New Performance Indicators for Robust Optimization Over Time

Yuanjun Huang[1, 3], Yaochu Jin[1,2, 4*]
[1] Engineering Research Center of Digitized Textile & Fashion Technology, Ministry of Education, Shanghai 201620, P. R. China
[2] College of Information Sciences and Technology, Donghua University, Shanghai 201620, P. R. China
Email: Huangyuanjungege@126.com /*yaochu.jin@surrey.ac.uk

Yongsheng Ding[1, 2*]
[3] Jiaxing Vocatonnal College, Zhejiang 314036, P. R. China
[4] Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom
Email: *ysding@dhu.edu.cn

***Abstract* —A new approach to solving dynamic optimization problems (DOPs), called robust optimization over time (ROOT) has recently been suggested and a number of optimization algorithms for finding ROOT solutions have been developed. These algorithms typically rely on the predicted fitness in the new environment to determine the robustness of a solution, which may be considerably different from the true robustness due to the errors in fitness prediction. Consequently, performance comparison based on the predicted fitness may be also misleading. In this paper, we propose a new formulation of the performance measures for quantitatively evaluate the performance of optimization algorithms for finding robust solutions over time, assuming that the actual optimal solution of the problem is known. We empirically verify the proposed performance measures by analyzing the robust optimal solutions obtained in two recently reported ROOT evolutionary algorithms. Our results indicate that the proposed performance measures are able to provide an effective means for performance comparison of ROOT algorithms.**

Keywords— dynamic environment; performance indicators; robust optimization, robustness over time

I. INTRODUCTION

Solving dynamic optimization problems (DOPs) has attracted increasing attention in the evolutionary optimization community [1-8]. Most of the research on DOPs has focused on the tracking moving optimum (TMO) as fast and as accurate as possible [7-10]. In other words, TMO algorithms aim to find the global optimum of a DOP at any point instant as soon as the environment is changed. However, the TMO approach may suffer from serious limitations in solving some real-world problems [11-14]. On one hand, tracking a moving optimum requires the optimization algorithm to be able to converge very quickly to the new optimum, which may be achieved at the cost of its explorative capability. On the other hand, it is impractical to frequently change the optimal solution to be implemented as change of the solution may be very costly.

To address the above concerns, Yu and Jin [15] have recently put forward a different approach to solving DOPs which aims to find optimal solutions that are robust over time. A solution is considered to be robust over time (ROOT) if its performance remains to be acceptable by the user over a certain period of time, even if there are environmental changes. In other words, this solution will continue to be used even if it is no longer an optimal solution due to environmental change. A new optimal solution will be sought once the performance of the current solution becomes worse than a predefined threshold. In addition, a ROOT algorithms does not target for the global optimum at the current time instant; instead, it searches for an optimal solution that is expected to be able to satisfy the required quality as long as possible. In this way, the frequency for changing the solution to be implemented is minimized, while the performance of the solution is maximized. From the above, we can see that the idea behind ROOT is considerably different from DTO that aims to locate the global optimum at any time instant in the presence of environmental changes A few ROOT algorithms have been proposed recently [16-19]. Jin et al [16] proposed a generic framework for finding robust optimization over time based on the average and variance of the performance over a given period of time. It has also been revealed that there is a trade-off between the average performance and the variance of the performance if multiple ROOT solutions exist Fu et al [17] suggested two slightly different measures for characterizing the robustness of solutions in a changing environment, namely the average fitness and the number of time instants a solution remains to be acceptable, based on which a few new variants of ROOT were developed. The proposed algorithms were tested on several benchmark DOPs, showing that the ROOT algorithms are able to find optimal solutions different from those found by traditional DTO algorithms.

However, in the above algorithms, the robustness of a solution over time is estimated based on the predicted fitness in new environments, which may differs with the true robustness of the solution due to the error in fitness estimation. In [15] various algorithms were compared based on average robustness of the solutions obtained by the compared algorithms. Such comparisons may be insufficient in evaluating the effectiveness of the ROOT algorithms due to the following reasons. Firstly, it is unclear if the solutions achieved by a ROOT algorithm is the actual ROOT solution of the problem and if not, what is the difference in the robustness over time between solutions obtained by a ROOT algorithm and the true ROOT solution. Secondly, in case the environment changes severely, no ROOT solution may exist for the given problem in some time instants. Thus, it is more reasonable to exclude these situations in comparing the performance of ROOT algorithms. Thirdly, it might be impractical to expect that a ROOT algorithm can always

978-1-4799-7492-4/15/$31.00 ©2015 IEEE

locate the ROOT solution in all environments. Thus, it is of great interest to see in how many time instants, the ROOT algorithms have been succeeded in locating the true ROOT solution, where the success rate becomes one of the most important performance indicators of the compared algorithms.

In this paper, we point out the weakness of performance measures suggested in [18] and put forward a performance indicator, termed *error* that quantitatively measures the difference between the robustness value of the solutions obtained by a ROOT algorithm and that of the true ROOT solutions. We also define a *success rate* to judge how often an algorithm is able to achieve the true ROOT solutions in various environments. Finally, we empirically verify the effectiveness of the suggested performance indicators by comparing two algorithms, one proposed by Jin et al in [16] and the other by Fu et al in [17]. Our results indicate that the *error* measure can clearly indicate the performance of the solution obtained by the algorithms in each environment, whereas the *success rate* can effectively compare the overall performance of the ROOT algorithm.

The remainder of paper is organized as follows. A brief overview of related work is given in Section II. Section III presents the proposed performance indicators for assessing the performance of ROOT algorithms. Experimental settings are described in Section VI. Empirical results on comparing two ROOT algorithms are presented and discussed in Section V. Section VI summarizes paper and suggests some future work.

II. RELATED WORK

A. Definition of ROOT

Without the loss of generality, we consider the following dynamic maximization problem:

$$\max F(\vec{\mathrm{X}}) = f(\vec{\mathrm{X}}, \vec{\alpha}(t)) \tag{1}$$

where f is objective function to be maximized, $\vec{X}$ represents the decision variables, $\vec{\alpha}(t)$ is a vector of environmental parameters that change over time. $\vec{\alpha}(t)$ can be continuous or discrete changes, which are supposed to remain constant for a period of time and then change at certain time points. This means that the dynamic optimization problem can be represented as a sequence of stationary fitness functions

$$\left(f(\vec{\mathrm{X}}, \vec{\alpha}_1), f(\vec{\mathrm{X}}, \vec{\alpha}_2), \cdots, f(\vec{\mathrm{X}}, \vec{\alpha}_l)\right) \tag{2}$$

where, $\vec{\alpha}_i$, $i = 1, 2, \cdots l$ are constants. The TMO aims to find the global optimal solution $\vec{\mathrm{X}}_i^*$ of the problem $f(\vec{\mathrm{X}}, \vec{\alpha}_i)$, where $i = 1, 2, \cdots l$. By contrast, the ROOT approach intends to find an optimal solution $\vec{\mathrm{X}}_i$ of problem $f(\vec{\mathrm{X}}, \vec{\alpha}_i)$, that has acceptable performance in multiple environments. More specifically, in the ROOT approach, a solution is said to be robust over time if its performance is acceptable for at least two consecutive time intervals (environments).Therefore, when the environment changes, the existing solution does not need to be changed.

B. Generic Framework for Finding ROOT Solutions

A framework for finding ROOT solutions was proposed in [16], which contains a solver, a database, an approximator and a predictor. The basic idea is that the solver searches for solutions on the basis of a performance measure as described in Equation (3), which is the average of the fitness in the past, the current fitness and future fitness. That is to say, at time step l_0, we estimate the robustness of the solution $\vec{\mathrm{X}}$ as follows

$$\hat{F}(\vec{\mathrm{X}}, l_0) = \sum_{l=l_0-p}^{l_0+q} \hat{f}(\vec{\mathrm{X}}, \vec{\alpha}_l) \tag{3}$$

where p, q control the number of past and future time steps that influence the robustness of the solution. Note that in calculating the robustness, the approximated fitness $\hat{f}$ replaces the real fitness values in order not to introduce extra fitness evaluations. The specific flow chart for finding ROOT solutions based on predicted fitness is shown in Fig. 1.

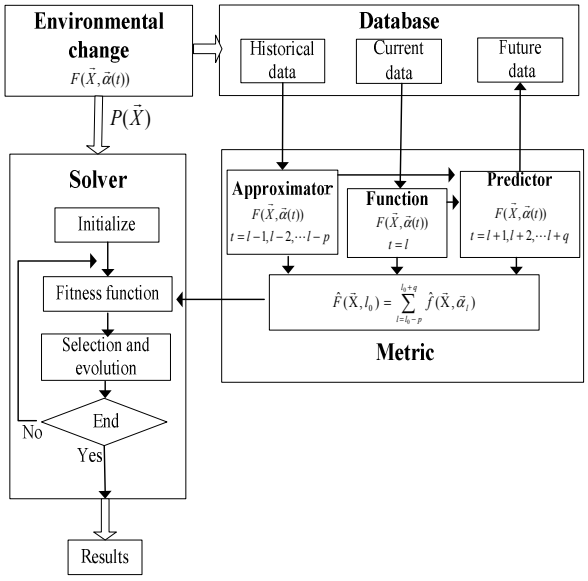

Fig. 1 The flow chart for finding ROOT solution.

This frame structure for searching for ROOT solutions is very generic yet intuitive. However, as pointed out in Fu *et al* [18], the adopted performance indicator in Equation (3) only takes into account the average fitness and does not impose any constraints on the acceptable worst fitness. In addition, the performance indicator in Equation (3) does not take into account the errors in estimating the past fitness and predicting the future fitness. To address these concerns, new performance measures have been suggested in [18], which will be discussed in the next section.

III. PERFORMANCE INDICATORS FOR FINDING AND ASSESSING ROOT SOLUTIONS

In this section, we discussion three related but different concepts: The robustness definition from the point of view of ROOT, the metrics for finding ROOT solutions and performance indicators for assessing the performance of ROOT algorithms.

A. Robustness definitions

Fu et al suggested two definitions for quantitatively measuring robustness over time [18].

1) Survival time:

$$R^s(\vec{X},t,\delta)=\max\left\{0\cup\left\{l\middle|F_i(\vec{X})\geq\delta,\forall i,t\leq i\leq t+l\right\}\right\} \tag{4}$$

where R^s are defined as the maximum time interval starting from time t during which the fitness of solution $\vec{X}$ keeps larger (better) than the pre-defined fitness threshold δ.

2) Average fitness R^a : Starting from time t, and given a pre-defined time windows T, the robustness R^a of solution $\vec{X}$ is defined as follows:

$$R^a(\vec{X},t,T)=\frac{1}{T}\sum_{i=t}^{t+T-1}F_i(\vec{X}) \tag{5}$$

B. Metric for finding ROOT solutions

For both robustness definitions, Fu et al also proposed two variants of metrics for finding robust solutions, one considering the accuracy of the predictor and the other not.

*1) Metrics for Survival Time:*The metric for robustness definition: *survival time*

$$\hat{R}^s(\vec{X},t,\delta)=\begin{cases}0 & \text{if } F_t(\vec{X})\leq\delta\\ w\cdot\hat{l} & \text{otherwise}\end{cases} \tag{6}$$

where $\hat{l}$ is used to represent the number of consecutive fitness which is not smaller than δ starting from the beginning of the fitness sequence $\left(\hat{F}_{t+1}(\vec{X})\cdots\hat{F}_{t+i}(\vec{X})\cdots\hat{F}_{t+l}(\vec{X})\right)$, w is the coefficient associated with the accuracy of the predictor. The root means square error E^{err} is employed as the accuracy measure:

$$E^{err}=\sqrt{\frac{\sum_{i=1}^{n_t}e_i^2}{n_t}} \tag{7}$$

where n_t is the number of samples, and e_i is the absolute difference between the estimated values produced by the predictor and the true values for the i -th sample point, refer to [18] for details. In order to make sure that a larger weight is assigned when the corresponding estimator is considered to be more accurate, w takes the following exponential function of E^{err} :

$$w=\mathrm{e}^{(-\theta\cdot E^{err})} \tag{8}$$

where $\theta\in[0,\infty)$ is a control parameter.

*2) Metric for average fitness:*The metric for robustness definition: *average fitness*

$$\hat{R}^a(\vec{X},t,T)=F_t(\vec{X})+\sum_{i=1}^{T-1}(\hat{F}_{t+i}(\vec{X})-\theta\cdot E^{err}) \tag{9}$$

where θ and E^{err} are the same defined in Equation (8).

3) Metric for finding actual robust values: In order to get the actual robust values, we use Equation (4) as the actual survival time and Equation (5) as the actual average fitness to evaluate the solutions by the solver in finding ROOT solutions.

Note that in the Equations (6), (9) $\hat{F}_{t+i}(\vec{X})$ is predicted, while in Equations (4), (5) $F_i(\vec{X})$ is a time-varying objective function. Note also that in practice, the future objective function is unknown due to its changing nature. However, as this paper focuses on performance evaluation using test problems, we assume that the dynamic objective function is known.

C. Proposed performance indicators

As a ROOT optimization problem can be represented as a sequence of stationary optimization problems $(F_1,F_2,\cdots,F_N)$ during the time interval $[t_0,t_{end})$. In [18] the following performance indicator has been suggested:

$$P=\frac{1}{N}\sum_{i=1}^{N}R_i \tag{10}$$

where R_i is the robustness of the solution determined by the algorithm in the i -th environment, N is the total number of environments. However, as discussed in the Introduction, the performance indicator in Equation (10) cannot fully correctly characterize the quality of the ROOT solutions found by a ROOT algorithm. To address the weakness of the performance indicator in Equation (10), we define the following *error* between robustness calculated by the algorithm during the search, which is based on the estimated future fitness, and the actual robustness value:

$$e=\left|\hat{R}-R\right| \tag{11}$$

where, R is the actual robustness of the test problem and $\hat{R}$ is the robustness of the solutions achieved by a ROOT algorithm.

In addition, we define a *success* in locating the ROOT solution by an optimization algorithm: as follow, which is also used to evaluate the performance of the algorithm:

$$S=\begin{cases}1 & \text{if } \left(1-\left|\frac{e}{R}\right|\geq\varepsilon\text{ , }R\geq 1\text{for }R^s,R\neq 0\text{ for }R^a\right)\\ 0 & \text{else}\end{cases} \tag{12}$$

where $S=1$ denotes a *success* and $S=0$ a failure, $\varepsilon\in(0,1]$ is a pre-defined success threshold. $R\geq 1$ is a constraint when *survival time* R^s is adopted for measuring robustness, and R^a, $R\neq 0$ is a constraint when *average fitness* R^a is adopted as robustness, respectively.

Finally, we define the *success rate* σ as follows to measure the overall performance of an algorithm:

$$\sigma=\frac{1}{N}\sum_{i=1}^{N}S_i\times 100\% \tag{13}$$

To summarize, we propose three performance indicators for assessing the performance of ROOT algorithms. e in Equation (11) can be used to quantitatively measure the difference in robustness between the solution obtained by a ROOT algorithm and the actual. S defined in Equation (12) can be used to judge whether the algorithm succeeds in identifying the ROOT solution at a time instant. Last but not the least, σ in Equation (13) can be used to measure the overall performance of the algorithm averaged over N different environments.

IV. EXPERIMENTAL SETUP

A. Experimental Settings

Two sets of experiments have been conducted on the modified Moving Peaks Benchmark (mMPB) [15,18,20].Test Problem1 (TP1) is described as follows:

$$
\begin{aligned}
F_t(\vec{X}) &= \max_{i=1}^{i=m}\left\{H_t^i - W_t^i \cdot \| \vec{X} - \vec{C}_t^i \|\right\} \\
H_{t+1}^i &= H_t^i + height\_severity^i \cdot N(0,1) \\
W_{t+1}^i &= W_t^i + width\_severity^i \cdot N(0,1) \\
C_{t+1}^i &= C_t^i + \vec{v}_{t+1}^i
\end{aligned} \tag{14}
$$

$$
\vec{v}_{t+1}^i = \frac{s \cdot \left((1-\lambda)\cdot \vec{r} + \lambda \cdot \vec{v}_t^i\right)}{\| (1-\lambda)\cdot \vec{r} + \lambda \cdot \vec{v}_t^i \|} \tag{15}
$$

where H_t^i, W_t^i and $\vec{C}_t^i$ denote the height, width and center of the i-th peak function at time t, $\vec{X}$ is the decision vector, and m is the total number of peaks. $N(0,1)$ denotes a random number drawn from a Gaussian distribution with zero mean and variance one. $\vec{r}$ is created by random numbers for each dimension and normalizing its length to s.

In Test Problem 2 (TP2), the H_t^i and W_t^i of the i-th peak change chaotically in Eq (16) reference in [22].

$$
\begin{aligned}
H_{t+1}^i &= H_{\min}^i + A \cdot (H_t^i - H_{\min}^i) \cdot (1 - \frac{H_t^i - H_{\min}^i}{\| H^i \|}) \\
W_{t+1}^i &= W_{\min}^i + A \cdot (W_t^i - W_{\min}^i) \cdot (1 - \frac{W_t^i - W_{\min}^i}{\| W^i \|})
\end{aligned} \tag{16}
$$

where, H_t^i is the value of H^i at time step t, $H_{\min}^i$ denotes the minimum value of H^i, $\| H^i \|$ denotes the range of H^i, W_t^i is the value of W^i at time step t, $W_{\min}^i$ denotes the minimum value of W^i, $\| W^i \|$ denotes the range of W^i, and A is a positive constant. The parameters of TP1 and TP2 are summarized in Table I.

TABLE II PARAMETERS OF PSO

Constants	Constriction factor	Swarm size	Generation	Constricted range
$c_1 = c_2$	χ	P	n	$[-V_{MAX}, V_{MAX}]$
2.05	0.729844	50	2500	50

TABLE I PARAMETERS OF MMPB BENCHMARK

Test problem	Peaks m	Dimension D	Change frequency Δe	Scale parameter s	Trend Paramete λ	Search Range	Initial Height	Initial Width	Width Range	Height Range	Height Severity range	Width Severity range	A
TP1	5	2	2500	1	1	[0,50]	50	6	[1,12]	[30,70]	[1,10]	[0.1,1]	/
TP2	5	2	2500	1	1	[0, 50]	50	6	[5,8]	[40,70]	/	/	3.6

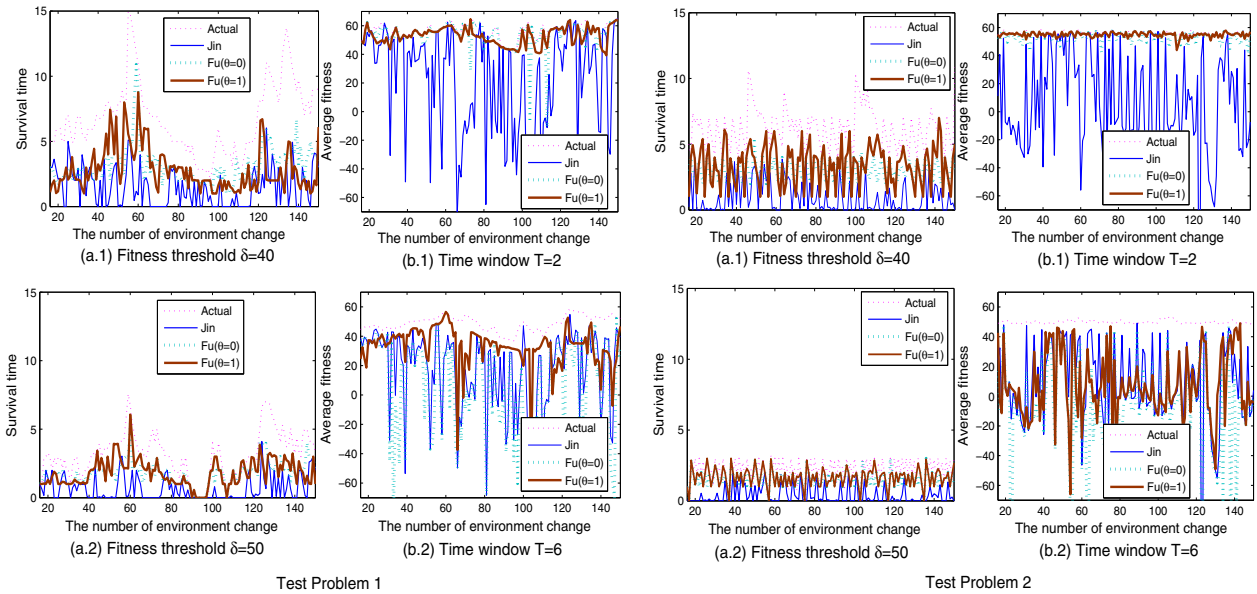

Fig. 2 The *survival time* and *average fitness* of each algorithm under different settings of δ and T on two test problems, respectively.

B. Parameter Settings

We adopt the framework proposed in [16] for finding ROOT solutions, where the canonical particle swarm optimization (PSO) algorithm is used as the optimizer in this work, referring to [11, 21] for more details of PSO. The parameter settings for the PSO are summarized in Table II. In the framework, we use the autoregressive (AR) model for the prediction. An AR model of order ψ takes the form $Y_t = \xi + \sum_{i=1}^{\psi} \eta_i \cdot Y_{t-i}$ where ξ is the white noise and Y_t is the time series data at time t. The least square method is used to estimate the parameters in the AR model $\vec{\eta}\left(\vec{\eta} = (\vec{\eta}_1, \vec{\eta}_2, \cdots \vec{\eta}_\psi)\right)$. In this work, ψ is set to 5 and the data in the 15 most recent time instants are used as the training data. If the AR model accuracy is considered, the first 12 time steps are chosen as training data, and the latest 3 time steps used to calculate E^{err}.

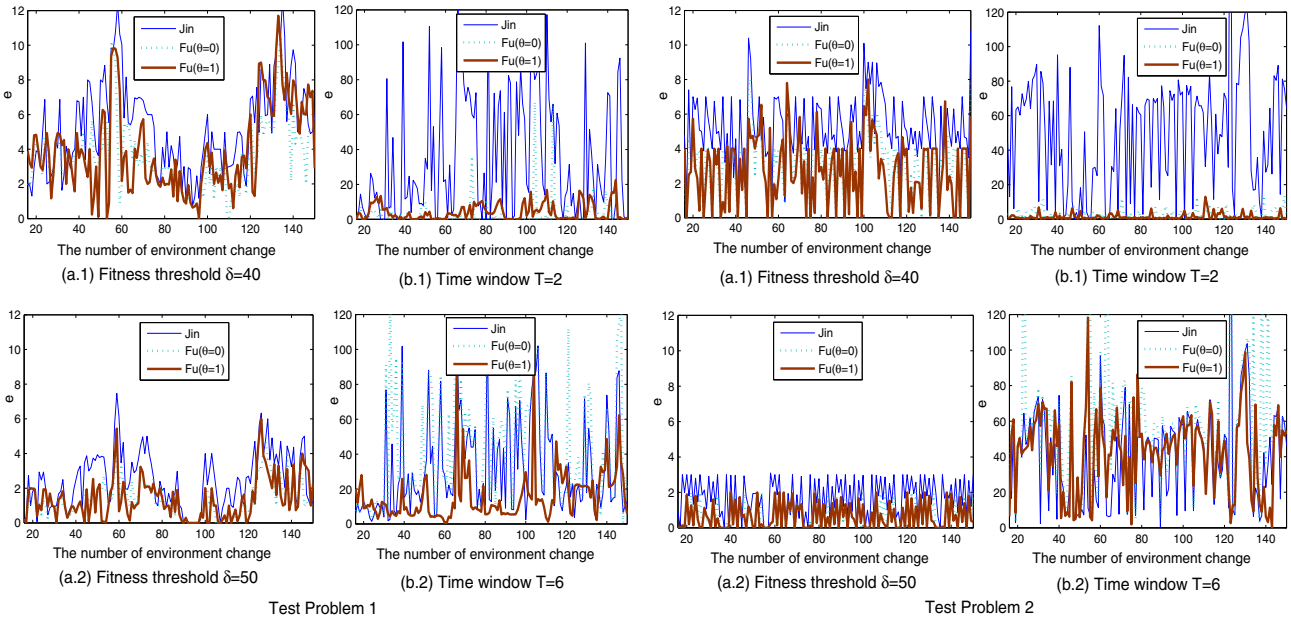

Fig. 3 The e of each algorithm under different settings of δ and T on two test problems, respectively.

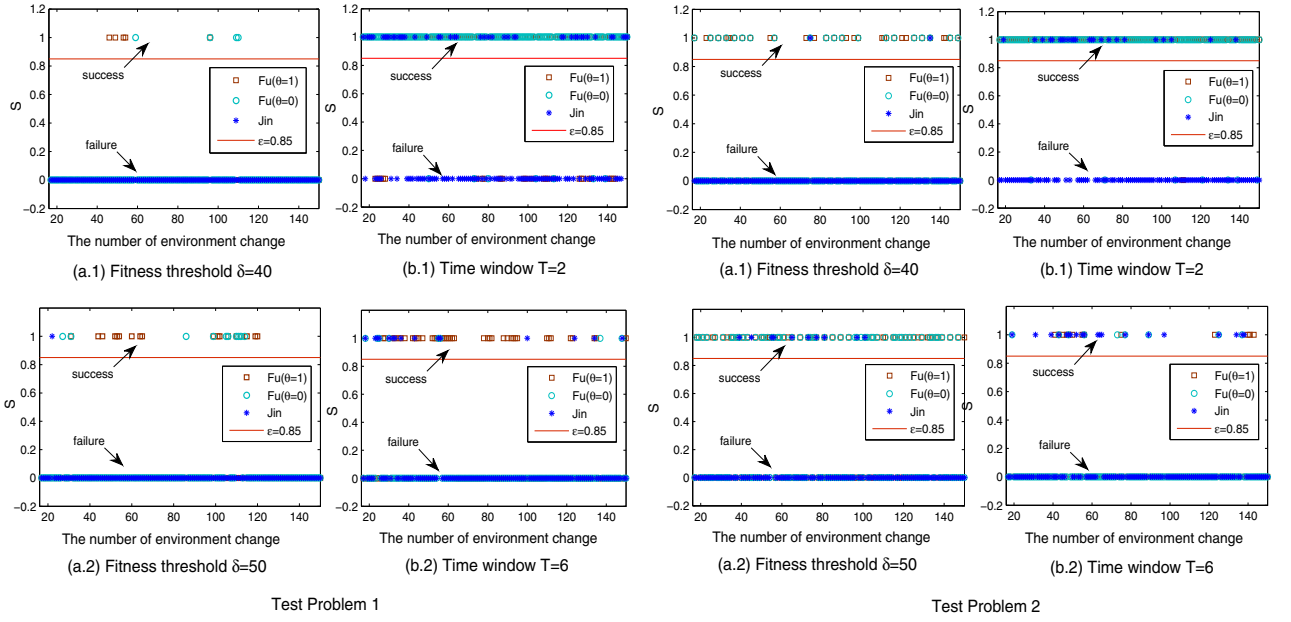

Fig .4 The S of each algorithm under different settings of δ and T on two test problems, respectively.

V. EXPERIMENTAL RESULTS AND ANALYSIS

In the experiment, we use a fixed random number generator to produce 150 consecutive changes of the fitness function. We compare the results obtained by the algorithm presented by Jin et al in [16], Jin's algorithm for short, and the two variants reported in Fu et al [18], where Fu($\theta = 1$) denoting the one considering the accuracy of the predictor, while Fu($\theta = 0$) means the algorithm that does not take the prediction error into account. All the results here are the average over 30 independently runs.

A. Analysis of Dynamic Performance

The results from the compared algorithms in terms of the *survival time* R^s and *average fitness* R^a on TP1 and TP2 under different settings of δ and T are shown in Fig. 2.From Fig. 2 we can see that the actual robustness values, denoted by the thin dash line are mostly larger than those found by the algorithms. This is expected as the robustness of the solutions achieved by the ROOT optimizers cannot be larger than that of the true one, and only in a small number of environments the true ROOT solution is identified. There is a small number of environments, in which the robustness of the solutions found by the optimizers is larger than the true one. This can most likely be attributed to the prediction errors in estimating the robustness. We also note that in a small number of environments, the true robustness equals zero, indicating that there is no ROOT solutions in these environments.

From Fig. 2, we also see that the larger δ is， the smaller the *survival time* R^s. Similarly, the larger T is, the smaller *average fitness* R^a. This is reasonable because an optimal solution is more difficult to survive if the threshold δ for acceptable solutions is larger. It is also straight forward that the average fitness will become smaller if the time window T is larger.

The *error* curves of the solutions found by the compared algorithms on the two test problems are given in Fig. 3. The *error* shows the difference between robustness values of solutions found by each algorithm and that of the true ROOT solution in each environment. From the figures, we can see that this performance indicator makes it possible to quantitatively compare the performance of the solutions found by different algorithms in each environment.

From Fig. 3 (a.1), (a.2), it is evident that δ increases, the *error* of each algorithm decreases, and reaches zero in more environments. The *error* equaling zero indicates that the robustness value found by an algorithm equals to the actual robustness value. In general, we can see that the large δ is, the better the dynamic performance of the algorithm. The reason may be that both Jin's and Fu's algorithms are based on function prediction. The larger δ is, the shorter an optimal solution can survive, and therefore, the smaller the number of environments the model needs to predict. The more accurate the prediction is, the more likely that a ROOT optimizer is able to identify the true ROOT solution, therefore the smaller the *error e*. The results on TP1 presented in Fig. 3(b.1), (b.2) indicate that the error of Fu's algorithm increases, and the number of environments in which the *error* equals zero decreases as T increases. This suggests that Fu's algorithm is more suited for problems having a smaller T. By contrast, the results obtained by Jin's algorithm exhibit a smaller *error* with an increasing T, indicating that Jin's algorithm is less sensitive to T.

The results on TP2 presented in Fig.3, from which conclusions similar to those from TP1 can be drawn.

Fig. 4 show the distribution of success s of each algorithm in each environment, where $\varepsilon = 0.85$ .Recall that the performance indicator s classifies the result obtained by a ROOT algorithm in each environment as success or failure using the threshold ε . The performance indicator *success* provides us a more intuitive and simpler measure than the *error* curve.

Results in Fig. 4, e.g., Fig. 4(a.2) show that Fu's algorithm considering the prediction accuracy achieves the largest number of successes, and Jin's algorithm the smallest. In other words, Fu's algorithm considering prediction accuracy performs the best in terms of success rate. Meanwhile, it is noticed from Fig. 4(a.1)(a.2) that with the increase of δ ,the number of successes increases for all algorithms.

Finally from Fig. 4, we can conclude that both algorithms perform well in terms of the *survival time* and the *average fitness* on two test problems.

B. Analysis of the Overall Performance

The overall performance of the compared algorithms in terms of P defined in Equation (10) and σ defined in Equation (13) are summarized in Table III for different settings of δ and T , respectively, where the better performance is highlighted. No consistent conclusions can be drawn in that different algorithms perform differently on different problems in different situations. For instance for TP1, when $\delta = 40$, Fu's algorithm that does not consider prediction accuracy performs the best in terms of p, while Fu's algorithm considering prediction accuracy is the best in terms of σ . For $\delta = 50$, Fu's algorithm outperforms others both in terms of p and δ. Note that p focuses on the overall average robustness value, while σ pays more attention to the success ratio of the algorithm. In practice, the first thing we want to make sure is that the algorithm is successful. Once this is satisfies, we can then check which algorithm can achieve a large average robustness. The reason is that a larger average robustness value does not guarantee a larger *success rate*. Therefore, σ is practically more important than p .

The performance of the solutions obtained by each algorithm under different δ for TP1 and TP2 are listed in Table III. From these results, we can observe that p ($\delta = 40$)> p ($\delta = 50$) for all algorithms. These results show that the larger δ is, the smaller p is. Because p is affected by the settings of δ , it cannot effectively measure performance of the algorithms under different δ . However, σ is related to ε only, therefore, σ can effectively indicate the performance of each algorithm under different settings for δ . The results on TP1 and TP2 obtained the algorithms under different T are more or less similar.

TABLE III PERFORMANCE MEASUREMENT PARAMETERS VALUES

Test problem	Algorithms	$\delta = 40$		$\delta = 50$		$T = 2$		$T = 6$	
		p	σ (%)	p	σ (%)	p	σ (%)	p	σ (%)
TP1	Actual	6.13		2.83		51.63		44.21	
	Fu(θ=1)	2.74	**3.68**	**1.71**	**15.00**	52.45	80.74	**32.84**	**31.11**
	Fu(θ=0)	**3.02**	2.94	1.70	8.33	**53.37**	**94.81**	8.80	5.93
	Jin	1.54	0.00	0.69	0.83	25.16	33.33	18.54	11.85
TP2	Actual	5.61		1.92		50.96		45.20	
	Fu(θ=1)	**3.40**	**25.00**	**1.62**	**49.19**	**54.95**	**100.00**	10.87	10.37
	Fu(θ=0)	3.08	14.71	1.50	45.97	53.90	95.56	-2.96	6.67
	Jin	1.21	1.47	0.50	8.87	11.11	18.52	**12.54**	**15.56**

From the above discussions and analyses, we can conclude that σ can measure the performance of different algorithms also for different settings of δ and T.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we pointed out the weaknesses of existing methods for evaluating the performance of ROOT algorithms and proposed three new performance indicators, including *error*， *success* and *success rate*. The proposed performance indicators are empirically evaluated on two test problems to show their effectiveness in evaluating two ROOT algorithms proposed in [16] and [18]. Our results show that *error* can be used to quantitatively compare the difference between robustness solutions found by a ROOT algorithm and that of the true ROOT solution. *Success* can be used to judge intuitively the degree of success of each ROOT algorithm in achieving the true ROOT solution in each environment. Finally, *success rate* can serve as a performance indicator for measuring the overall performance of each algorithm. Using these performance indicators, we are able to compare different ROOT algorithms in different environments.

Prediction of the fitness in future environments is critical in ROOT algorithms. Therefore, finding strategies to improve the accuracy of prediction will be important future research. In addition, we will consider the cost for switching solutions in the ROOT framework, which has not been taken into account in existing ROOT algorithms.

ACKNOWLEDGMENT

This work was supported in part by the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (No. 61428302),the Key Project of the National Nature Science Foundation of China (No. 61134009), the National Nature Science Foundation of China (No. 61473077, 61473078), Program for Changjiang Scholars from the Ministry of Education, Specialized Research Fund for Shanghai Leading Talents, Project of the Shanghai Committee of Science and Technology (No. 13JC1407500), and the scientific research project of science and Technology bureau of Jiaxing city project (2013AY11026).

REFERENCES

[1] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments – A survey," IEEE Transactions on Evolutionary Computation, vol. 9, no. 3, pp. 303–317,June 2005.

[2] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," Swarm and Evolutionary Computation, vol. 6, pp. 1–24,2012.

[3] C. Cruz, J. R. Gonzalez, D. A. Pelta, "Optimization in dynamic environments: A survey on problems, methods and measures," Soft Computing vol.15,no. 7,pp.1427–1448,2011.

[4] Y. Jin and B. Sendhoff, "Constructing dynamic optimization test problems using the multi-objective optimization concept," EvoWorkshop 2004, Lecture Notes in Computer Science, vol. 3005, pp. 526–536,2004.

[5] S. Yang, Y.,Jin, Y. S. Ong,.: "Evolutionary Computation in Dynamic and Uncertain Environments." Springer, Heidelberg ,2007.

[6] X. Yu, K. Tang, T. Chen, and X. Yao, "Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization," Memetic Computing, vol. 1, no. 1, pp. 3–24,2009.

[7] J. Brest, A. Zamuda, B. Boskovic , M. S. Maucec, and V. Zumer,"Dynamic optimization using self-adaptive differential evolution," in Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 415–422,2009.

[8] E. L. Yu and P. N. Suganthan, "Evolutionary programming with ensemble of explicit memories for dynamic optimization," in Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 431–438,2009.

[9] H. K. Singh, A. Isaacs, T. T. Nguyen, T. Ray, and X. Yao, "Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems," in Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 3127–3134,2009.

[10] S. Yang, C. Li. "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments."IEEE Transactions on Evolutionary Computation, vol.14,no,6,pp. 959–974, 2010.

[11] X.Hu and R.C.Eberhart,"Adaptive particle swarm optimization: Detection and response to dynamic systems," in Proceedings of the2002 IEEE Congress on Evolutionary Computation, pp. 1666–1670, 2002.

[12] H. Richter, "Detecting change in dynamic fitness landscapes," in Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp.1613–1620, 2009.

[13] J.Branke, "Memory enhanced evolutionary algorithms for changingoptimization problems," in Proceedings of the 1999 IEEE Congress on Evolutionary Computation, vol. 3.pp. 1875–1882, 1999.

[14] P. A. N. Bosman, "Learning, anticipation and time-deception in evolutionary online dynamic optimization," in Proceedings of the 2005 Genetic and Evolutionary Computation Conference, pp. 39–47, 2005.

[15] X.Yu,Y. Jin, K. Tang, and X. Yao, "Robust optimization over time-A new perspective on dynamic optimization problems,"in 2010 IEEE Congress on Evolutionary Computation. pp.1-6,2010.

[16] Y.Jin, K.Tang, X. Yu,, B.Sendhoff, and X.Yao. "A framework for finding robust optimal solutions over time," Memetic Computing, vol.5, pp. 3-18, 2013.

[17] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Characterizing environmental changes in robust optimization over time," in IEEE Congress on Evolutionary Computation (CEC), pp. 1–8,2012.

[18] H. Fu, B. Sendhoff, K. Tang, and X. Yao. "Finding robust solutions to dynamic optimization problems," in Proceedings of the 16th European conference on Applications of Evolutionary Computation. Springer-Verlag, pp. 616–625,2013.

[19] Y. Guo, M. Chen, H. Fu, and Y. Liu. "Find robust solutions over time by two-layer multi-objective optimization method" in Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), pp.1528–1535,2014.

[20] R. W. Morrison and K. A. De Jong, "A test problem generator for non-stationary environments," in Proceedings of the 1999 IEEE Congress on Evolutionary Computation, pp. 2047–205. March, 1999.

[21] Clerc, M., Kennedy, J "The particle swarm-explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation vol.6,no.1, pp.58–73, 2002.

[22] C. Li, S. Yang, T. Nguyen, E. Yu, X. Yao, Y. Jin, H. Beyer, and P. Suganthan, "Benchmark generator for CEC 2009 competition on dynamic optimization," University of Leicester, University of Birmingham,Nanyang Technological University, Tech. Rep, 2008.