

Robust Optimization over Time – A New Perspective on Dynamic Optimization Problems

Xin Yu, *Student Member, IEEE*, Yaochu Jin, *Senior Member, IEEE*, Ke Tang, *Member, IEEE*,
and Xin Yao, *Fellow, IEEE*

Abstract—Dynamic optimization problems (DOPs) are those whose specifications change over time during the optimization, resulting in continuously moving optima. Most research work on DOPs is based on the assumption that the goal of addressing DOPs is to track the moving optima. In this paper, we first point out the practical limitations on tracking the moving optima. We then propose to find optimal solutions that are robust over time as an alternative goal, which leads to a new concept of robust optimization over time (ROOT) problem. In order to investigate the properties of ROOT in more depth, we study the new characteristics of ROOT and investigate its similarities to and differences from the traditional robust optimization problem, which hereafter is referred to as robust optimization for short. To facilitate future research on ROOT, we suggest a ROOT benchmark problem by modifying the moving peaks test problem. Several performance measures for comparing algorithms for solving ROOT problems are proposed.

I. INTRODUCTION

Optimization problems can be found everywhere in science, technology and even in daily life. For example, stock market investors seek to find the best portfolio that maximizes the profit, and aerodynamic engineers work on designs of the blades that maximize the energy efficiency of turbine engines. Meanwhile, most of the real-world optimization problems are subject to various types of uncertainties. For example in stock markets, political and economical environments are changing from time to time; The velocity of the airplane varies considerably during take-off and landing compared to cruising.

Population-based evolutionary algorithms (EAs) have been shown successful in solving stationary optimization problems and are expected to cope well with optimization problems with uncertainties because natural evolutionary dynamics also happened in a highly uncertain environment.

In this work, we focus on dynamic optimization problems (DOPs), where the specifications (often known as objectives or fitness functions in evolutionary optimization) of the optimization change over time, causing the global optima to shift as well [1], [2]. In DOPs, the fitness function is deterministic at any time instant, but dependent on time t ,

The authors are with the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Yaochu Jin is also with the Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, UK. Xin Yao is also with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK. (emails: hughyx@mail.ustc.edu.cn, yaochu.jin@surrey.ac.uk, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

i.e.,

$$f(\vec{x}, \vec{\alpha}(t)), \quad (1)$$

where \vec{x} represents design parameters, $\vec{\alpha}(t)$ represents time-dependent problem parameters.

Over last two decades, DOPs have drawn much research attention in the evolutionary optimization community. Many evolutionary algorithms have been developed to address DOPs [2], [3], [4], [5], [6]. The most common goal in solving DOPs is to track the moving optima over time, which can be unrealistic in many real-world optimization problems due to the following reasons:

- (1) In order to track the moving optima, many EAs for solving DOPs require the detection of the environmental changes. Nonetheless, how to detect environmental changes is not trivial as well [7], [8].
- (2) Most approaches to tracking moving optima attempt to make use of knowledge from the past [2], which is based on the assumption that the environmental changes are small to medium. Otherwise, a random restart strategy may be a better choice [9]. Nonetheless, severe environmental changes can occur very often in practice. Thus it is inefficient to re-start the search from scratch. Worse, the environment may change again before the optima are located. This is also true even for small or medium environmental changes. The main reason is that the process of relocating the optima can be slowed down by evaluations of candidate solutions, which may be very time-consuming in real-world applications.
- (3) Even if the new solution was able to be found successfully before the environment changes, in real-world applications, it can still be impractical to use the solution due to limited resources such as time and cost.
- (4) In some real-world problems with “time-linkage” [10], such as scheduling and vehicle routing, tracking the moving optima is not the best choice, because the decision made to maximize the performance at present may influence the performance in the future, decreasing the overall performance in the long run.

To address the above concerns, we suggest to find solutions that are robust over time, instead of following the changing optima. A solution is called robust over a certain time interval when its quality remains acceptable and is relatively insensitive to the environmental changes during this time interval. A found solution that is robust over time will be used until its quality degrades to an unacceptable level in the current

environment. When the solution quality is unsatisfactory, a new robust solution must be found. Therefore, the task for addressing the DOPs now becomes to find a sequence of robust solutions over time intervals. The ideal situation takes place when only one solution is enough and robust over the whole life cycle of the problem. The process of finding such a sequence of robust solutions is referred to as *robust optimization over time (ROOT)*.

It is worth stressing the difference between the traditional definition for robustness [11] and the definition of ROOT. Robustness in the traditional sense mainly concerns the uncertainties in the parameter space, either design parameters or environmental parameters. In contrast, robustness in ROOT not only takes into account uncertainties in the parameter space, but also the cumulative effect of these uncertainties in the time space. Since dealing with uncertainties in constraints is often the task of reliability-based optimization [12], we focus on unconstrained continuous problems in this study.

The main contributions of this paper are the proposal of a new concept for DOPs that searches for optimal solutions robust in the time space, which we believe is a more practical way of addressing continuously changing DOPs. In addition, we suggest a benchmark problem and several performance measures for benchmarking optimization algorithms that target for robust optimal solutions over time.

The remainder of the paper is outlined as follows. Section II presents a class of widely studied DOPs and describes formally the task of tracking the moving optima and ROOT. In Section III, the analyses of ROOT on the problem presented in Section II are given with respect to different types of uncertainties. Section IV provides a brief overview of the existing benchmark problems for DOPs, followed by a proposal of a benchmark problem and a few criteria for algorithms targeting for ROOT. Conclusions and future work are presented in Section V.

II. DYNAMIC OPTIMIZATION PROBLEMS

A. Problem Definition

A dynamic problem can be generally described as Eq. 1, where \vec{x} represents design parameters, $\vec{\alpha}(t)$ represents time-dependent problem parameters, t is the time index with $t \in [0, T_{end}]$ (T_{end} is the life cycle of the problem, $T_{end} > 0$), and f is the objective function. Note that time t is assumed to be discrete and can be generally measured in function evaluations. Here we consider a class of dynamic optimization problems that are most widely studied in the literature: The parameters of the problem change over time with stationary periods between changes. In other words, $\vec{\alpha}(t)$ does not have to change continuously over time, and thus within T_{end} there may be a sequence of $l = \lceil T_{end}/\tau \rceil$ different instances of the same problem (the problem with parameters $\langle \vec{\alpha}_1, \vec{\alpha}_2, \dots, \vec{\alpha}_l \rangle$), where $1/\tau$ is the frequency of the change and usually remains constant but could also be time-variant. Therefore, the problem can be re-formulated as:

$$\langle f(\vec{x}, \vec{\alpha}_1), f(\vec{x}, \vec{\alpha}_2), \dots, f(\vec{x}, \vec{\alpha}_l) \rangle . \quad (2)$$

B. Tracking the Moving Optima

In the field of DOPs, tracking the moving optima is the most frequently pursued target. For the dynamic problem defined in Eq. 2, tracking the moving optima means to find the global optimal solution S_i within the i th time interval T_i for the problem $f(\vec{x}, \vec{\alpha}_i)$ where $i = 1, 2, \dots, l$. Thus the task is to find a sequence of corresponding global optimal solutions $\langle S_1, S_2, \dots, S_l \rangle$.

C. Robust Optimization over Time

As mentioned in Section I, the task of ROOT is to find a sequence of solutions which are robust over time. Formally, for the dynamic problem defined in Eq. 2, the goal is to find a sequence of solutions $\langle S_1, S_2, \dots, S_k \rangle$, and a solution S_i ($i = 1, 2, \dots, k$) is said to be robust if its performance is acceptable for at least two consecutive time intervals. Thus we have $1 \leq k \leq l$. Note that $k = l$ means no such robust solutions exist and for every problem instance a new solution must be found. On the other hand, the ideal situation occurs when $k = 1$, which means that only one optimal solution is sufficient for all problem instances, and thus there is no need to search for another optimal solution when the environment changes.

III. PROPERTIES OF ROOT

ROOT can be regarded as a consideration of both robust optimization and DOPs. It concerns the cumulative effect on the time axis of the uncertainties deriving from the parameter space. Therefore, ROOT inherits the properties of robust optimization and has its own new properties. In the remainder of this section, we will analyze the properties of ROOT based on the dynamic problem defined in Eq. 2 in the time interval $[t_L, t_U] \subseteq [0, t_{end}]$. Let $l_L = \lceil t_L/\tau \rceil$ and $l_U = \lceil t_U/\tau \rceil$, we are going to analyze ROOT on the following sequence of problems:

$$\langle f(\vec{x}, \vec{\alpha}_{l_L}), f(\vec{x}, \vec{\alpha}_{l_L+1}) \dots, f(\vec{x}, \vec{\alpha}_{l_U}) \rangle . \quad (3)$$

Without the loss of generality, we assume f is to be maximized.

To facilitate the analysis, the dynamics between two successive changes should be defined. Though there exist numerous dynamic models such as the logistic map in the DF1 benchmark [13], the Fourier function in TCG benchmark [14], here we consider simple additive dynamics between two successive changes, i.e.,

$$\vec{\alpha}_i = \vec{\alpha}_{i-1} + \Delta\vec{\alpha}, \quad (4)$$

where $i = 2, 3, \dots, l$. In the next, we will analyze ROOT in the context of different types of uncertainties on $\Delta\vec{\alpha}$, namely, deterministic, probabilistic, and no prior knowledge is known about the uncertainties. These analyses are similar to those in robust optimization.

A. Deterministic Uncertainties

When $\Delta\vec{\alpha}$ is subject to deterministic uncertainties, it takes deterministic values within a specified interval $[\Delta\vec{\alpha}_L, \Delta\vec{\alpha}_U]$. Here the specified interval is assumed to be static.

In the case of deterministic uncertainties, in structural optimization, interval analysis is often used to estimate the bounds of the structural displacement [15], [16]. In robust optimization, this technique is also referred to as the *robust counterpart approach* [11]. In ROOT, we use this term as well. For the problem in Eq. 3, the robust counterpart function $F(\vec{x}; l_L, l_U; \Delta\vec{\alpha}_L, \Delta\vec{\alpha}_U)$ is defined as

$$F(\vec{x}; l_L, l_U; \Delta\vec{\alpha}_L, \Delta\vec{\alpha}_U) = \inf_{i \in [l_L, l_U], \Delta\vec{\alpha} \in [\Delta\vec{\alpha}_L, \Delta\vec{\alpha}_U]} f(\vec{x}, \vec{\alpha}_i). \quad (5)$$

This kind of technique presents a worst case scenario in that it considers the minimal f -value within the neighborhood of the dynamics of parameter $\Delta\vec{\alpha}$ during the time interval $[t_L, t_U]$.

Consider a simple one-dimensional function:

$$f(x, \alpha) = x + \alpha, \quad (6)$$

which is to be maximized within the range $x \in [x_L, x_U]$, and α is a time-dependant parameter changed by adding variations bounded in the interval $[\Delta\alpha_L, \Delta\alpha_U]$. It is easy to calculate its robust counterpart function in the time interval $[t_L, t_U]$, i.e.,

$$F(x; l_L, l_U; \Delta\alpha_L, \Delta\alpha_U) = \begin{cases} x + \alpha_{l_L}, & \text{if } \Delta\alpha_L \geq 0 \\ x + \alpha_{l_L} + (l_U - l_L)\Delta\alpha_L, & \text{otherwise.} \end{cases} \quad (7)$$

Therefore, we only need to find a solution $x \in [x_L, x_U]$ such that the fitness value calculated according to Eq. 7 is acceptable, and then this solution can be continuously used regardless of the changes in fitness function during the time interval $[t_L, t_U]$.

B. Probabilistic Uncertainties

In the case of probabilistic uncertainties, the dynamics parameter $\Delta\vec{\alpha}$ is regarded as random variable obeying a certain distribution functions such as a Gaussian distribution or a uniform distribution. Here the parameters of the distribution functions (e.g., the mean and the standard deviation of the Gaussian distribution function) are assumed to be time-invariant.

In robust optimization, the *momentum measures* are used to address probabilistic uncertainties [17]. For the problem in Eq. 3, the momentum measure is defined as

$$F_K(\vec{x}; l_L, l_U) = \frac{1}{l_U - l_L + 1} \sum_{i=1}^{l_U - l_L + 1} \int \text{sign}(f) |f(\vec{x}, \vec{\alpha}_{l_L} + (i-1)\Delta\vec{\alpha})|^K p(\Delta\vec{\alpha}) d\Delta\vec{\alpha}, \quad (8)$$

where $p(\Delta\vec{\alpha})$ is the probability density function of $\Delta\vec{\alpha}$. Note that we use $\text{sign}(f)|f|^K$ instead of f^K in order to correctly

treat the even K cases for negative f -values. When $K = 1$, we have

$$F_1(\vec{x}; l_L, l_U) = \frac{1}{l_U - l_L + 1} \sum_{i=1}^{l_U - l_L + 1} \int f(\vec{x}, \vec{\alpha}_{l_L} + (i-1)\Delta\vec{\alpha}) p(\Delta\vec{\alpha}) d\Delta\vec{\alpha}. \quad (9)$$

We can see that the integral part of F_1 is a standard measure often encountered in the literature of robust optimization and has been termed ‘‘effective fitness’’ in [18]. The measure for ROOT calculates the cumulative effect in the time space (i.e., the discrete sum part) of the robust optimization. From Eq. 9, it can be seen that F_1 actually measures the average performance over all possible problem instances in the time interval $[t_L, t_U]$.

Sometimes, for a solution to the ROOT problem, we also want its performance not to vibrate drastically when the environment changes. To this end, a dispersion measure can be defined as follows:

$$F_d(\vec{x}; l_L, l_U) = \frac{1}{l_U - l_L + 1} \sum_{i=1}^{l_U - l_L + 1} \int (f(\vec{x}, \vec{\alpha}_{l_L} + (i-1)\Delta\vec{\alpha}) - F_1(\vec{x}; l_L, l_U))^2 p(\Delta\vec{\alpha}) d\Delta\vec{\alpha}, \quad (10)$$

where the subscript ‘‘d’’ means ‘‘dispersion’’.

As a simple example, consider the one-dimensional objective function

$$f(x, \alpha) = \alpha - (\alpha - 1)x^2, \quad \alpha \in \mathbb{R}, x \in \mathbb{R} \quad (11)$$

to be maximized, where α changes according to Eq. 4 and $\Delta\alpha$ is assumed to be normally distributed $\Delta\alpha \sim N(0, 1)$. In the time interval $[t_L, t_U]$, according to Eq. 9 and Eq. 10, the momentum measure and dispersion measure can be calculated as follows:

$$F_1 = \alpha_{l_L} - (\alpha_{l_L} - 1)x^2, \quad (12)$$

and

$$F_d = \frac{(l_U - l_L)[2(l_U - l_L) + 1]}{6} (1 - x^2)^2. \quad (13)$$

From Eq. 12 and Eq. 13, we can see that if $\alpha_{l_L} \leq 1$, maximizing F_1 and minimizing F_d are not conflicting goals. However, if $\alpha_{l_L} > 1$, maximizing F_1 and minimizing F_d represent conflicting goals, and hence we are dealing with *multi-objective robust optimization*. Using the goals in Eq. 12 and Eq. 13 (as an example Pareto-front, here we set $\alpha_{l_L} = 2$ and $l_U - l_L = 3$), the Pareto-front of the example function in Eq. 11 is shown in Fig. 1.

C. No Assumption on the Distribution of the Uncertainties

In real-world applications, it is quite often that there is no prior knowledge about the characteristics of the uncertainties. In the field of robust optimization, a max-min optimization strategy is frequently used to address this kind of uncertainties. The max-min optimization strategy is similar to the techniques used for deterministic uncertainties (see

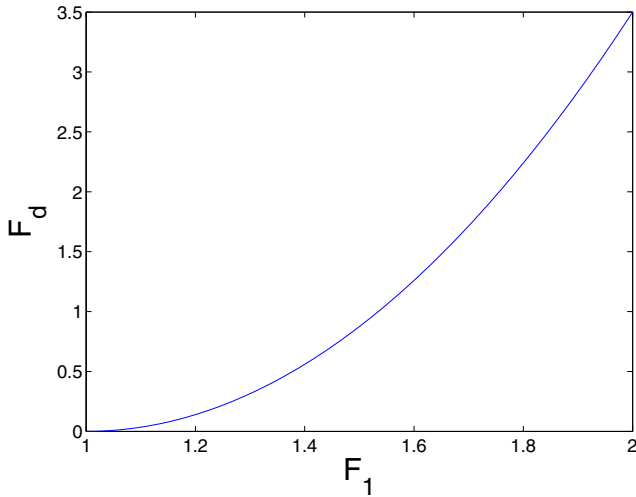


Fig. 1. Pareto-front of example function in Eq. 11 using the goals of maximizing F_1 in Eq. 12 and minimizing F_d in Eq.13: The curve represents the non-dominated solutions of the multi-objective optimization problem.

Section III.A) in that they are all considered as a worst case philosophy, while they are different in that the former is often employed empirically.

Ong et al. [19] estimated the worst performance of each candidate solution within its neighborhood determined by the bounds in which the uncertainty parameters vary. To reduce the computational cost introduced by the nested search (which is used to find the worst performance solution), they used a Baldwinian trust-region framework employing local surrogate models. In practical, the bounds where the uncertainty parameters vary are usually unknown. In this situation, their approach is not applicable. Recently, Lim et al. [20] proposed an inverse multi-objective robust evolutionary (IMORE) algorithm to address the robust optimization problems. Their method does not require any prior knowledge, neither the probability distribution nor the bounds of the uncertainties. They employed a nested search to find the maximum uncertainty given the maximum degradation tolerable for the final solution. Then they regarded this uncertainty as a second objective apart from the nominal fitness function, and utilized a multi-objective evolutionary algorithm to solve the problem.

In the context of ROOT, we can still resort to the above methodologies. For example, during a time interval, we can either estimate the worst performance within the bounds of the uncertainties or assess the maximum uncertainty given the maximum tolerable degradation in the performance of the final solution. However, now we must consider the effect of uncertainties in the time space. Sometimes we even should know the impact of the effect in the future. Therefore, classical DOP algorithms and robust optimization algorithms cannot be applied (with some simple changes) to ROOT.

IV. BENCHMARKING ROOT ALGORITHMS

In order to compare the performance of algorithms solving ROOT problems, an important task is to develop proper

benchmark problems. There have been many benchmark problems for DOPs. However, they are mostly used to test the ability of algorithms to track moving optima. As aforementioned, ROOT can be regarded as a more practical way to address DOPs. Therefore, we can create benchmark problems for ROOT by modifying existing benchmark problems for DOPs.

A. Existing Benchmark Problems for DOPs

Over the years, a number of dynamic test problems have been proposed for comparing the performance of algorithms solving DOPs. In continuous optimization, the most frequently used benchmarks are Branke’s moving peaks benchmark (MPB) [9] and Morrison and De Jong’s DF1 generator [13]. Both of them model the search space as a field of cones, each of which can be individually controlled. The height, width and location of each peak are varied over time. Jin and Sendhoff [21] proposed a single and multi-objective dynamic test problem generator by dynamically combining different objective functions of exiting stationary multi-objective benchmark problems. Due to the lack of research on continuous dynamic constrained optimization, Nguyen and Yao [22] introduced a continuous dynamic constrained benchmark set. Criticizing existing benchmark for DOPs could not reflect the real-world problems, Ursem et al. [14] suggested a realistic test-case generator and designed a greenhouse benchmark problem.

On the other hand, in combinatorial optimization, the exclusive-or (XOR) operator [23], [24] is the most frequently used benchmark. It can generate a dynamic binary problem from its stationary version by rotating the search points without changing the actual problem. Besides, Li et al. [25] proposed a dynamic traveling salesman problem (DTSP) and a dynamic multi knapsack problem (DKP). Furthermore, Rohlfschagen and Yao [26] revisited the dynamic knapsack problem and suggested a challenging dynamic knapsack problem that can represent well real-world scenarios. Recently, Li et al. [27] proposed a generalized dynamic benchmark generator (GDBG). It is a unified approach to constructing dynamic problems across the binary space, real space and combinatorial space.

B. A Modified Moving Peaks Benchmark for ROOT

As a first step towards creating ROOT benchmark problems, we focus on dynamic unconstrained continuous space optimization problems, and aim at building benchmark problems based on Branke’s MPB [9], which is among the earliest and the most often used dynamic benchmark problems.

The MPB is a multidimensional landscape consisting of several peaks, where the height, the width and the position of each peak vary slightly when an environmental change takes place. An n -dimensional test function with m peaks is formulated as:

$$F(\vec{x}, t) = \max(B(\vec{x}), \max_{i=1 \dots m} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))), \quad (14)$$

where $B(\vec{x})$ is a time-invariant basis landscape, and P is the function defining peaks' shapes, where each of the m peaks has its own time-varying parameters including height h , width w , and location \vec{p} .

Every Δe evaluations the height and width of the i th peak are changed as follows:

$$\begin{aligned} \sigma &\in N(0, 1) \\ h_i(t) &= h_i(t-1) + \text{height_severity} \cdot \sigma \\ w_i(t) &= w_i(t-1) + \text{width_severity} \cdot \sigma. \end{aligned} \quad (15)$$

The location is moved by a vector \vec{v}_i of a fixed length s in a random direction ($\lambda = 0$) or a direction exhibiting a trend ($\lambda > 0$) as follows:

$$\vec{p}_i(t) = \vec{p}_i(t-1) + \vec{v}_i(t). \quad (16)$$

The shift vector $\vec{v}_i(t)$ is a linear combination of a random vector \vec{r} and the previous shift vector $\vec{v}_i(t-1)$, and is normalized to a length of s , i.e.

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|} ((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)). \quad (17)$$

The random vector \vec{r} is created by drawing random numbers for each dimension and normalizing its length to s . The function's complexity may easily be scaled by increasing the number of dimensions and/or the number of peaks, or by using complex peak- and basis functions.

Observing the MPB described above, we find that the height and width of all peaks are changed at the same frequency and severity in addition to Gaussian noise. In the context of ROOT, each peak should change with different frequencies and severities. In this way, the quality of some solutions change much faster or more severely than others. Only in this case, it makes sense to find solutions that are robust over time.

Hence in order to tailor the MPB to ROOT, we require that each peak has its own `height_severity` and `width_severity`. We still allow all peaks change at the same pace (i.e., all peaks's changing frequency is $1/\Delta e$). Now, every Δe evaluations, the i th peak changes as follows:

$$\begin{aligned} \sigma &\in N(0, 1) \\ h_i(t) &= h_i(t-1) + \text{height_severity}_i \cdot \sigma \\ w_i(t) &= w_i(t-1) + \text{width_severity}_i \cdot \sigma. \end{aligned} \quad (18)$$

The position of each peak varies according to Eq. 16 and Eq. 17 as well.

C. Evaluation Criteria for ROOT

When a sequence of solutions $S = \langle S_1, S_2, \dots, S_k \rangle$ is found for the dynamic problem defined in Eq. 2, the task now is to evaluate how good the solution is. As mentioned in Section II.C, a solution is called robust when it is used for at least two consecutive problem instances and thus we have $1 \leq k \leq l$. As to the criteria of the evaluation, we can take the following considerations into account:

- (1) From the perspective of a single solution S_i , its quality can be determined by

- N_i : the number of different problem instances in which it is used.
- ϵ_i : the average error defined as follows:

$$\epsilon_i = \frac{1}{N_i} \sum_{j=l_0}^{l_0+N_i-1} |\text{opt}_j - f(S_i, \vec{\alpha}_j)|, \quad (19)$$

where $f(\vec{x}, \vec{\alpha}_{l_0})$ is the first problem instance S_i is used, $f(S_i, \vec{\alpha}_j)$ is the fitness value of the solution S_i , and opt_j is the fitness value of the true optimum of the j th problem instance.

- If S_i is robust, we can also evaluate its sensitivity to the environment using the measure below:

$$\text{std}_i = \sqrt{\frac{1}{N_i - 1} \sum_{j=l_0}^{l_0+N_i-1} (|\text{opt}_j - f(S_i, \vec{\alpha}_j)| - \epsilon_i)^2}. \quad (20)$$

It can be seen from the above three measures that the larger N_i is, the smaller the ϵ_i and the std_i , the better the S_i .

- (2) From the perspective of the whole sequence of solutions $S = \langle S_1, S_2, \dots, S_k \rangle$, its quality can be determined by

- k : the length of the sequence;
- The following four measures:

$$E_{\text{best}} = \min_{j=1}^k \epsilon_j,$$

$$E_{\text{avg}} = \frac{1}{k} \sum_{j=1}^k \epsilon_j,$$

$$E_{\text{worst}} = \max_{j=1}^k \epsilon_j,$$

$$\text{STD}_E = \sqrt{\frac{1}{k-1} \sum_{j=1}^k (\epsilon_j - E_{\text{avg}})^2}. \quad (21)$$

Smaller k and the above four values indicate better quality of $S = \langle S_1, S_2, \dots, S_k \rangle$.

Denote the sequence of all robust solutions by $S_r = \langle S_{r_1}, S_{r_2}, \dots, S_{r_k} \rangle$ ($1 \leq r_k \leq k$, and we denote the sequence $\langle r_1, r_2, \dots, r_k \rangle$ by R), we can evaluate the sensitivity of S_r in the same way, i.e.:

$$\text{std}_{\text{best}} = \min_{j \in R} \text{std}_j,$$

$$\text{std}_{\text{avg}} = \frac{1}{|R|} \sum_{j \in R} \text{std}_j,$$

$$\text{std}_{\text{worst}} = \max_{j \in R} \text{std}_j,$$

$$\text{STD}_{\text{std}} = \sqrt{\frac{1}{|R|-1} \sum_{j \in R} (\text{std}_j - \text{std}_{\text{avg}})^2}, \quad (22)$$

where $|R|$ is the length of the sequence R .

- (3) From the perspective of the search efficiency of the algorithm: The search efficiency of the algorithm can be measured by the time to find the first robust solution. The solution is robust if it is used for at least two

problem instances. Note that the time here is measured in the number of problem instances.

V. CONCLUSIONS AND FUTURE WORK

Over the years, most research on DOPs has been based on the goal of tracking moving optima. In this paper we attempted to investigate DOPs from a new perspective. We first pointed out some practical limitations when tracking moving optima is set to be the goal of addressing DOPs. To address these limitations, we proposed a new concept to search for robust optimal solutions over time (ROOT). Next, in the context of different types of uncertainties on the dynamics, we analyzed the properties of ROOT on a frequently studied class of DOPs. Our analyses showed that (at least for the case studied in this work), ROOT actually is concerned with the cumulative effect in the time space of uncertainties and in the parameters space. Finally, we suggested a benchmark problem for ROOT by modifying the moving peaks benchmark and proposed several performance measures.

We considered only one type of change dynamics in this paper. It is then natural to extend the analyses to other types of dynamics in our future work, since ROOT inherits properties of both DOPs and robust optimization. To this end, we are going to check if the above conclusions can be generalized. In addition, it is interesting to investigate the existing approaches to robust optimization and tracking the moving optima in the context of ROOT. Based on the investigation, we hope to come up with approaches to solving ROOT. Furthermore, from Section IV.C, we can see that ROOT has many different evaluation criteria. Therefore, it is worth analyzing ROOT in the context of multi-objective optimization. Finally, we will solve real-world dynamic problems from the perspective of ROOT.

ACKNOWLEDGMENT

This work was supported partially by an EPSRC grant (no.EP/E058884/1) on “Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications”, and the Fund for Creative Research for Graduate Students of University of Science and Technology of China.

REFERENCES

- [1] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer, 2002.
- [2] Y. Jin and J. Branke, “Evolutionary optimization in uncertain environments – A survey,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.
- [3] 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.
- [4] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer, “Dynamic optimization using self-adaptive differential evolution,” in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 415–422, 2009.
- [5] 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.
- [6] 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.
- [7] X. Hu and R. C. Eberhart, “Adaptive particle swarm optimization: Detection and response to dynamic systems,” in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, pp. 1666–1670, 2002.
- [8] H. Richter, “Detecting change in dynamic fitness landscapes,” in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 1613–1620, 2009.
- [9] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1875–1882, 1999.
- [10] 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.
- [11] H.-G. Beyer and B. Sendhoff, “Robust optimization – A comprehensive survey,” *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 3190–3218, 2007.
- [12] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, “Reliability-based optimization using evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054–1074, 2009.
- [13] 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. 2047–2053, 1999.
- [14] R. K. Ursem, T. Krink, M. T. Jensen, and Z. Michalewicz, “Analysis and modeling of control tasks in dynamic systems,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 378–389, 2002.
- [15] Z. Qiu and I. Elishakoff, “Antioptimization of structures with large uncertain-but-non-random parameters via interval analysis,” *Computer Methods in Applied Mechanics and Engineering*, vol. 152, pp. 361–372, 1998.
- [16] S. McWilliam, “Anti-optimisation of uncertain structures using interval analysis,” *Computers and Structures*, vol. 79, pp. 421–430, 2001.
- [17] H.-G. Beyer, M. Olhofer, and B. Sendhoff, “On the behavior of $(\mu/\mu_I, \lambda)$ -ES optimizing functions disturbed by generalized noise,” in K. De Jong, R. Poli, J. Rowe (Eds.), *Foundations of Genetic Algorithms*, vol. 7, Morgan Kaufman, San Francisco, CA, pp. 307–328, 2003.
- [18] S. Tsutsui, A. Ghosh, and Y. Fujimoto, “A robust solution searching scheme in genetic search,” in *Parallel Problem Solving from Nature*, Springer-Verlag, pp.543-553, 1996.
- [19] Y. S. Ong, P. B. Nair, and K. Y. Lum, “Max-min surrogate-assisted evolutionary algorithm for robust design,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, 2006.
- [20] D. Lim, Y. S. Ong, Y. Jin, B. Sendhoff, and B. S. Lee, “Inverse multi-objective robust evolutionary design,” *Genetic Programming and Evolvable Machines*, vol. 7, no. 4, pp. 383–404, 2006.
- [21] 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.
- [22] T. T. Nguyen and X. Yao, “Benchmarking and solving dynamic constrained problems,” in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 690–697, 2009.
- [23] S. Yang and X. Yao, “Experimental study on population-based incremental learning algorithms for dynamic optimization problems,” *Soft Computing*, vol. 9, no. 11, pp. 815–834, 2005.
- [24] S. Yang and X. Yao, “Population-based incremental learning with associative memory for dynamic environments,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 542–561, 2008.
- [25] C. Li, M. Yang, and L. Kang, “A new approach to solving dynamic TSP,” in *Proceedings of the 6th International Conference on Simulated Evolution and Learning*, pp. 236–243, 2006.
- [26] P. Rohlfshagen and X. Yao, “The dynamic knapsack problem revisited: A new benchmark problem for dynamic combinatorial optimisation,” *EvoWorkshops 2009*, Lecture Notes in Computer Science, vol. 5484, Springer-Verlag, Berlin, pp. 745–754, 2009.
- [27] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, “Benchmark generator for CEC’2009 competition on Dynamic Optimization,” Technical Report, 2008.