# Fitness Function for Finding out Robust Solutions on Time-Varying Functions

Hisashi Handa
Graduate School of Natural Science and Technology
Okayama University
Tsushima-Naka 3-1-1, Okayama 700-8530, Japan
handa@sdc.it.okayama-u.ac.jp

## ABSTRACT

Evolutionary Computations in dynamic/uncertain environments have attracted much attention. Studies regarding this research subjects can be classified into four categories: Noise, Robustness, Fitness approximation, and Time-Varying function. In research on Time-Varying function, the tracking property over changes of fitness landscape has been broadly and deeply researched so far. In this paper, instead of tracking new peaks, robust solution to Time-Varying functions is introduced. Moreover, two weighted fitness functions, Exponential Weight and Linear Weight, are proposed. Experiments on modified Branke's benchmark problems on Time-Varying function reveal the effectiveness of the weighted approaches.

## Categories and Subject Descriptors

F.1.2 [**Computation by abstract devices**]: Modes of Computation—*Probabilistic computation*; G.1.6 [**Numerical Analysis**]: Optimization; I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

## General Terms

Algorithms

## Keywords

Robust Solution, Dynamic Environments, Time-Varying Functions

## 1. INTRODUCTION

Because of the population/probabilistic search of Evolutionary Computations, dynamic/uncertain environments are one of the most important application areas of Evolutionary Computations. Robustness and Time-Varying function are key subtopics of studies of Evolutionary Computations on dynamic/uncertain environments. In general, robust solutions are needed if design variables are subject to be under perturbations. In this case, in order to evaluate the fitness of individuals precisely, several fitness samplings are required. On the other hand, Time-Varying function is often formulated by a temporal series of deterministic functions. Evolutionary Computation studies based on Time-Varying function try to yield new algorithms which can always pursue optimal peaks moving in design variable space. Now, suppose that acquired design variable values are manipulated by human. Dynamic solutions by such tracking Evolutionary Computations will confuse the human operator even if he/she is expert of domain tasks.

In this paper, a novel problem class on dynamic/uncertain environments for Evolutionary Computation is introduced, i.e., robust solutions on Time-Varying function. In this problem class, instead of tracking optimal peaks, a static solution, which may not be optimal but performs well at any time steps, is provided by Evolutionary Computations. In order to acquire such solutions, three fitness functions are introduced in this paper: Naive Evaluation, Exponential-Weighted Fitness, and Linear-Weighted Fitness. Experiments on modified Branke's benchmark problems of Time-Varying function confirm the effectiveness of the weighted approaches.

This paper is organized as follows: The next section briefly introduces two topics of Evolutionary Computation studies in Dynamic/Uncertain Environments: Robustness and Time-Varying Function. Section 3 introduces robust solutions on Time-Varying function and fitness function for them. Experimentations on Branke's benchmark problems are examined in section 4. Next, future works are summarized by referring to related works. Finally, this paper is concluded.

## 2. ECS IN DYNAMIC/UNCERTAIN ENVIRONMENTS

In a recent survey paper [6], Evolutionary Computation in dynamic/uncertain environments were summarized into four categories: Noise, Robustness, Fitness approximation, and Time-varying fitness function. This section briefly introduces two topics, i.e., Robustness and Time-varying fitness function, which are relevant with this paper.

Robust solutions are needed if design variables are affected by perturbations. In order to find out robust solutions, Evolutionary Computations should work on expected

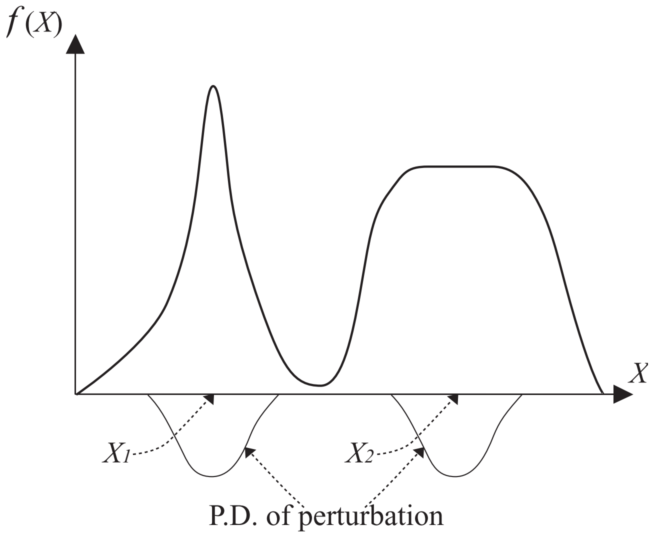

**Figure 1: One dimensional fitness function with two peaks.**

fitness function, called Effective Fitness Function in [8][9].

$$F(X) = \int_{-\infty}^{\infty} f(X+\delta)p(\delta)d\delta, \tag{1}$$

where $p(\delta)$ indicates the probability distribution of possible perturbations $\delta$. Suppose that there is a one dimensional fitness function with two peaks, i.e., high-narrower one and low-broader one, as depicted in Figure 1. In addition, the probability distribution of possible perturbations to the design variable $X$ is represented by normal distributions as in the figure. The effective fitness $F(X_1)$ at $X_1$ in the high-narrower peak is smaller than the one at $X_2$ in the lower-broader peak. Thus, Evolutionary Computations for robust solutions are designed to find out better stable-solutions against disturbances.

Time-varying fitness functions are regarded as a deterministic function at any time steps. However, the landscape of the time-varying fitness functions is temporally changed. Therefore, the peak positions on the time-varying fitness function is moved over time. Figure 2 illustrates an example of the movement of peak positions on time-varying functions. In general, the purpose of Evolutionary Computation in time-varying functions is to develop new algorithms which can rapidly adapt to environmental changes.

## 3. ROBUST SOLUTIONS FOR TIME VARYING FUNCTIONS

### 3.1 Motivation

Because of the population search of Evolutionary Computations, ECs can adapt to dynamic environments if populations can maintain the genetic diversity. Hence, such adaptation is one of key features of Evolutionary Computations. This paper, however, does not pursue the tracking property of Evolutionary Computations, but provides fitness functions for robust solutions on time-varying functions. Robust solutions in this paper are considered as a tuple of design variable values, which may not be optimum

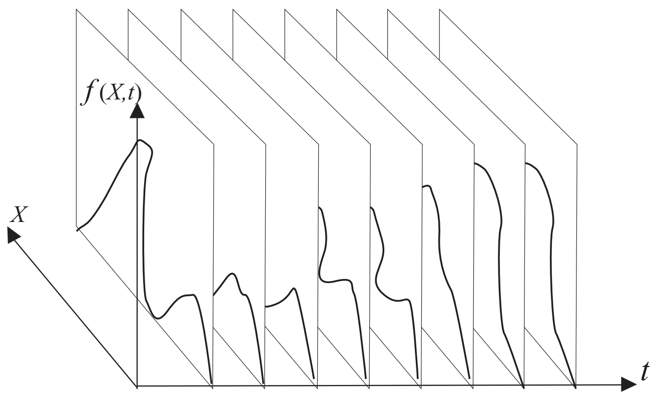

**Figure 2: Temporal changes of peak positions on time-varying functions.**

solution at each time step, but which works well against the changes of fitness landscape over time. It does not always find out effective robust solutions for any time-varying problems since it is strongly depending on how fitness landscape would be changed temporally. However, we conjecture that such robust solutions can be found out if design variable values with better fitness is not widely shifted so much over temporal changes of fitness landscape.

Problems, where we assume robust solutions in this paper are needed, are 1) that problems are formulated by time-varying functions and 2) that design variable values are used by human operators. That is, we assume that the human operators shall achieve their tasks by referring to the resultant design variable values acquired by Evolutionary Computations. Although simple function optimization problems are employed as test bed problems in this paper, we can exemplify some practical problems where robust solutions in this paper are required: Practical vehicle routing problems for garbage collection, products delivery, and so on. These problems basically cope with the same area everyday so that the road network in the area rarely changed. However, demands from customers or clients are daily changed. If we optimized vehicle routes everyday, truck drivers would daily change their routes. As a consequence of route changes, the truck drivers may be confused. By using robust solutions in this paper, the truck drivers can fix their routes with better performance although the fixed routes may be different with optimal routes at that day.

### 3.2 Formalization

Suppose that time-varying function is denoted as $f(X,t)$, where $X$ is a vector of design variables and $t$ denotes time. Robust solutions on time-varying functions may be optimal solutions of $F(X)$:

$$F(X) = \int f(X,t)dt. \tag{2}$$

If time-varying functions are defined on discrete time steps, the function $F(X)$ is rewritten as follows:

$$F(X) = \sum_i f(X,t_i), \tag{3}$$

where $t_i$ is the $i^{\text{th}}$ time steps. The above equation can also be used if the calculation of the integral is expensive in the

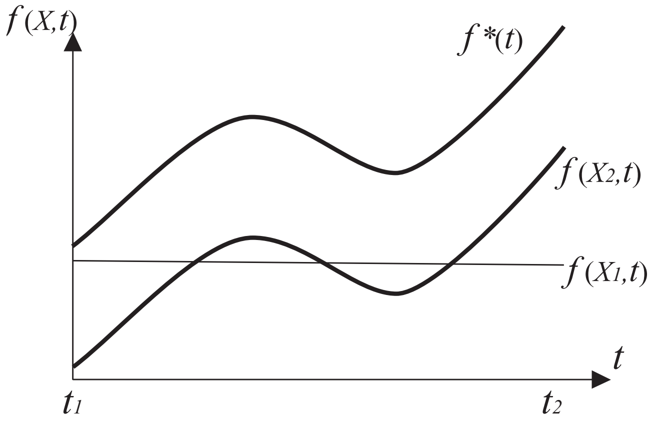

**Figure 3: Evaluation of solutions $X_1$ and $X_2$ on Time-varying function.**

sense of computational efforts. That is, problem instances $f(X, t_i)$ at several time steps $t_i$ which indicate typical characteristics of $f(X, t)$ is collected in advance. Instead of the integral calculation, $F(X)$ is approximated by summing up the typical problem instances $f(X, t_i)$.

Such naive integral/sum forms may often cause a certain problem as explained in Figure 3: Suppose that $f(X, t)$ is a function to be maximized. $f^*(t)$ is optimal value or upper bound at time $t$. By using equation (2), fitness evaluations for solutions $X_1$ and $X_2$ are the same evaluation. As you can see from this figure, however, the solution $X_1$ works well around time $t_1$ but shows poor performance around time $t_2$ in comparison with corresponding optimal value. On the other hand, the solution $X_2$ exhibits marginal and stable performance in the whole period in the figure. Therefore normalization is needed for realizing robust solution on time-varying function. The normalized function $f_N(X, t)$ is defined as follows:

$$f_N(X, t) = \frac{f^*(t) - f(X, t)}{f^*(t)}. \tag{4}$$

This paper treats time-varying function on discrete time steps so that the following function is used to evaluate robustness solutions:

$$F(X) = \sum_i f_N(X, t_i). \tag{5}$$

## 3.3 Weighted Fitness Function

This paper introduces three kinds of fitness functions for finding out robust solutions: Naive Evaluation, Exponential-Weighted Fitness, and Linear-Weighted Fitness. The Naive Evaluation uses equation (5) as fitness function. For two weighted fitness evaluation, the following equation is adopted:

$$F(X) = \sum_i^N w_i f_N(X, t_i), \tag{6}$$

where $w_i$ is weight value for the $i^{\text{th}}$ time step $t_i$ ($0 \le w_i \le 1$ for all i, and $\sum_i^N w_i = 1$), and $N$ indicates the number of time steps. The weights $w_i$ are updated every predefined interval $l$ and is initially set to be $1/N$. The difference between these fitness is updating method:

**Exponential-Weighted Fitness** Let $f_N^b(X, t_i)$ the best evaluation of function $f_N(X, t_i)$ after the previous weight change. In the Exponential-Weighted Fitness, the weights are updated in the same formula of Boltzmann equation:

$$w_i = \frac{\exp f_N^b(X, t_i)}{\sum_{j=1} \exp f_N^b(X, t_j)} \tag{7}$$

**Linear-Weighted Fitness** In the case of Linear-Weighted Fitness, the weights $w_i$ are changed by the following equation:

$$w_i = \frac{f_N^b(X, t_i)}{\sum_{j=1} f_N^b(X, t_j)} \tag{8}$$

## 4. EXPERIMENTS

### 4.1 Test Problems

Branke's test function [2] is used in this paper: This 5-dimensional function with 5 peaks is represented by the following equation,

$$f(X, t) = \max_{i=1\ldots5} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{5} (x_j - P_{ij}(t))^2}, \tag{9}$$

where $P_{ij}$ denotes the $j^{\text{th}}$ coordinate of the $i^{\text{th}}$ peak at time step $t$. $H_i(t)$ and $W_i(t)$ indicate the height and the width of $i^{\text{th}}$ peak at time step $t$, respectively.

The coordinates $P_i(t)$, the height $H_i(t)$, and the width $W_i(t)$ are temporally changed by the following formula:

$$\begin{aligned} \sigma &\in N(0, 1) \\ H_i(t) &= H_i(t-1) + 7\sigma \\ W_i(t) &= W_i(t-1) + 0.01\sigma \\ P_i(t) &= P_i(t-1) + V \end{aligned}$$

$V$ is a vector of fixed length one in a random direction.

Range of each design variable $x_i$ is set to $0 \le x_i \le 100$. In original Branke's test function, range of each coordinate of peaks is also set to be the same range as design variable. In this paper, range of all the coordinates of peaks is set to $20 \le P_{ij}(t) \le 50$ in order to attract the peak positions within a certain area. Initial values of the coordinates $P_i(0)$, the height $H_i(0)$, and the width $W_i(0)$ are slightly changed from Branke's paper, due to the modification of peak positions' range, as described in Table 1.

From the same initial problem parameters, 10 sequences of problem instances are generated with the different random seeds. As depicted in Figure 4, each sequence consists of 150 successive problem instances $f(X, t_i)$ at each time step: the first 100 problem instances $f(X, t_i)$ are for evolution data set and the subsequent 50 problem instances $f(X, t_i)$ are for test data set.

### 4.2 Fast Evolutionary Programming

Fast Evolutionary Programming proposed by Xin *et al.* [12] is used in this paper because the Fast Evolutionary Programming is easy to implement and performs well due to the Cauchy distribution mutation. Although other Evolutionary Computations have not been examined yet, it is considered that the results acquired in this paper can be applied to the other algorithms.

**Table 1: Initial value of $P_i(0)$, $H_i(0)$, and $W_i(0)$.**

| | $P_{i1}$ | $P_{i2}$ | $P_{i3}$ | $P_{i4}$ | $P_{i5}$ | $W_i$ | $H_i$ |
|---|---|---|---|---|---|---|---|
| peak1 | 22.0 | 41.0 | 42.0 | 38.0 | 21.0 | 0.1 | 50.0 |
| peak2 | 45.0 | 24.0 | 45.0 | 25.0 | 22.0 | 0.1 | 50.0 |
| peak3 | 23.0 | 26.0 | 29.0 | 42.0 | 28.0 | 0.1 | 50.0 |
| peak4 | 42.0 | 49.0 | 41.0 | 28.0 | 34.0 | 0.1 | 50.0 |
| peak5 | 45.0 | 30.0 | 34.0 | 38.0 | 41.0 | 0.1 | 50.0 |

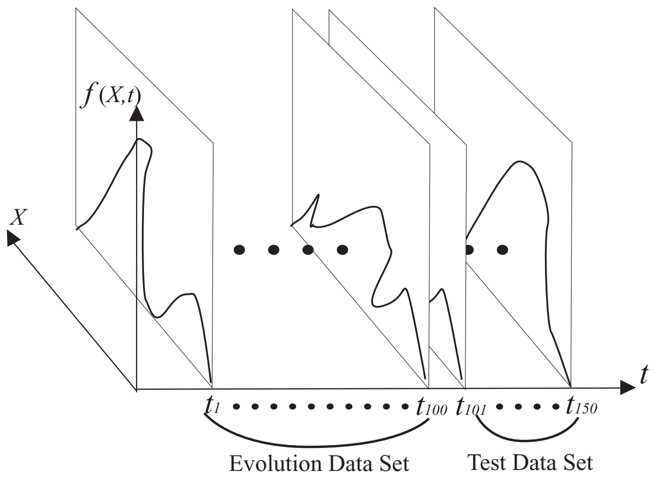

**Figure 4: Generation of an Evolution data set and a Test data set**

Individuals in Evolutionary Programming are composed of a pair of real valued vectors $(X, \eta)$, where $X$ and $\eta$ indicate design variables in given problems and variance parameter used in self-adaptive mutation [1]. The Fast Evolutionary Programming is carried out in following steps [12]:

1. Generate the initialize population of $\mu$ individuals
2. Evaluate individuals
3. Each individual (parent) create single offspring $X', \eta'$ by using the following equation,

$$\begin{aligned} x'_j &= x_j + \eta_j \delta_j, \\ \eta_j &= \eta_j \exp(\tau' N(0,1) + \tau N_j(0,1)), \end{aligned}$$

   where $\eta_j$ denotes $j^{\text{th}}$ component of vector $\eta$. $\delta_j$, $N(0,1)$, and $N_j(0,1)$ denote a Cauchy random variable, a normal distribution random number with mean zero and standard deviation one used in all the design variables, and a normal distribution random number used in a design variable $j$, respectively. Moreover, coefficient $\tau$ and $\tau'$ are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively.

4. Evaluate offsprings
5. Conduct pairwise comparison over parents and offsprings. For each individual, $q$ opponents are chosen uniformly at random. For each comparison, the individual receives "win" if its fitness is not smaller than opponent's one.
6. Select the $\mu$ individuals, which have the most wins, from parents and offsprings
7. Stop if halting criterion is satisfied. Otherwise go to Step 3.

## 4.3 Experimental Results

### *4.3.1 Results on Evolution Data Set*

Experiments in this subsection are carried out as follows: First, the Fast Evolutionary Programmings with three fitness functions, i.e., Naive Evaluation, Exponential-Weighted Fitness, and Linear-Weighted Fitness, are executed. At that time, 100 problem instances $f(X, t_i), (1 \leq i \leq 100)$ in the Evolution data set are used in fitness evaluation. We examined 10 sequences mentioned in section 4.1. For each sequence, 50 runs are carried out with different random seed. Next, the best individual acquired by the Fast Evolutionary Programming for each execution is re-evaluated by using equation (5). Some parameters for running algorithms are described as follows: population size $\mu$ is set to 100. the interval length of changing weight in the proposed method is set to 200. The number of generations for each run is set to 5,000.

Table 2 shows re-evaluation results of Fast Evolutionary Programmings with three fitness functions described in section 3.3: Naive Evaluation, Exponential-Weighted Fitness, and Linear-Weighted Fitness. All the numbers at "Naive Evaluation" column and upper numbers at "Exponential-Weight" and "Linear-Weight" columns denote the average evaluation of the best individual. Lower numbers at these columns indicate p-values of t-test. The italic numbers mean that corresponding experiments exhibits worse performances than the result of Naive Evaluation, which show a statistically significant difference. The reason why the Naive Evaluation performs well on evolution data set is that the weighted approaches proposed in this paper did not refer equation (5) during evolution.

Changes of weight values for Exponential Weight and Linear Weight during evolution are plotted in Figure 5. This result is a typical one. Darker colors in both graphs denote less weighted instances. Weights in the Exponential Weight fitness are bigger than ones in the Linear Weight fitness. Moreover, early generations, around problem instance $f(X, t_{40})$ are darken. After 1000 generations, that area is slightly lightened. Although it is impossible to see from these figures, weights in the Linear Weight fitness are oscillated much than ones in the Exponential Weight fitness in fact. The maximum and minimum weights for the Exponential Weight in Figure 5 are 0.0144 and 0.00697, respectively. On the other hand, the maximum and minimum weights for the Linear Weight are 0.0219 and 0.0012, respectively. Therefore, it is considered that the Linear Weight tries to search for solutions which indicate better performance at difficult problem instances.

### *4.3.2 Results on Test Data Set*

The best individuals in the previous subsection are examined on the test data set, by using equation (5). We mention again that the successive 50 problem instances $f(X, t_i)(101 \leq i \leq 150)$ are not used in evolution phase. They are just used to investigate the generalization property in the sense of inductive learning.

Table 3 summarizes experimental results on the test data set. Numbers in the table are in the same format as in Table 2, except that results in this table are examined on

**Table 2: Experimental results on Evolution Data Set: average evaluation of the best individual by each fitness evaluation methods; lower numbers at Exponential Weight column and at Linear Weight column denote p-value of T-test against the Naive Evaluations**

| | Naive Evaluation | Exponential-Weight | Linear-Weight |
|---|---|---|---|
| Sequence 1 | 54.77868 | 54.76463<br>0.9574 | 54.73118<br>0.8526 |
| Sequence 2 | 42.70853 | 42.82250<br>0.2194 | *43.17296*<br>4.246e-05 |
| Sequence 3 | 42.47384 | *42.50562*<br>2.2e-16 | *47.44566*<br>2.2e-16 |
| Sequence 4 | 22.95461 | *22.99131*<br>2.2e-16 | *23.56934*<br>2.2e-16 |
| Sequence 5 | 34.04960 | *34.06635*<br>2.2e-16 | *34.15811*<br>2.2e-16 |
| Sequence 6 | 51.15507 | *51.25281*<br>2.2e-16 | *51.39275*<br>2.2e-16 |
| Sequence 7 | 28.80670 | *28.94273*<br>2.2e-16 | *29.41778*<br>2.2e-16 |
| Sequence 8 | 23.82788 | *23.82974*<br>2.2e-16 | *23.86324*<br>2.2e-16 |
| Sequence 9 | 47.55176 | *47.57253*<br>2.2e-16 | *47.85345*<br>2.2e-16 |
| Sequence 10 | 43.17423 | 43.60613<br>0.6793 | 43.63554<br>0.6587 |

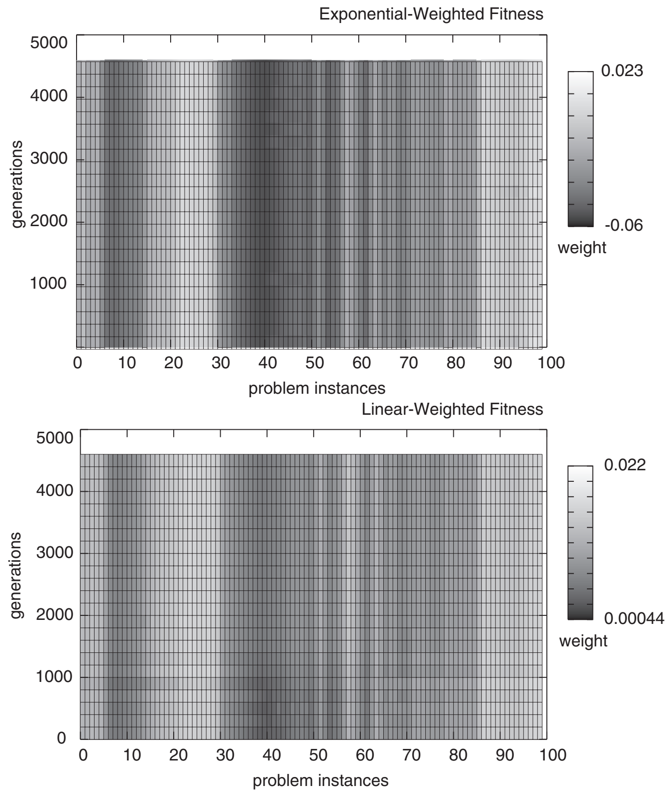

**Figure 5: Changes of weights: Exponential Weight Fitness (UPPER), Linear Weight Fitness (LOWER)**

**Table 3: Experimental results on Test Data Set: Each number is the same format as in Table 2**

| | Naive Evaluation | Exponential-Weight | Linear-Weight |
|---|---|---|---|
| Sequence 1 | 49.21772 | 49.49733<br>0.5074 | 49.67809<br>0.2164 |
| Sequence 2 | 32.99147 | 32.44501<br>0.6012 | 31.68971<br>0.1855 |
| Sequence 3 | 35.24063 | **35.05889**<br>2.2e-16 | **26.24324**<br>2.2e-16 |
| Sequence 4 | 19.52107 | **19.42020**<br>2.2e-16 | **19.09799**<br>2.2e-16 |
| Sequence 5 | 31.29067 | **31.08309**<br>2.2e-16 | **30.88144**<br>2.2e-16 |
| Sequence 6 | 33.81956 | *34.09091*<br>2.2e-16 | *33.99676*<br>2.2e-16 |
| Sequence 7 | 29.85894 | **28.92194**<br>2.2e-16 | **28.51422**<br>2.2e-16 |
| Sequence 8 | 16.83420 | *16.87692*<br>2.2e-16 | *17.02256*<br>2.2e-16 |
| Sequence 9 | 26.36128 | **26.12703**<br>2.2e-16 | *27.57804*<br>2.2e-16 |
| Sequence 10 | 40.13892 | *40.90530*<br>0.001042 | **39.30670**<br>0.001103 |

the test data set. The numbers with bold face font denotes that corresponding result shows better performance than the result of the Naive Evaluation, which show a statistically significant difference. As you can see this table, the weighted approaches perform statistically well in half of sequences. That is, the weighted methods have better generalization property.

### *4.3.3 Relationship with Peak Position Distributions*

As mentioned in section 4.1, the peak positions' range is restricted in a certain region. In this section, for some axes, the range of peak positions is set to original settings in the Branke's paper [2]: For selected axe $j$, $0 \leq P_{ij} \leq 100, (i = 1, \ldots, 5)$.

Table 4 describes the experimental results on Test Data Set (sequence 3) for various numbers of restricted axes. In Similar with experiments in the previous subsection, this test data set is not appeared in evolution phase. Peak positions are distributed in the whole design variable space over time if the number of restricted axes is small. As described in this table, it is quite difficult to find out robust solutions if peak positions are distributed.

## 5. RELATED WORKS

This section refers related works, and describes future works. The notion of Robust solutions on time-varying functions was proposed in this paper. Theoretical studies [4][5][9] in conventional robust solutions might be useful for this paper.

Tradeoff between the robustness and the optimality of solutions is also an important topic for the robust solutions on time-varying functions. As in Jin's paper [7], multi-objective approach of a robustness measure and fitness might be effective for the robust solutions on time-varying functions.

In order to find out simplified solutions for human op-

**Table 4: Experimental results on Test Data Set: Changes of peak positions' range restriction: Each number is the same format as in Table 2**

| No. restricted axes | Naive Evaluation | Exponential-Weight | Linear-Weight |
|---|---|---|---|
| 0 | 35.09107 | 34.99220<br>0.8282 | 35.01544<br>0.8672 |
| 1 | 35.13044 | 35.02153<br>0.8109 | 35.11955<br>0.9807 |
| 2 | 36.03343 | **34.99138**<br>0.001575 | **32.75871**<br>0.006888 |
| 3 | 35.08048 | 35.03912<br>0.8976 | **25.06468**<br>2.2e-16 |
| 4 | 35.13194 | **34.97776**<br>2.2e-16 | **25.36159**<br>2.2e-16 |
| 5 | 35.24063 | **35.05889**<br>2.2e-16 | **26.24324**<br>2.2e-16 |

erators, the robust solutions on time-varying functions are introduced in this paper, where only one solution for all the time steps is found out. However, a few solutions which divides time-domain may be acceptable for human operators. Multi-populations [3][10] or multi-ploid can be used to find out such solutions. At that time, tradeoff between the complexity and the quality of solutions must be an important notion.

Finally, other benchmark problems on dynamic environments [11] should be examined in order to investigate the effectiveness of weighted approaches.

## 6. CONCLUSIONS

In this paper, robust solution on Time-Varying functions was introduced. As described in section 3.1, the notion of robust solutions on Time-Varying functions plays crucial role if human operators shall employ resultant solutions by Evolutionary Computations: If Evolutionary Computations tracked new peak points after environmental changes and provided new solutions at each time steps, the human operators would be confused by dynamic solutions.

In order to find out robust solutions, two weighted fitness functions, i.e., Exponential Weight and Linear Weight, were proposed. Experiments on modified Branke's benchmark problems of Time-Varying function are carried out in subsections 4.3.1 and 4.3.2. The experiments reveal that the weighted fitness functions yields more "robust" solutions, which show better generalization property. As indicated in subsection 4.3.3, it is difficult to find out robust solutions if peak positions are distributed over the whole area of design variable space.

## 7. REFERENCES

[1] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[2] J. Branke. Memory-enhanced evolutionary algorithms for dynamic optimization problems. In *Proc. Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 1875–1882, 1999.

[3] J. Branke, T. Kausler, C. Schmidt, and H. Schmeck. A multipopulation approach to dynamic optimization problems. In *Proc. Adaptive Computing in Design and Manufacturing 2000*, 2000.

[4] U. H. D. Wiesmann and T. Bäck. Robust design of multilayer optical coatings by means of evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 2(4):162–167, 1998.

[5] I. Das. Robustness optimization for constrained nonlinear programming problem. *Engineering Optimization*, 32(5):585–618, 2000.

[6] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Trans. Evolutionary Computation*, 9(3):303–317, 2005.

[7] Y. Jin and B. Sendhoff. Tradeoff between performance and robustness: An evolutionary multiobjective approach. In *Proc. Evolutionary Multi-Criterion Optimization, LNCS 2632*, pages 237–251, 2003.

[8] S. Tsutsui and A. Ghosh. A robust solution searching scheme in genetic search. In *Parallel Problem Solving from Nature IV*, pages 543–552, 1996.

[9] S. Tsutsui and A. Ghosh. Genetic algorithms with a robust solution searching scheme. *IEEE Trans. Evolutionary Computation*, 1(3):201–208, 1997.

[10] R. K. Ursem. Multinational ga optimization techniques in dynamic environments. In *Proc. Genetic and Evolutionary Computation Conference*, pages 19–26, 2000.

[11] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proc. Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 2039–2046, 1999.

[12] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Trans. Evolutionary Computation*, 3(2):82–102, 1999.