A New Simple and Highly Efficient Multi-objective Optimal Evolutionary Algorithm

Chuan Shi
State Key Laboratory of Software Engineering,
Wuhan University,
Wuhan, China, 430072
ch_shi@hotmail.com

Yan Li
Computation Center,
Wuhan University,
Wuhan, China, 430072
llyyan2000@21cn.com

Li-shan Kang
State Key Laboratory of Software Engineering,
Wuhan University,
Wuhan, China, 430072
kang_whu@yahoo.com

Abstract- Multi-objective optimal evolutionary algorithms (MOEA) are effective algorithms to solve multi-objective optimal problem (MOP). Because ranking which used by most MOEAs has some disadvantages, in this paper, we propose a new method that uses better function to compare candidate solutions and tree structure to express the relationship of solutions. Experiments show that the new algorithm can converge to the Pareto front, and maintains the diversity of population. When the algorithm is extended to a MOP with constraints, it can also get a good result. Most important of all, the algorithm is simple but highly efficient.

1 Introduction

Multi-objective optimization problems are those problems that involve simultaneous optimization of more than two objectives (often competing) and usually with no single optimal solution [1]. Traditional multi-objective optimal approaches convert multi-objective optimization problems into single objective optimization problems, and then use well-studied algorithms for single objective optimization to solve the problem. These approaches are attractive and popular, but have many disadvantages.

Recently, a new method - Multi-objective Optimal Evolutionary Algorithm (MOEA) - has been used to solve multi-objective optimization problems. There are some efficient MOEAs, for example vector evaluated genetic algorithm (VEGA)[2], Hajela and Lin's genetic algorithm (HLGA)[3], niched Pareto genetic algorithm (NPGA) [4], Fonseca and Fleming's multi-objective EA(FFGA)[5], nondominated sorting genetic algorithm (NSGA)[6] and the strength Pareto evolutionary algorithm (SPEA)[7]. Recently some of the proposed methods have made further progress, for instance, NSGAII[8], SPEA2[9] . A good MOEA must satisfy the two aspects: 1) the distance between the resulting nondominated set and the Pareto-optimal front should be minimized; 2) a good distribution of the gotten solutions is desirable. Thus, some methods such as Pareto ranking, mating restriction, Pareto niching, and fitness sharing[11] are used in MOEAs.

In this paper, we firstly analyze the disadvantages of MOEAs, and then introduce a better function to compare the candidate solutions and a tree structure to express the relationship of solutions. At last we extend the algorithm to solve the MOP with constraint.

2 Disadvantages of Former MOEAs

Compared with single objective optimization problems, one of the difficulties of multi-objective optimization problems is to represent the relationship of the candidate solutions. For a MOP, the relationship is multi-dimensional and difficult to compare. At present, the most common method is to use ranking technique [11], which gives a rank value for each candidate solution according to a given rule, and uses the rank value to compare these solutions.

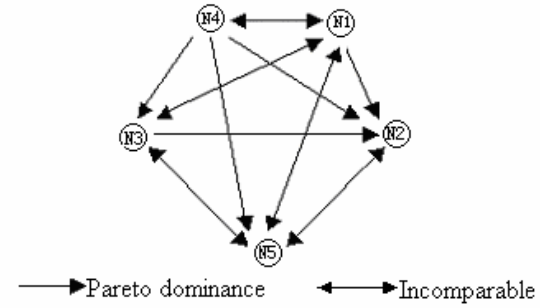

Fig.1 Solutions represent by chart

Usually a chart is used to represent the relationship between every two candidate solutions. however, although the chart can represent the relationship very well, it may become redundant because Pareto dominance are transferable and because only the Pareto optimal set of every generation should be considered. Fig.1 shows the relationship between five nodes (In the figure, a node represents an individual. The same representation will be used throughout the paper). In order to get rid of the redundancy, the new algorithm uses tree structure to represent the relationship, see Fig.2.

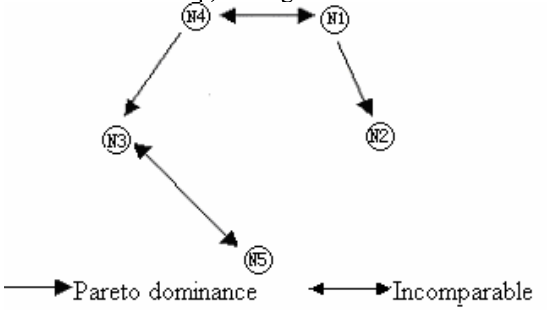

Fig.2 Solutions represent by tree

A better function is introduced to compare two candidate solutions and tree structure to represent the relationship between every two solutions as follows:

Definition 1: Pareto dominance [11]

If vector $U=(u_1,u_2,\cdots u_k)$ Pareto dominates $V=(v_1,v_2,\cdots v_k)$, denotes as $U\preceq V$, that is $U\preceq V$ if and only if $\forall i\in\{1,2,\cdots k\}\quad u_i\le v_i\wedge \exists i\in\{1,2,\cdots k\}\quad u_i<v_i$

Definition 2: better function

Considering a MOP with k objectives:

$$F(\mathbf{x})=(f_1(\mathbf{x}),f_2(\mathbf{x}),\cdots,f_k(\mathbf{x}))$$

A better function is defined as follows:

$$better(\mathbf{x}_1,\mathbf{x}_2)=\begin{cases}1 & F(\mathbf{x}_1)\preceq F(\mathbf{x}_2)\\ 2 & F(\mathbf{x}_2)\preceq F(\mathbf{x}_1)\\ 0 & \text{others}\end{cases}$$

Definition 3: tree structure

```
struct node
{   int id;      //the ID of the node
    int depth;   //the number of nodes not better than a given node (including itself)
    struct node *child;  //a pointer that points to node worse than the given node
    struct node *lsibling, *rsibling; // a pointer that points to the node incomparable with the given node
}
```

In order to compare two Pareto optimal sets, introduce the following definition [10]:

Definition 4: Let $X,Y\subseteq\Omega$ be two sets of decision vectors. The function C maps the ordered pair (X, Y) to the interval [0, 1]:

$$C(X,Y)=\frac{|\{b\in Y;\exists\, a\in X: a\preceq b\}|}{|Y|}$$

The value $C(X,Y)=1$ means that all solutions in Y are dominated by or equal to solutions in X. The opposite, $C(X,Y)=0$ represents the situation when none of the solutions in Y are covered by the set X. Noted that both $C(X,Y)$ and $C(Y,X)$ have to be considered, since $C(Y,X)$ is not necessarily equal to $1-C(X,Y)$.

3 The Description and Analysis of the New Algorithm

3.1 The Description of the New Algorithm

The algorithm can be described as follows:

Step 1 t: = 0.

Step 2 Initialize randomly $P(t)=\{\mathbf{x}_1,\mathbf{x}_2,\cdots\mathbf{x}_N\}$ (N is the population size), where $\mathbf{x}_i\in U$.

Step 3 Sort the individuals using tree structure according to the better function.

Step 4 For any given $\mathbf{x}$, **if** for every $\mathbf{y}$ (not equal to $\mathbf{x}$) in $P(t)$, better($\mathbf{x},\mathbf{y}$) == 0 **then** stop.

Step 5 Generate the offspring

Choose M individuals randomly from the population; M individuals make up the subspace V. Randomly choose one individual $\mathbf{x}_c$ from $V\bigcap\Omega$. (The crossover operator of multi-parents)

For the new individual $\mathbf{x}_c$, sort it in the tree according to the better function.

Step 6 Find the worst individual (the leaf node of the node with the biggest depth) and get rid of it.

Step 7 t: = t+1, then turn to **Step 4**.

Two functions in the algorithm are used to sort by tree structure.

a) AddinTree (node *proot, node *pnewnode) ; pnewnode is added to the left subtree of the tree whose tree root is proot.

b) AddinSibling (node *pparent, node *pchild, node *pnewnode) ; pnewnode is added to the sibling subtree of pchild; pparent is the parent node of pchild.

The two functions are described as follows:

```
AddinTree (node *proot, node *pnewnode )
Begin
    proot->depth=proot->depth+pnewnode->depth;
    if ( proot->child == null )
     then  let pnewnode be the child of proot;
     else        AddinSibling(proot,proot->child, pnewnode);
End
AddinSibling ( node *pparent, node *pchild, node *pnewnode)
Begin
   Switch(better(pnewnode, pchild))
   Begin
     case 0:
     if (pchild->rsibling == null)
     then let pnewnode be the right sibling of pchild;
     else AddinSibling(pparent, pchild->rsibling, pnewnode);
      break;
      case 1:
     pnewnode takes the place of pchild;
      AddinTree(pnewnode,pchild);
     while there is a pnode (right sibling of pnewnode) which is worse than the pnewnode, do AddinTree(pnewnode, pnode);
     while the depth of pnewnode is bigger than that of its left sibling, do pnewnode moves in the left direction along the chain.
     break ;
     case 2:
     AddinTree(pchild, pnewnode);
      while the depth of pchild is bigger than that of its left sibling, do pchild moves in the left direction along the chain.
      break;
   End
End
```

Because the process of building a sort-tree is

complicated, an example demonstrates the algorithm. There are five nodes in Fig.1. Their relationship is shown in Table 1, where "--" means a node does not compare with itself. The input order of these nodes is N1, N2, N3, N4, N5. The process of building the sort-tree is shown in Fig.3, where the new node N6 represents the offspring.

	N1	N2	N3	N4	N5	N6
N1	--	1	0	0	0	0
N2	2	--	2	2	0	0
N3	0	1	--	2	0	0
N4	0	1	1	--	1	0
N5	0	0	0	2	--	2
N6	0	0	0	0	1	--

Table 1 The relationship of nodes

Step 1: build a tree rooted with node N1, see Fig.3.a.

Step 2: node N2 is worse than node N1, so N2 is the child of node N1, see Fig.3.b.

Step 3: node N3 is incomparable with node N1, so is the right sibling of node N1, see Fig.3.c.

Step 4: node N4 is incomparable with node N1, then become the right sibling of node N1, and node N4 is better than node N3, node N3 becomes the child of node N4, see Fig.3.d.

Step 5: the node N5 is incomparable with node N1, then be the right sibling of node N1; node N4 is better than node N5, node N5 becomes the child of node N4; and the node N5 is incomparable with node N3, then be the right sibling node of node N3, see Fig.3.e.

Till now, the five nodes have been input. Because the number of nodes worse than node N4 is more than that of node N1 (the depth of node N4 is 3, and the depth of node N1 is 2), then move up the sub-tree rooted with node N4, see Fig.3.f. The act to move the sub-tree is to ensure that the tree is normalized and easier to find the worst node.

Step 6 for the new-built node N6, it is incomparable with node N4, then be the right sibling of node N4, and it is incomparable with node N1, then be the right sibling of node N1, see Fig.3.g. Then the worst individual is node N5 (a leaf node of the node with biggest depth) and get rid of it, see Fig.3.h

3.2 Analysis of the new algorithm

The criterion to evaluate a MOP is the degree of closeness to the Pareto front and decentralization of the solutions [13]. The following paragraphs will analyze the new algorithm from the convergence, the diversity of solutions and the efficiency aspects.

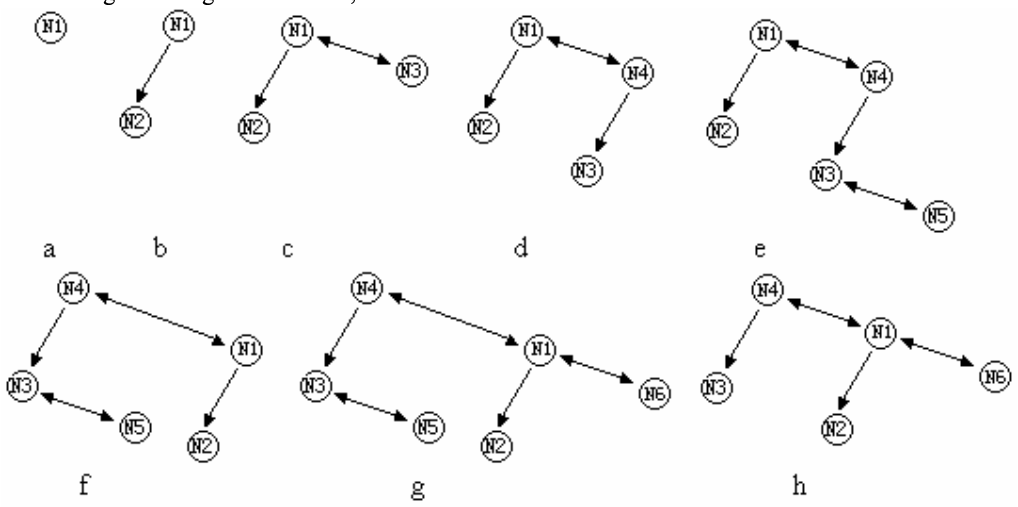

Fig. 3 The process of building a sort tree

3.2.1 Convergence

In order not to converge to local Pareto front, the new algorithm uses the crossover operator of multi-parents; see line 5 in the description of the new algorithm. The crossover operator of multi-parents is proposed in Ref. [14]. The way to make up the subspace is

$$V=\{\ \mathbf{x} \mid \mathbf{x}\in\Omega\ \ \mathbf{x}=\sum_{i=1}^{m} a_i \mathbf{x}_i\ \ \}$$

$(\sum_{i=1}^{m} a_i = 1,\ a_i \in [-0.5,1.5])$. The operator has been proved to be able to avoid local optimal effectively [14].

3.2.2 Diversity

In order to maintain the diversity, many MOEAs use Pareto niching and fitness sharing. These methods mainly restrict the number of solutions in the neighborhood of the good solution [11]. The cost in time is high. The new algorithm deletes the leaf node of the node with the biggest depth, which maintains the diversity naturally. Because a node with big depth means that there are many nodes in its neighborhood, deleting the leaf node of the node with the biggest depth forces the nodes in its neighborhood to leave its neighborhood. It makes solutions evenly distribute in the decision space. This is also proved in experiments.

3.2.3 Efficiency

Since comparing two candidate solutions is the main operation in the process to evaluate an individual, it is used as a criterion of time complexity of algorithms. Here denote n as the population size. By using ranking technique in the new algorithms, the space complexity is

usually $O(n^2)$, and the time complexity is $O(n^2)$ [11]. In the algorithm, with the use of tree structure, the space complexity is $O(n)$, and the time complexity is $O(1)$ in best condition, or $O(n)$ in the worst condition. Because the sorting process is complex, it is hard to calculate the time complexity in theory. So experiments are done to estimate the time complexity. In the experiments, add up the number of comparison for all nodes, and then divide the sum by the total number of nodes. The mean value is calculated and showed in Table 2. It shows the relationship between the average number of comparison and population size. From the result, it is believed that the new algorithm is better than former MOEAs both in space and time efficiency.

Population size	100	200	300	500	1000
Average number	9.3	12	14.5	20	34.5

Table 2 The relationship between the average number of comparison and population size

4 The Numerical Experiments

4.1 Test Function

In order to compare with other MOEAs, the test functions are derived from Ref. [10]. Because of the different encoding, the algorithm cannot calculate the fifth test function in Ref. [10].

Each of the test function defined below is structured in the same manner and consists itself of three functions f_1, g, h:

Minimize $\boldsymbol{T}(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x}))$

Subject to:

$f_2(\mathbf{x}) = g(x_2, \cdots, x_m) h(f_1(x_1), g(x_2, \cdots, x_m))$

where $\mathbf{x} = (x_1, x_2, \cdots, x_m)$

1) The test function T1 has a convex Pareto-optimal front:

$$f_1(x_1) = x_1$$
$$g(x_2, \cdots, x_m) = 1 + 9\sum_{i=2}^{m} x_i / (m-1)$$
$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

Where m=30, and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g(X) = 1$.

2) The test function T2 is the nonconvex counterpart to T1:

$$f_1(x_1) = x_1$$
$$g(x_2, \cdots, x_m) = 1 + 9\sum_{i=2}^{m} x_i / (m-1)$$
$$h(f_1, g) = 1 - (f_1 / g)^2$$

Where m=30, and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g(X) = 1$.

3) The test function T3 represents the discreteness feature; its Pareto-optimal front consists of several noncontiguous convex parts:

$$f_1(x_1) = x_1$$
$$g(x_2, ..., x_m) = 1 + 9\sum_{i=2}^{m} x_i / (m-1)$$
$$h(f_1, g) = 1 - \sqrt{f_1 / g} - (f_1 / g)\sin(10\pi f_1)$$

Where m=30,and $x_i \in [0,1]$ .The Pareto-optimal front is formed with $g(X) = 1$.

4) The test function T4 contains 21^9 local Pareto-optimal fronts and, therefore, tests for the EA's ability to deal with multimodality:

$$f_1(x_1) = x_1$$
$$g(x_2, \cdots, x_m) = 1 + 10(m-1) + \sum_{i=2}^{m} (x_i^2 - 10\cos(4\pi x_i))$$
$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

Where m=10, $x_1 \in [0,1]$,and $x_2, \cdots x_m \in [-5,5]$. The global Pareto-optimal front is formed with $g(\mathbf{x}) = 1$, the best local Pareto-optimal front with $g(\mathbf{x}) = 1.25$ .Note that not all local Pareto-optimal sets are distinguishable in the objective space.

5) The test function T5 includes two difficulties caused by the nonuniformity of the search space: first, the Pareto-optimal solutions are nonuniformly distributed along the global Pareto front (the front is biased for solutions for which $f_1(\mathbf{x})$ is near one); second, the density of the solutions is lowest near the Pareto-optimal front and highest away from the front :

$$f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$
$$g(x_2, \cdots, x_m) = 1 + 9((\sum_{i=2}^{m} x_i) / (m-1))^{0.25}$$
$$h(f_1, g) = 1 - (f_1 / g)^2$$

Where m=10,and $x_i \in [0,1]$ .The Pareto-optimal front is formed with $g(\mathbf{x}) = 1$.

4.2 Test Result

The test condition is PII 500MHZ, 128M memory.

The parameters of the new algorithm are adjusted as follows: population size N=100, subpopulation size M=10, the number of generations is 25000(The algorithm evaluates an individual in one generation. It is equal to 250 generations of other algorithms). Each function is calculated 30 times, and then save the Pareto-optimal set in the current population after each calculation. These Pareto-optimal sets are combined to form a new set, and then calculate the Pareto-optimal set in the new set as result. It is the same as other algorithms' result. The aim is to reduce the random influence.

The average running time of each test function is shown in Table 3.

Test function	T1	T2	T3	T4	T5
Average running time(s)	3	2.3	3	1.4	1

Table 3 Average running time for each test function

The results are shown in Fig.4 (Because the result of T4 is very poor, it is not visualized here).

According to definition 4, the algorithm is compared with other algorithms. The results are shown in Table 4 and Table 5(RAND represents a random search algorithm; SOEA represents a single-objective evolutionary algorithm using weighted-sum aggregation).

	RAND	FFGA	NPGA	HLGA	VEGA	NSGA	SOEA	SPEA
T1	1	1	1	1	1	1	1	0.912
T2	1	1	1	1	1	1	1	0.804
T3	1	1	1	1	1	1	0.526	0
T4	0.294	0	0	0	0	0	0	0
T5	1	1	1	0.333	0.333	0.889	0.714	0

Table 4 The *C* values of resulting solutions between our algorithm and other algorithms [10]

	RAND	FFGA	NPGA	HLGA	VEGA	NSGA	SOEA	SPEA
T1	0	0	0	0	0	0	0	0.066
T2	0	0	0	0	0	0	0	0.075
T3	0	0	0	0	0	0	0.082	1
T4	0.667	1	0.333	1	1	1	1	1
T5	0	0	0	0	0	0	0	0.001

Table 5 The *C* values of resulting solutions between other algorithms and our algorithm

After analyzing the two tables, the following conclusions can be drawn:

(1) The result of the algorithm is better than any other algorithms (except SPEA) except for T4.

(2) The result of our algorithm is better than that of SPEA for T1, T2. But for T3, T4, T5, it is worse than that of SPEA.

(3) For T4, the result of the algorithm is very poor. It shows that the algorithm has little ability to deal with multimodality.

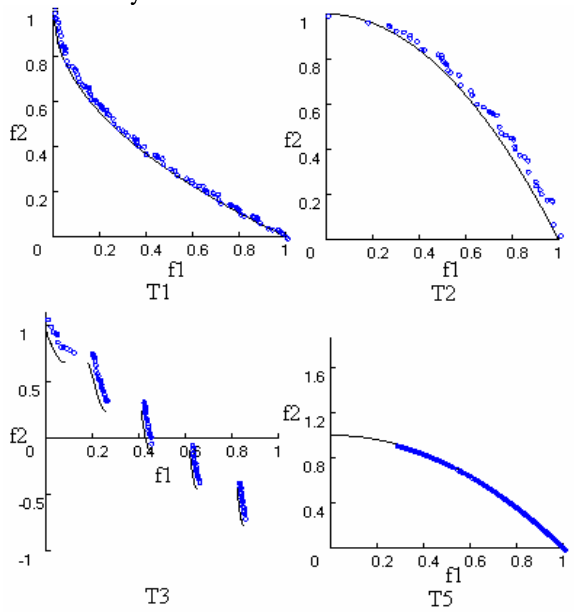

Fig.4 The result of test function

5 The Extension of the Algorithm

5.1 Preparation for the New Algorithm

Definition 5: In general, a MOP with constraints can be defined as follows:

$$\text{Minimize} \quad \boldsymbol{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}) \cdots f_k(\mathbf{x}))$$

$$\text{Subject to} \quad g_i(\mathbf{x}) \le 0$$

$$\mathbf{x} \in \grave{U}$$

$$(i = 1,2 \cdots m;\ \mathbf{x} = (x_1, x_2 \cdots x_n))$$

For a MOP with constraints, the feasible search space is the space generated by constraints. A legal solution can be compared by Pareto dominance, only if it satisfies the constraints. The constraints can be handled as follows:

Definition 6:

$$h_i(\mathbf{x}) = \begin{cases} 0 & g_i(\mathbf{x}) \le 0 \\ g_i(\mathbf{x}) & g_i(\mathbf{x}) > 0 \end{cases}$$

$$H(\mathbf{x}) = \sum_{i=1}^{m} h_i(\mathbf{x})$$

$$better1(\mathbf{x}) = \begin{cases} true & H(\mathbf{x}) = 0 \\ false & H(\mathbf{x}) > 0 \end{cases}$$

If the solutions cannot satisfy the constraints, they can be compared as follows:

Definition 7:

$$better2(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} true & H(\mathbf{x}_1) \le H(\mathbf{x}_2) \\ & \wedge H(\mathbf{x}_1) > 0 \wedge H(\mathbf{x}_2) > 0 \\ false & H(\mathbf{x}_1) > H(\mathbf{x}_2) \\ & \wedge H(\mathbf{x}_1) > 0 \wedge H(\mathbf{x}_2) > 0 \end{cases}$$

If the solutions satisfy the constraints, they can be compared as follows:

Definition 8:

$$better3(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} 1 & \boldsymbol{F}(\mathbf{x}_1) \preceq \boldsymbol{F}(\mathbf{x}_2) \wedge \\ & H(\mathbf{x}_1) = 0 \wedge H(\mathbf{x}_2) = 0 \\ 2 & \boldsymbol{F}(\mathbf{x}_2) \preceq \boldsymbol{F}(\mathbf{x}_1) \wedge \\ & H(\mathbf{x}_1) = 0 \wedge H(\mathbf{x}_2) = 0 \\ 0 & \text{others} \end{cases}$$

Because the relationship of solutions that cannot satisfy the constraints is linear, a queue can be used to express it. Define its data structure as follows:

Definition 9:

```
struct Qnode
{    int  id; //the ID of the node
     struct Qnode  *pnext   //a pointer that points
      to the next node
}
```

5.2 The description of new algorithm

For a MOP with constraints, the algorithm can be described as follows:

Step1 $t := 0$.

Step2 Initialize randomly $P(t) = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_N\}$

(N is the population size), where $\mathbf{x}_i \in \Omega$.

Step3 For each individual $\mathbf{x}_i$, sorts it as follows:

if better1 ($\mathbf{x}_i$) == true

then sort it using tree structure according to better3;

else sort it using queue structure according to better2;

Step4 For any given $\mathbf{x}$, **if** for every $\mathbf{y}$ (not equal to $\mathbf{x}$) in $P(t)$, better1($\mathbf{x}$) = true, better1($\mathbf{y}$) = true and better3($\mathbf{x}$, $\mathbf{y}$) = 0, **then** stop;

Step5 Generate the offspring

Choose M individuals randomly from the population; M individuals make up the subspace V. Randomly choose one individual $\mathbf{x}_c$ from $V \cap \Omega$.

For the new individual $\mathbf{x}_c$, sort it according to **Step 3**.

Step6 Delete the worst individual.

if There are nodes in the queue.

then Find the worst node in the queue, and get rid of it.

else Find the leaf node of the node with biggest depth in the tree, and get rid of it.

Step7 t: = t+1, then turn to **Step 4**.

5.3 Test Function

Test function (BNH) [1]

$Min(f_1, f_2)$

$$f_1 = 4*x_1^2 + 4*x_2^2$$

$$f_2 = (x_1 - 5)^2 + (x_2 - 5)^2$$

s.t:

$$(x_1 - 5)^2 + x_2^2 - 25 \le 0$$

$$(x_1 - 8)^2 + (x_2 + 3)^2 - 7.7 \le 0$$

$$x_1, x_2 \in [0,5]$$

Test function (TNK) [1]

$Min(f_1, f_2)$

$$f_1 = x_1$$

$$f_2 = x_2$$

s.t:

$$x_1^2 + x_2^2 - 1 - 0.1*\cos(16 \arctan(\frac{x_1}{x_2})) \le 0$$

$$(x_1 - 0.5)^2 + (x_1 - 0.5)^2 - 0.5 \le 0, x_1, x_2 \in [0,5]$$

The parameters of the algorithm are set as follows: population size N=100, subpopulation size M=10.Calculate each function for 5 times, and then save the Pareto-optimal set in the current population after each calculation. These Pareto-optimal sets are combined to form a new set, and then calculate the Pareto-optimal set in new set as result. For BNH, the average generations are 2150; the average running time is less than 1s. For TNK, the even generation is 3450; the even running time is about 1s. The results are shown in Fig.5.

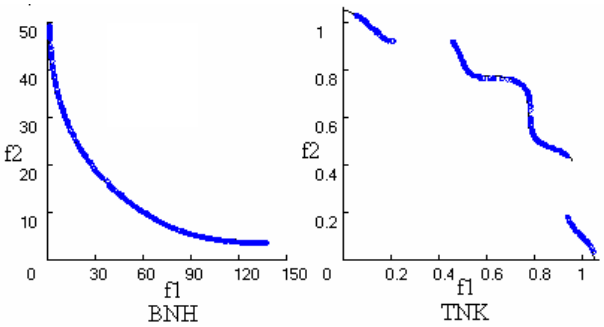

Fig.5 the result of test function with constraint

6. Conclusions

In this paper, we propose a better function to compare candidate solutions and a suitable data structure—tree structure —to express the relationship of candidate solutions. Experiments show that the solutions converge to the Pareto front and maintain the diversity. The new algorithm is better than former MOEAs both in space and time efficiency. It is simple but highly efficient. When extending the algorithm to solve a MOP with constraints, it also gets good result.

But there is still a lot to be done in future: to compute the time efficiency of sorting in the algorithm according to tree structure; new good operators should be introduced to solve multimodality and so on.

Acknowledgement

This work was supported by National Natural Science Foundation of China (No. 40275034, No. 60133010, No. 60073043, No. 70071042).

Bibliography

[1] Kalyanmoy Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Chichester, NewYork, Weinheim, Brisbane. 2001,330-342.

[2] Schaffer J D. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms: [Ph. D thesis]*, Vanderbilt: Vanderbilt University, 1984.

[3] Hajela P C, Lin Y. *Genetic Search Strategies in Multi-criterion Optimal Design*. Structural Optimization, 1992,4:99-107.

[4] Horn J, Nafpliotis N. *Multi-objective Optimization Using the Niched Pareto Genetic algorithm*. IlliGAL Report 93005, Illionois Genetic Algorithms Laboratory, University of Illionois, Urbana, Champaign.1993.

[5] Fonseca C M, Fleming P J. *Genetic Algorithms for Multi-objective Optimization: Formulation, discussion and generalization*. In: S Forrest Ed. Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California: Morgan Kaufmann, 1993. 416-423.

[6] Srinivas N, Deb K. *Multi-objective Optimization Using Non-dominated Sorting in Genetic Algorithms*. Evolutionary Computation, 1994, 2 (3): 221-248.

[7] Zitzler E. *Evolutionary Algorithms for Multi-objective Optimization: Methods and Applications: [Ph. D Thesis]*, Zurich Switzerland: Swiss Federal Institute of Technology (ETH). 1999.

[8] Deb K, Agrawal S, Pratap A, et al. *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGAII*. In: M.S. et al. Ed. Parallel Problem Solving from Nature –PPSN VI, Berlin: Springer. 2000.849-858.

[9] Zitzler E, Laumanns, M, Thiele L. *SPEA2:Improving the Strength Pareto Evolutionary Algorithm*. TIK-Reprot 103. ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, SwitxerLand. 2001.

[10] Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele. *Comparison of Multi-objective Evolutinary Algorithms, Empirical Results*. Evolutionary Computation, 2000, 18(2):173-195.

[11] David A Van, Veldhuizen, Gary B Lamont. *Multi-objective Evolutionary Algorithms, Analyzingthe State-of-the-Art*. Evolutionary Computation, 2000, 18(2): 125-147.

[12] IKEDA Kokolo1. *Failure of Pareto-based MOEAs: Does Non-dominated Really MeaNear to Optimal?* Proceedings of the 2001 IEEE Congress on Evolutionary Computation. Seoul, 2001, 27-30.

[13] Kalyanmay Deb, Jeffrey Horn. *Introduction to the Special Issue Multicriterion Optimization*. Evolutionary Computation, 2000, 8(2): 3-4.

[14] Guo Tao, Kang Li-shan, Li Yan. *A New algorithm for function Optimization Problems with Non-equality constraints*. Journal of Wuhan University (Natural Science Edition), 1998, 45(B): 771-775.