the network size increases up to a certain extent due to the realistic CPU constraints.

References

[1] J.D. Schaffer. “Multiple Objective Optimization with Vector Evaluated Genetic Algorithm”, *Proc of 1st Int. Conf. Genetic Algorithm*, Hillsdale, New Jersey, 1985: 93-100.

[2] Zhimin Fang. “Artificial Intelligence and Computational Intelligence”, AICI ’09. *International Conference on 2009*. 2009(1): 373-377.

[3] Jiaquan Gao. “WBMOIGA: Weight-based multiobjective immune genetic algorithm and its application”. *Information and Automation, 2009*. ICIA ’09. International Conference on. 2009: 1-6.

[4] Johan Andersson. “Multiobjective Optimization in Engineering Design, Applications to Fluid Power Systems [Dissertations]”. *Division of Fluid and echanical Engineering Systems*, Department of Mechanical Engineering Link*ö*pings universitet SE-581 83 Link*ö*ping, Sweden Link*ö*ping 2001.

[5] Goldberg D.,“ Genetic Algorithms in Search and Machine Learning”. Reading, Addison Wesley, 1989.

[6] Srinivas N, Deb K. “Multi-objective optimization using nondominated sorting in genetic algorithms”. *Evolutionary Computation*. 1994, 2(3): 221-248.

[7] Hom J, Nafpliotis N, Goldberg DE. “A niched Pareto genetic algorithm for multi-objective optimization”. In: *Procof the 1st IEEE Conf. on Evolutionary Computation*. Piscataway: IEEE World Congress on Computational Computation. 1994, 1: 82-87.

[8] Fonseca CM, Fleming PJ. “Mul-tiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: A unified formulation”. *IEEE Trans. Systems, Man, and Cybernetics*, 1998, 28(1): 26-37.

[9] Zitzler E, Thiele L. “Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach.” *IEEE Trans. Evolutionary Computation*. 1999, 3(9): 257-271.

[10] Deb Kalyanmoy, Sagrawal, Amrit Pratab. “A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II [J].” *IEEE Transactions on Evolutionary Computation*, 2002, 6(2): l82-197.

[11] Pan XiaoYin, Liu Fang, Jiao LiCheng. “Multiobject Social Evolutionary Algorithm Based on Multi-Agent”. *Journal of Software*. 20(7): 1703-1713.

[12] Zttzler E, Deb K, Thiele L. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results [J]”. *Evolutionary Computation*, 2000, 8(2): 173-195.

[13] Runhe Huang, Jianhua Ma, D. Frank Hsu. “A Genetic Algorithm for Optimal 3-connected Telecommunication Network Designs”. *Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN ’97)*. 1997, pp.344-350.

[14] Gen M, Cheng R W, Lin L. “Network models and optimization [M]”. London: Springer-Verlag, 2008: 30-82.

[15] Ahn C W. “A genetic algorithm for shortest path routing problem and the sizing of populations [J]”. *IEEE Trans on Evolutionary Computation*, 2002, 6(6): 566-579.

[16] Wen H Y.“ Genetic algorithm based computation of shortest path in discrete-time networks [J]”. *Journal of South China University of Technology*, 2008, 36(2): 13-16.