Dynamic Fitness Landscape Analysis on Differential Evolution Algorithm

Shuling Yang [1], Kangshun Li [1], Wei Li [1], Weiguang Chen [1], Yan Chen [1]

1 College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China

Email: celineyang514@gmail.com

Abstract Dynamic fitness landscape analysis contain different metrics to attempt to analyze optimization problems. In this paper, we select some of dynamic fitness landscape metrics to discuss differential evolution (DE) algorithm's properties and performance. Based on traditional differential evolution algorithm, benchmark functions and dynamic fitness landscape measures such as fitness distance correlation for calculating the distance to the nearest global optimum, ruggedness based on entropy, dynamic severity for estimating dynamic properties, fitness cloud for getting a visual rendering of evolvability and gradient for analyzing micro changes of benchmark functions in differential evolution algorithm, we obtain useful results and try to apply effective data, figures and graphs to analyze the performance differential evolution algorithm and make conclusions. Those metrics have great value and more details as DE performance.

Keywords dynamic fitness landscape analysis; differential evolution algorithm; benchmark functions; global optimum

I. INTRODUCTION

There are a variety of evolutionary algorithms that people have applied them to solve some complex problems or improved traditional evolutionary algorithms in order to attain better performance. Evolutionary algorithms (EAs) (T. Back et al. 1997), including Evolutionary Programming (EP), Evolutionary Strategy (ES), Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and so on, is generated from the evolution of nature. Dr. Lawrence J. Fogel (L.J. Fogel et al. 1966) proposed Evolutionary Programming used mutation operators in order to achieve artificial intelligence. There are three mutation operators such as Gaussian (L.J. Fogel et al. 1966), Cauchy (X. Yao et al. 1999) and $L\acute{e}vy$ (C.Y. Lee and X. Yao 2004) in Evolutionary Programming. The mutation operator plays a great role in generating new candidate solutions which we expect better solutions. Evolutionary Strategy (ES) which is very similar to Evolutionary Programming was invented by I. Rechenberg and H. P. Sehwefel in 1963. ES as an approach to solve the optimization of parameters generates candidates which follow the Gaussian distribution of zero mean and a variance based on biological evolution principle. The essential difference between Evolutionary Programming and Evolutionary Strategy is that crossover operators in ES are optional and there is no any crossover operator in EP. In addition, evolutionary operators are selected dynamically based on online learning and fitness landscape analysis (Pietro A. Consoli et al. 2016). Overall, it is more flexible to apply Evolutionary Strategy to solve optimization problems, and using Evolutionary Programming (EP) is easier and more convenient. Genetic Algorithm (EA) proposed by J. Holland in 1975 computes by searching and solves the problems according to the natural evolutional rules of "Survival of the fittest". It seeks a global optimal solution through imitating the natural selection and the genetic mechanism. That a genetic algorithm has strong abilities of searching, concurrency and expansibility makes it be faster to find a global optimal solution. However, the coding process of GA is so complicated that it requires more time to run. Besides, there are many parameters in three operators including selection, crossover and mutation. Differential evolution (DE) (R. Storn and K. Price 1997), one of evolutionary algorithms, proposed by Storn and Price in 1995, is a simple and efficient heuristic for global optimization over continuous spaces. Firstly, the reason why differential evolution has a doughty ability to handle non-differentiable, non-linear and multimodal cost functions is that it was designed to be a stochastic direct search method which is easily applied to experimental minimization where the cost is derived from a physical experimental rather than a computer simulation. Secondly, using a vector population in DE has a great influence in computationally demanding optimizations because the stochastic disturbance of the population vectors can be done independently. Thirdly, the differential evolution is an easy and simple programming due to its self-organizing scheme which employs the difference vector of two randomly selected population vectors to disturb an existing vector. Furthermore, the performance of evolutionary algorithms focuses on the convergence property. And there is no doubt that differential evolution has a great convergence property and the convergence speed is extremely rapid. Particle Swarm Optimization algorithm (PSO), proposed by Kennedy and Eberhart in 1995 (J Kennedy et al. 1995), is a novel intelligent optimization algorithm which tries to simulate social behavior (J Kennedy et al. 1997) of bird flock or fish school. PSO algorithm saves all particles of better solutions because of good memories. In addition, PSO algorithm is achieved easily because PSO doesn't have crossover operation and mutation operation and has fewer parameters.

Recently, fitness landscape are used in many aspects such as search-based software testing problems (Aldeida Aleti et al. 2016), producing approximate solutions of a combinatorial optimization problem through climbing combinatorial fitness landscapes (Matthieu Basseur and Adrien Goeffon 2015). Different evolutionary algorithms are improved from different aspects based on traditional algorithms in order to attain better performance including faster convergence speed, fewer parameters, less running time and so on. But here, we are not going to focus on how to improve or propose new algorithms. On the contrary, we are going to employ fitness landscape analysis to study the performance of differential evolution algorithm in many ways. The concept of fitness landscape has been adopted widely in recent years in many fields. (Vitaly et al. 2011) considered applying fitness landscape analysis in simulation optimization for meta-optimisation purposes and new insights are obtained in the field of fitness landscapes analysis for stochastic problems. (Katherine et al. 2013) proposed using ruggedness, funnels and gradients in the fitness landscapes analyzed and evaluated the performance and the effect of traditional particle swarm optimization (PSO). According to experimental results, these three metrics have valued as part-predictors of PSO performance on unknown problems if used in conjunction with measures approximating other features that have been linked to problem difficulty for PSOs. Meanwhile, they investigated whether a link can be found between problem characteristics and algorithm performance for PSOs. But there are not new insights to show algorithm performance clearly.

In this paper, we are going to apply dynamic fitness landscape to study the performance of traditional differential evolution algorithm through new insights and effective data. The paper is organized as follows: Section II describes traditional differential evolution algorithm. Analyzing how metrics of fitness landscape could evaluate the performance of differential evolution algorithm and find out the correlation in section III. Section IV gives the selected benchmark and evaluation criteria which are based on metrics of dynamic fitness landscape. Experiments are showed through figures and data and experimental results are summarized in section V, followed by conclusion and future directions in section VI.

II. Differential Evolution Algorithm

Differential evolution algorithm, an evolutionary algorithm, attempts to solve global optimization problems of continuous variable. DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, et al. (P Rocca et al. 2011). The main process of differential evolution algorithm is similar with other evolutionary algorithms including mutation, crossover and selection. More details about differential evolution are in the literature 5. Here, the procedure of differential evolution algorithm is illustrated in Algorithm 1.

Algorithm 1 Traditional Differential Evolution Algorithm

Input:
The object function $f: R^n \to R$;
All agents $X = \{x_1, x_2, \dots, x_n\}$;
Random indexes $r_1, r_2, r_3 \in \{1,2,\dots,NP\}$;
The crossover constant $\mathrm{CR} \in [0,1]$;
The differential weight $F \in [0,1]$;
Output:
The best candidate solution;

1. Initialize all agents X with random position in the effective search-space;
2. Repeat
3. For (each agent X in the population) do:
4. Pick three agents r_1, r_2, r_3;
5. Pick a random index between 1 and NP;
6. Compute the agent's potentially new position $y = [y_1, \dots, y_n]$ as follows:
7. For (for every component of the agent) do
8. Copy or not copy based on mutation, crossover and selection;
9. If $f(y) < f(x)$ then replace the agent in the population with the improved candidate solution, that is, replace x with y in the population.
10. Until a termination criterion is met (e.g. number of iterations performed, or adequate fitness reached).

III. Fitness Landscape Analysis

Differential evolution algorithm belongs to one of evolutionary algorithms and is an optimal algorithm. Solving optimization problems with time varying objective functions by methods of evolutionary computation can be grounded on the theoretical framework of dynamic fitness landscapes (Hendrik Richter 2013). The purpose for dynamic fitness landscape analysis is to make people more understand the performance of evolutionary algorithms through effective data and illustrative diagrams. Hence, we are going to analyze the performance of differential evolution algorithm using analysis of topological properties, the variation tendency of gradient and evolvability based on dynamic fitness landscape (Shinichi Shirakawa and Tomoharu Nagao 2016).

A. Fitness Distance Correlation

Fitness distance correlation, as a measure of search difficulty, proposed by Jones and Forrest in 1995 (Jones T and Forrest S 1995), qualifies the relationship between the values taken by the function and the distance to the global optimum value. Given a set $F = \{f_1, f_2, \dots, f_n\}$ of n individual fitness and a corresponding set $D = \{d_1, d_2, \dots, d_n\}$ of the n distances to the nearest global optimum, computing the coefficient *r* as follows:

$$\mathrm{r} = \frac{C_{FD}}{\sigma_F \sigma_D}$$

where

$$C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})$$

is the covariance of F and D, and $\sigma_F, \sigma_D, \bar{f}$ and $\bar{d}$ are the standard deviations and means of F and D respectively. Fitness distance correlation is acted as our measure of problem difficulty and is considered as an analysis of topological properties on dynamic fitness landscape.

B. Ruggedness

Vassilev et al. (V. K. Vassilev et al. 2003) proposed the first entropic measure (FEM) which is as a measure of ruggedness in regard to neutrality for discrete problems and was applied on real-valued problems (K. M. Malan and A. P. 2009). Besides, ruggedness can quantify for constrained continuous fitness landscapes (Shayan Poursoltan and Frank Neumann 2015). Here, a random walk is used to calculate a measure for ruggedness on fitness landscape. The random walk (Hendrik Richter 2013) includes the length T and the step size t_S. Therefore, we can get a random walk (Stefan Nowak and Joachim Krug 2015) on the fitness landscape as

$$x(j+1) = x(j) + t_S \cdot rand$$

Here, if the step size is 10% of the problem's domain, it is named as micro ruggedness ($FEM_{0.1}$). And if the step size is 50% of the problem's domain, it is called as macro ruggedness ($FEM_{0.5}$) (Katherine M. Malan et al. 2014).

Table 1. Benchmark Functions

Function Name	Definition	Domain	Global Optimum
Ackley	$f_{ack}(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i{}^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$x_i \in [-32,32]$	$f_{ack}^* = f_{ack}(0,\dots,0) = 0$
Beale	$f_{bea}(x_1,x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$x_i \in [-4.5,4.5]$	$f_{bea}^*(3,0.5) = 0$
Goldstein -Price	$f_{gp}(x_1,x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$x_i \in [-2,2]$	$f_{gp}^*(0,1) = 3$
Griewank	$f_{gew}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$x_i \in [-600,600]$	$f_{grw}^* = f_{grw}(0,\dots,0) = 0$
Quadric (Schwefel 1.2)	$f_{qdr}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	$x_i \in [-100,100]$	$f_{qdr}^* = f_{qdr}(0,\dots,0) = 0$
Quartic	$f_{qrt}(x) = \sum_{i=1}^{D} ix_i^4$	$x_i \in [-1.28,1.28]$	$f_{qrt}^* = f_{qrt}(0,\dots,0) = 0$
Rastrigin	$f_{ras}(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$x_i \in [-5.12,5.12]$	$f_{ras}^* = f_{ras}(0,\dots,0) = 0$
Rosenbrock (generalized)	$f_{ros}(x) = \sum_{i=1}^{D-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), D \geq 2$	$x_i \in [-2.048,2.048]$	$f_{ros}^* = f_{ros}(0,\dots,0) = 0$
Salomon	$f_{sal}(x) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2} + 1$	$x_i \in [-100,100]$	$f_{sal}^* = f_{sal}(0,\dots,0) = 0$
Schwefel 2.22	$f_{sch2.22}(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	$x_i \in [-10,10]$	$f_{sch2.22}^* = f_{sch2.22}(0,\dots,0) = 0$
Schwefel 2.26	$f_{sch2.26}(x) = -\sum_{i=1}^{D}\left(x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$x_i \in [-500,500]$	$f_{sch2.26}^* = f_{sch2.26}(420.9687,\dots,420.9687) = -12569.5$
Spherical	$f_{sph}(x) = \sum_{i=1}^{D} x_i{}^2$	$x_i \in [-100,100]$	$f_{sph}^* = f_{sph}(0,\dots,0) = 0$

Record the fitness value as time series:

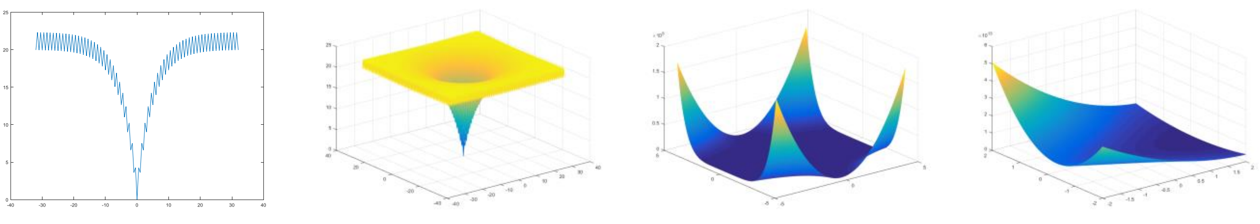
$$f(j,k) = f(x(j),k), j = 1,2,\dots,T$$

where k is time. And $r(t_L,k)$ is defined as the spatial correlation which can be obtained from the autocorrelation function of the time series with time lag t_L. Random walk correlation function is defined as follows:

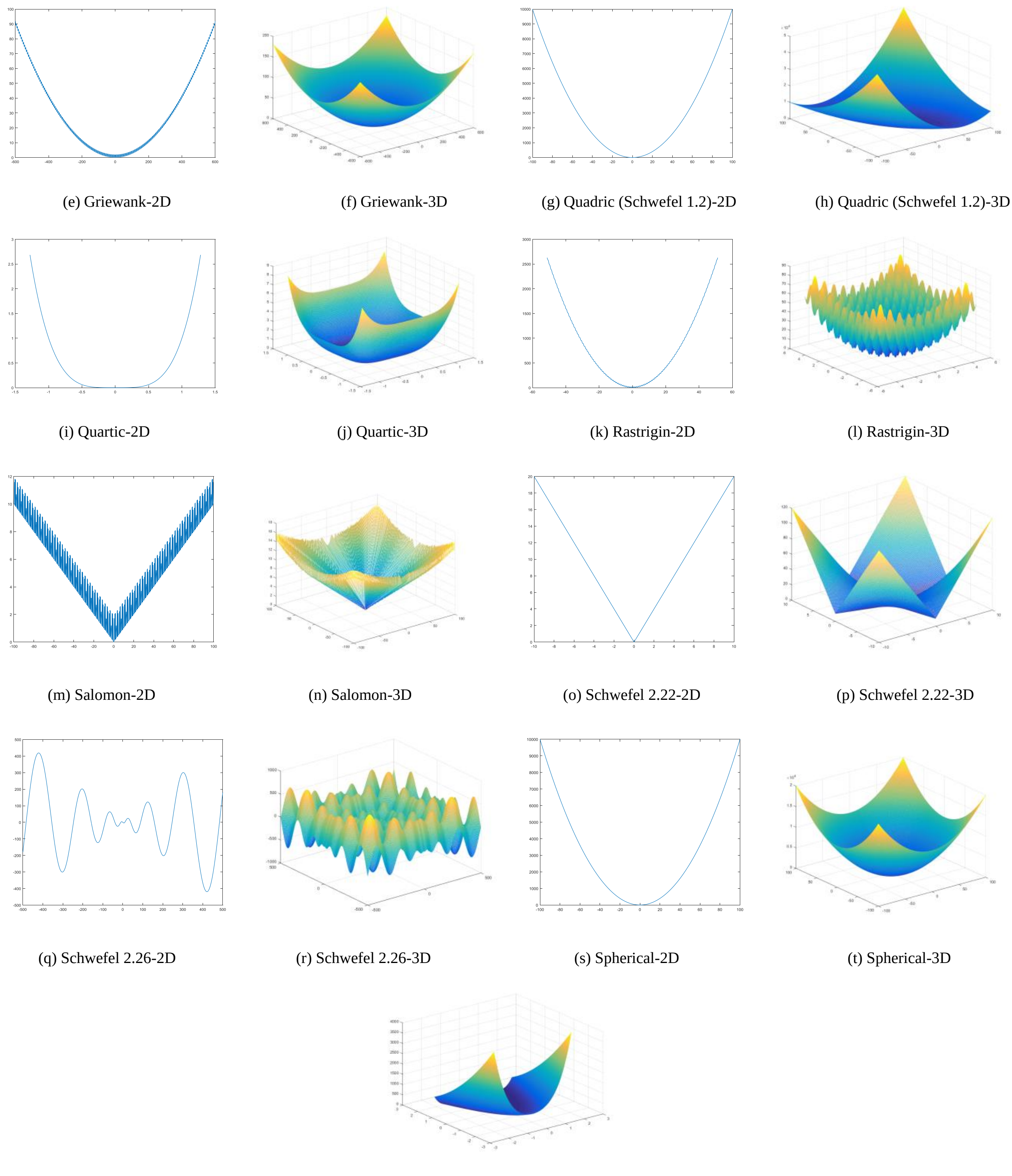
$$r(t_L,k) = \frac{\sum_{j=1}^{T-t_L}\left(f(j,k) - \bar{f}(k)\right)\left(f(j+t_L,k) - \bar{f}(k)\right)}{\sum_{j=1}^{T}\left(f(j,k) - \bar{f}(k)\right)^2}$$

where $\bar{f}(k) = \frac{1}{T}\sum_{j=1}^{T} f(j,k)$ and $T \gg t_L > 0$.

Ruggedness is a measure of analysis of topological properties on dynamic fitness landscape. It is not practical to use modality to calculate a given fitness landscape, although it can be work in theory. Hence, we choose ruggedness as a measure to evaluate a part of performance of differential evolution algorithm.



(a) Ackley-2D (b) Ackley-3D (c) Beale-3D (d) Goldstein-Price-3D



(e) Griewank-2D (f) Griewank-3D (g) Quadric (Schwefel 1.2)-2D (h) Quadric (Schwefel 1.2)-3D

(i) Quartic-2D (j) Quartic-3D (k) Rastrigin-2D (l) Rastrigin-3D

(m) Salomon-2D (n) Salomon-3D (o) Schwefel 2.22-2D (p) Schwefel 2.22-3D

(q) Schwefel 2.26-2D (r) Schwefel 2.26-3D (s) Spherical-2D (t) Spherical-3D

(u) Rosenbrock-3D

Figure 1. Two dimension or three dimension of benchmark functions

C. Dynamic Severity

Analysis of dynamical properties including change frequency and dynamic severity plays a very important role in evaluating the performance of evolutionary algorithms. Dynamic severity is one of features that influences on the success that an evolutionary search has in solving the corresponding dynamic optimization problem (Hendrik Richter 2013). The corresponding dynamic optimization problem is defined as follows

$$f_S(k) = \max_{x \in S} f(x, k), \quad \forall k \geq 0$$

which $f_S(k)$ is the temporarily highest fitness, S is the search space and k is the landscape time. The solution trajectory of $f_S(k)$ reads

$$x_S(k) = arg f_S(k), \quad \forall k \geq 0$$

Then, put dynamic severity as

$$\eta(k+1) = \|x_S(k+1) - x_S(k)\|$$

We calculate the time average severity as follows

$$\eta = \lim_{K\to\infty} \sum_{K=0}^{K-1} \eta(k)$$

Dynamic severity tried to estimate the dynamic property which is related with some notations (Weicker, K 1917; Branke, J 2001; Richter, H 2005). Therefore, we are going to apply dynamic severity to analyze changes of differential evolution algorithm.

D. Fitness Cloud

Fitness cloud is one of dynamic fitness landscape analysis to try to get a visual rendering of evolvability through between parent-fitness and offspring-fitness. Generally speaking, fitness cloud includes three aspects of FC_{max}, FC_{min} and FC_{mean} (Sebastien et al. 2007). In this paper, we focus on FC_{mean} of every fitness value. The formula is illustrated as follows:

$$\bar{f}_{mean}(F) = \left(1 - \frac{K+1}{N}\right) f + \left(\frac{K+1}{N}\right) \beta$$

Where NK-landscape with $N = 25, K = 25$ and $K = 5$. Meanwhile, the β fitness level is always set to 0.5.

E. Gradient Measures

Gradient measures are applied to analyze micro behaviors of evolutionary algorithms through fitness values. We apply random walk to decide which steps should be selected. Set fitness function *f*, begin at solution vector $x(t)$ with T steps of equal distance will record in the following sequence of *T*+1 points: $x(t), x(t+1), \dots, x(t+T)$. Estimating fitness gradients reads:

$$g(t) = \frac{f\big(x(t+1)\big) - f\big(x(t)\big)}{x(t+1) - x(t)}$$

Here, we can get a sequence of fitness gradients:

$$g(t), g(t+1), \dots, g\big(t + (T-1)\big)$$

The average gradient is defined as:

$$G_{avg} = \frac{\sum_{t=0}^{T-1} |g(t)|}{T}$$

Furthermore, the standard deviation of gradient measures from the mean would give an indication of how much gradient measures on a walk differ from the average and can be defined as (Katherine M. Malan et al. 2013):

$$G_{dev} = \sqrt{\frac{\sum_{t=0}^{T-1} \big(G_{avg} - |g(t)|\big)^2}{T-1}}$$

All of measures are considered and used in the experiment by differential evolution algorithm.

IV. Experimental Preparations

This section describes benchmark functions which are used in experiment in order to test the performance of differential evolution algorithm. And some useful and basic metrics for evaluating a particular algorithm would be elaborated.

A. Benchmark Functions

Table 1 shows the benchmark functions which are going to use in differential evolution as testing functions. Here, 12 benchmark functions are chosen including Ackley (X.Yao et al. 1999), Beale (S. K. Mishra 2006), Goldstein-Price (X.Yao et al. 1999), Griewank (X.Yao et al. 1999), Quadric (Schwefel 1.2) (X.Yao et al. 1999), Quartic (X.Yao et al. 1999), Rastrigin (X.Yao et al. 1999), Rosenbrock (generalized) (X.Yao et al. 1999), Salomon (K.V. Price et al. 2005), Schwefel 2.22 (X.Yao et al. 1999), Schwefel 2.26 (X.Yao et al. 1999) and Spherical (K. A. De Jong 1975). We can see figure 1 that Ackley, Griewank, Rastrigin, Salomon and Schwefel 2.26 are rugged. On the contrary, Quartic and Spherical are smooth. Hence, different modalities will show different influence to measure the performance of differential evolution algorithm.

B. Metrics of algorithm performance

Using dynamic fitness landscapes attempts to evaluate the performance of differential evolution algorithm through some effective metrics. Some basic metrics must be used like the success rate which is referred to the literature 9. Besides, the processing capacity and the standard deviation are also applied in experiment.

1) SRate: The success rate (SRate) is defined as the number of successful runs that reach a solution within the fixed accuracy level of the global optimum divided by the total number of runs (P. N. Suganthan et al. 2005). The success rate is a value in the range from 0 to 1. And when the value is equal to 1, it means the highest possible rate of success. The formula is as follows:

$$\text{SRate} = \frac{ST}{ST + FT}$$

Where *ST* means success times, and *FT* represents failure times.

2) PC: The processing capacity function can evaluate the ability of processing with a specified algorithm in a particular environment. And the formula reads:

$$\text{PC} = \frac{STime}{RTime}$$

Here, *STime* represents successful time which differential evolution algorithm spends. *RTime* means running time. Processing capacity would be calculated when each benchmark function was tested in differential evolution algorithm.

3) SDev: The standard deviation can reflect the degree of dispersion of a data set. Although there are the same mean of different data sets, standard deviations are not always same. Set a dataset as $X = \{x_1, x_2, \dots, x_N\}$, the standard deviation is defined as:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}$$

Therefore, the standard deviation can show whether operations of mutation, selection and crossover affect how to reach the global optimum. Maybe those values which evolutionary algorithms calculate keep close or discrete distance. The assumption is necessary to prove through experiment.

TABLE 2. Fitness landscape metrics according to differential evolution algorithm for 12 benchmark functions in table 1. (D = dimension, C_{FD} = fitness distance correlation, $r_{FEM_{0.1}}$ = micro ruggedness, $r_{FEM_{0.5}}$ = macro ruggedness, $\bar{f}_{mean}$ = the mean fitness cloud, G_{avg} = the average gradient, G_{dev} = the standard deviation of gradient, SRate = the success rate, PC = the processing capacity.)

Function	D	C_{FD}	$r_{FEM_{0.1}}$	$r_{FEM_{0.5}}$	$\bar{f}_{mean}$	G_{avg}	G_{dev}	SRate	PC
Ackley	2	0.000	0.000	0.000	0.000	0.000	0.000	0.667	0.9659
	3	0.137	0.588	0.586	0.040	0.005	0.065	0.957	0.952
	5	483.731	0.803	0.806	4.110	0.027	0.150	0.910	0.908
	15	96859.354	0.887	0.888	96.491	0.054	0.220	0.717	0.719
	30	1591188.371	0.958	0.956	777.078	0.059	0.235	0.207	0.213
Beale	3	0.000	0.840	0.839	0.001	0.000	0.000	0.947	0.943
Goldstein-Price	3	1.766	0.515	0.522	3.166	0.001	0.035	0.953	0.954
Griewank	2	0.138	0.669	0.659	0.018	0.001	0.038	0.7625	0.693
	3	0.996	0.364	0.362	0.034	0.001	0.025	0.779	0.872
	5	74.353	0.611	0.607	0.565	0.013	0.107	0.850	0.850
	15	4727.634	0.882	0.893	10.157	0.775	0.000	0.603	0.602
	30	37127.610	0.976	0.972	67.537	1.087	0.000	0	NAN
Quadric (Schwefel 1.2)	2	0.001	0.013	0.011	0.000	0.000	0.000	0.762	0.771
	3	0.332	0.633	0.635	0.025	0.000	0.018	0.586	0.592
	5	1.668	0.861	0.870	0.170	0.002	0.040	0	NAN
	15	27.255	0.929	0.932	0.953	0.012	0.107	0	NAN
	30	158.887	0.962	0.958	2.642	NaN	NaN	0	NAN
Quartic	2	0.000	0.023	0.446	0.000	0.00	0.00	0.939	0.943
	3	0.000	0.504	0.504	0.000	0.000	0.001	0.955	0.951
	5	0.001	0.701	0.729	0.002	0.000	0.015	0.798	0.801
	15	0.0185	0.944	0.945	0.015	0.000	0.020	0.580	0.578
	30	0.058	0.941	0.944	0.043	0.001	0.025	0	NAN
Rastrigin	2	0.000	0.349	0.602	0.000	0.000	0.003	0.905	0.907
	3	0.003	0.685	0.681	0.001	0.000	0.015	0.592	0.600
	5	0.130	0.883	0.893	0.036	0.001	0.026	0.417	0.410
	15	0.576	0.981	0.981	0.271	0.003	0.057	0	NAN
	30	1.097	0.997	0.991	0.388	0.004	0.061	0	NAN
Rosenbrock (generalized)	3	0.000	0.425	0.425	0.001	0.000	0.000	0.867	0.877
	5	0.002	0.517	0.489	0.003	0.000	0.006	0.7982	0.800
	15	0.054	0.900	0.894	0.029	0.003	0.056	0.503	0.504
	30	0.200	0.964	0.962	0.100	0.001	0.038	0	NAN
Salomon	2	0.011	0.858	0.862	0.014	0.000	0.019	0.750	0.775
	3	0.170	0.880	0.881	0.043	0.001	0.027	0.496	0.509
	5	4.111	0.860	0.859	0.342	0.004	0.061	0	NAN
	15	84.600	0.961	0.959	2.385	0.036	0.176	0	NAN
	30	264.740	0.973	0.973	7.885	0.052	0.218	0	NAN
Schwefel 2.22	2	0.000	0.614	0.629	0.001	0.000	0.004	0.837	0.844
	3	0.014	0.831	0.854	0.012	0.000	0.017	0.498	0.508
	5	0.124	0.932	0.931	0.145	0.001	0.025	0	NAN

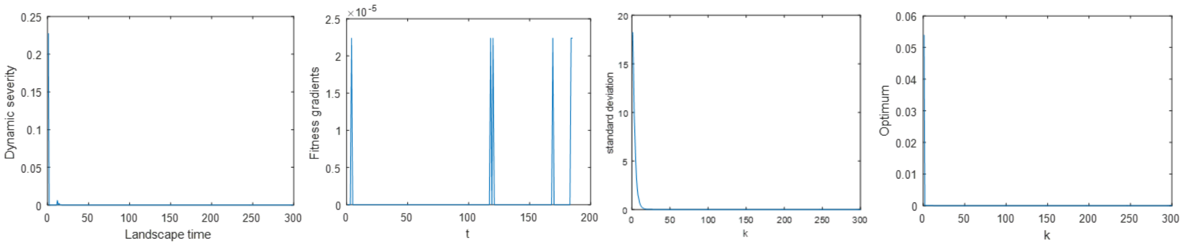
	15	1.315	0.973	0.978	0.576	0.004	0.066	0	NAN
	30	1.935	0.989	0.982	1.421	0.003	0.059	0	NAN
Schwefel 2.26	2	0.020	0.466	0.474	-831.388	0.000	0.000	0.967	0.967
	3	1075.065	0.697	0.699	-836.224	0.087	0.000	0.900	0.902
	5	44328.505	0.925	0.934	-2028.428	1.233	0.000	0.747	0.758
	15	1014835.697	0.953	0.968	-4581.603	10.767	0.000	0	NAN
	30	430334.571	0.955	0.958	-5338.474	5.666	0.000	0	NAN
Spherical	2	0.030	0.500	0.498	0.013	0.001	0.023	0.980	0.980
	3	97.311	0.308	0.307	0.805	0.029	0.000	0.907	0.911
	5	32352.805	0.896	0.895	36.139	0.228	0.000	0.717	0.723
	15	10501477.655	0.861	0.860	1106.768	29.343	0.000	0	NAN
	30	125514525.865	0.936	0.948	6856.253	92.203	0.000	0	NAN

V. Experiments and Results

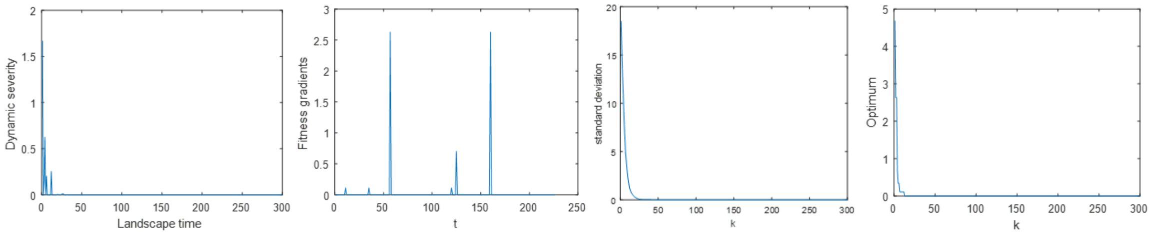
Figure 1 shows two-dimension or three-dimension of 12 benchmark functions. Every function has their special properties which some of them are very rugged because of many mountain peaks and some of them are very smooth. Here, we are going to test performance of differential evolution algorithm when benchmark functions are in different dimensions. The experimental results show in table 2. Here, we use fitness landscape metrics to evaluate the performance of traditional differential evolution algorithm and try to figure out correlations between different results.

We can see clearly differential evolution algorithm can not get 100% correct to obtain the global optimum values of benchmark functions. Particularly, when benchmark functions are in high dimensions, differential evolution algorithm doesn't calculate correct optimum basically or fall into local optimum. For example, when Griewank function is in 30 dimension, differential evolution algorithm can not calculate its optimum value. And we can see through Figure 5, the optimum is very close to 0, but while iterations arrive the maximum value, differential evolution algorithm has no ability to get the optimum. There are same situations like other benchmark functions including Quadric (Schwefel 1.2), Quartic, Rastrigin, Rosenbrock (generalized), Salomon, Schwefel 2.22, Schwefel 2.26 and Spherical. Fitness distance correlation reflects how values of benchmark functions are close to global optimum values. We could see when the fitness distances are very short and it is much close to the global optimum value. But when a benchmark function is so ruggedness, it may influence the relationship between fitness distance and global optimum values. $r_{FEM_{0.1}}$ and $r_{FEM_{0.5}}$ are attempt to describe shapes of benchmark functions. In most cases, that shapes are not flat is hard to obtain a higher success rate. There are some differences between ruggedness and gradient. As we all know, gradients focus on micro changes. Here, we need to separate different small parts to calculate values of gradients. In general, when dimensions are high, each benchmark function of average gradients and standard deviation of gradients are smaller, but some of them are exceptive such as the benchmark function Schwefel 2.26. If values of gradient are changed frequently, it means it is very difficult to close global optimum values. Sometimes, it will offer deceptive directions and lead to wrong ways. Therefore, maybe it will take much time to find out a global optimum or it can't get a global optimum any more. Besides, we can see results of ruggedness that we set two different effective space for each benchmark function. Generally speaking, ruggedness and success rate have very close relationship. When a value of ruggedness is much bigger, success rate is lower maybe it will be zero. And the performance of differential evolution algorithm is not very well because it is really hard to find a global optimum value and it is possible to stay in a local optimum. We believe each metric must interact and we try to combine some metrics to find something new and analyze them. We could see when some benchmark functions are rough like Rastrigin and Schwefel 2.26 that the values of ruggedness and dynamic severity are bigger. And it is a little difficult to find their own global optimum.

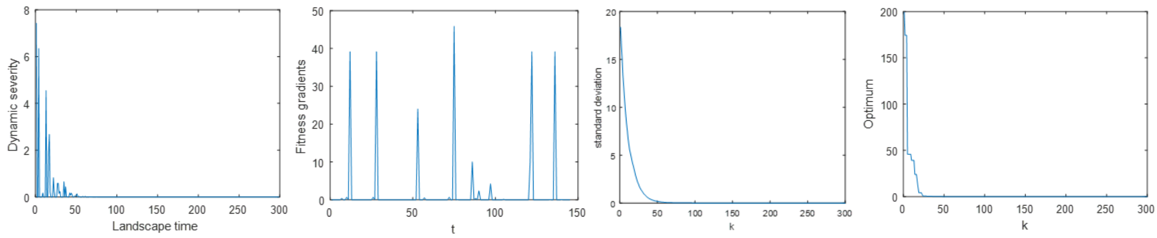
From Figure 2 to Figure 13, we could see clearly that dynamic severity, fitness gradients, standard deviation and optimum have deep intimate connection which we can infer optimum through other three metrics. For example, in (e) of figure 6, when dynamic severity and fitness gradients change frequently, it is obvious that Quadric is hard to get optimum using differential evolution algorithm. Meanwhile, in (a) and (b) of figure 1, Ackley function is comparatively smooth with other benchmark functions. In table 2, differential evolution algorithm can easily calculate optimum of Ackley function in different dimensions except 30 dimension. On the other, in (c) to (e) of figure 10, the experimental results show differential evolution algorithm can not calculate optimum value and we could see the value of dynamic severity change constantly. Therefore, after several generations, optimum of salomon benchmark function is very close to the global optimum, but it can not still get the global optimum in the end. There are enough experimental results and data which show clearly differential evolution algorithm can greatly deal with some benchmark functions which are smooth and low dimensions. Applying dynamic fitness landscape metrics to illustrate performance of genetic algorithms is very clear, because people can see the experimental results and performance of genetic algorithms through graphs.



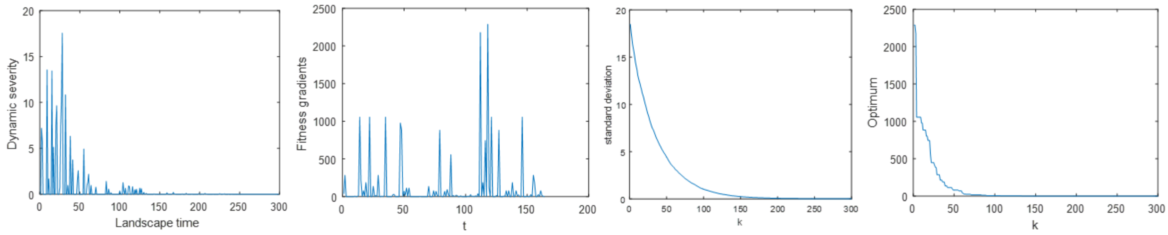
(a) Ackley-2D



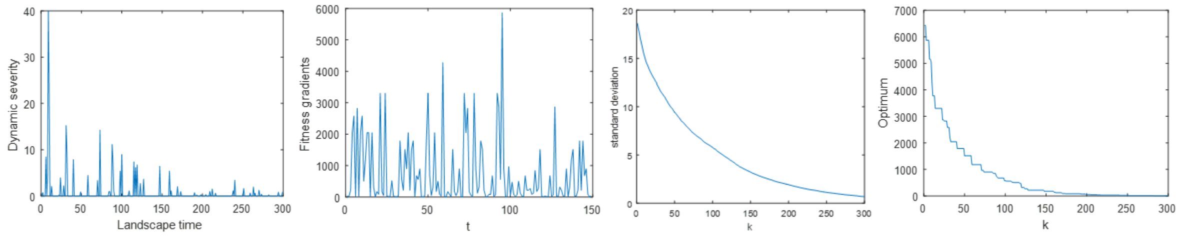
(b) Ackley-3D



(c) Ackley-5D



(d) Ackley-15D



(e) Ackley-30D

Figure 2. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Ackley function change dynamically based on different dimensions

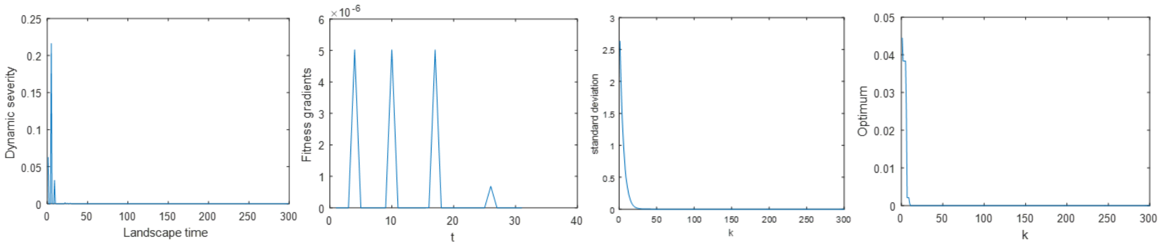


Figure 3. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Beale function change dynamically based on three-dimension

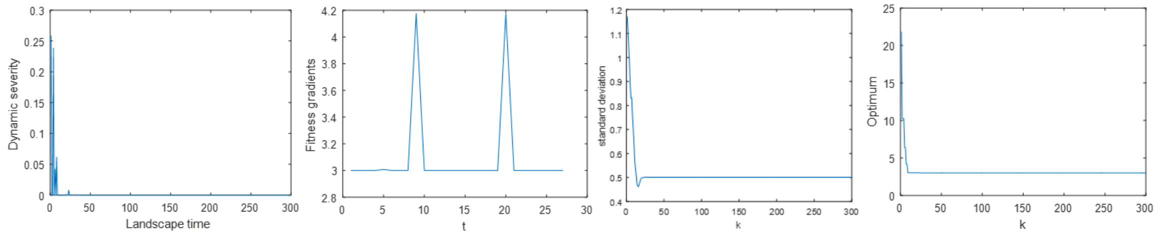
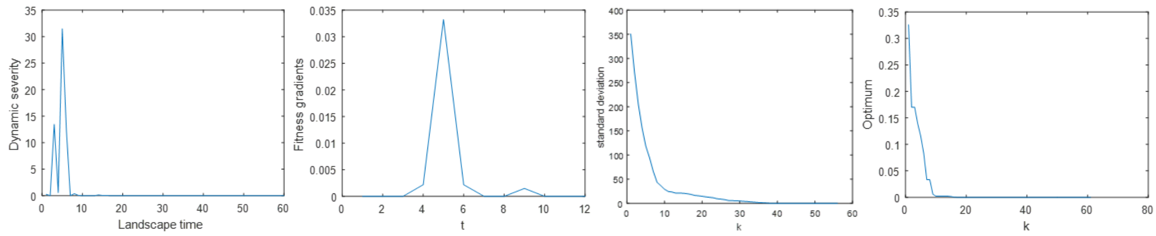
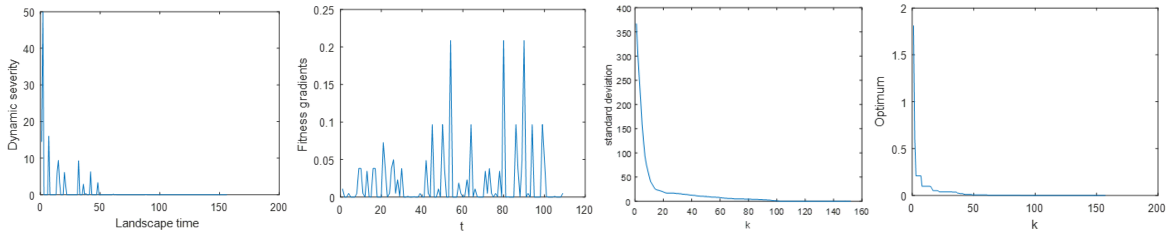


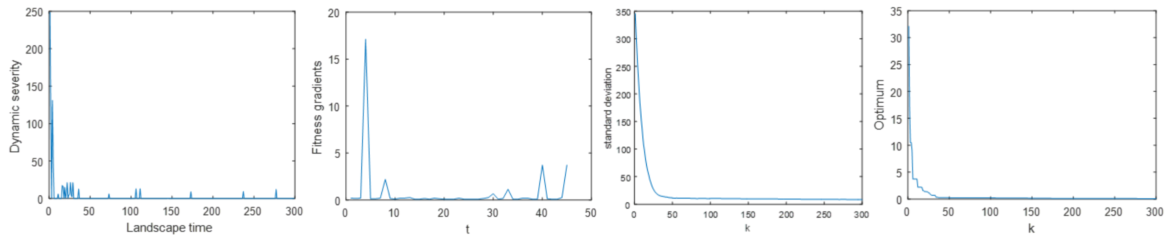
Figure 4. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Goldstein-Price function change dynamically based on three-dimension



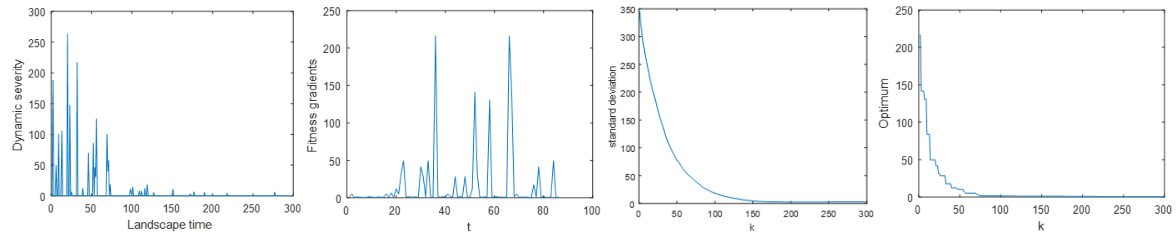
(a) Griewank-2D



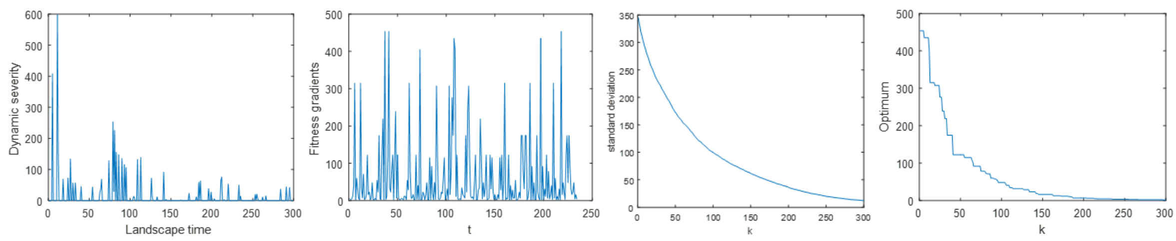
(b) Griewank-3D



(c) Griewank-5D

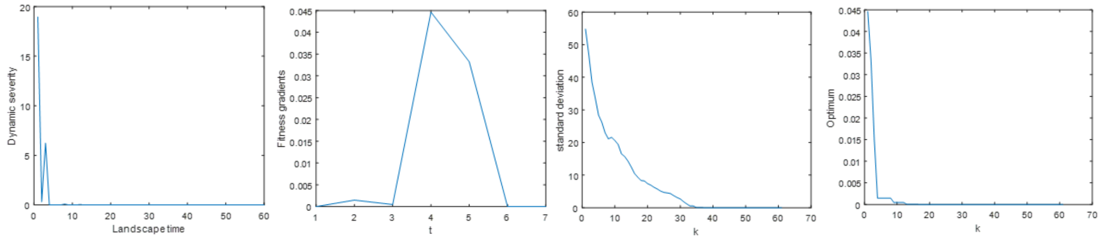


(d) Griewank-15D

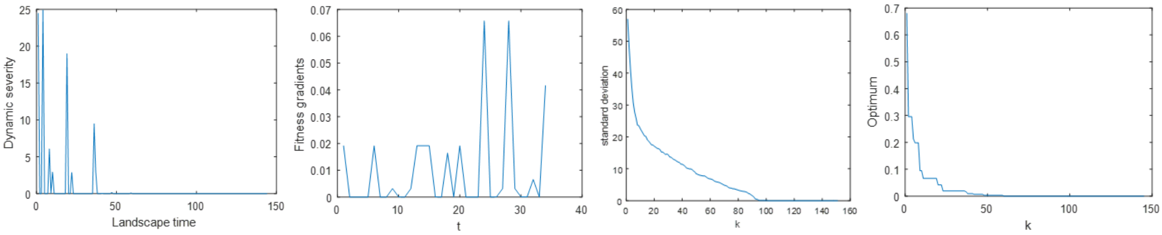


(e) Griewank-30D

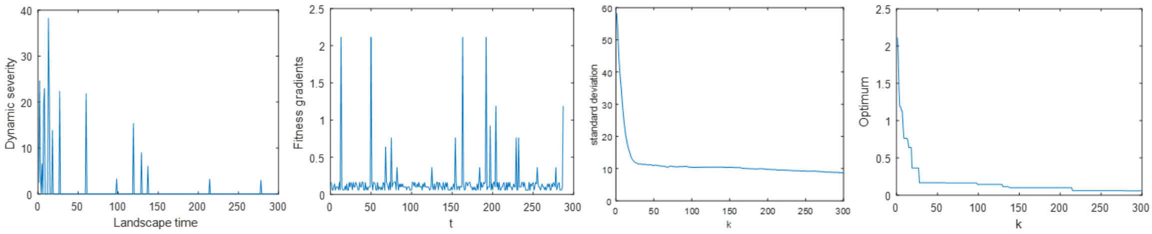
Figure 5. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Griewank function change dynamically based on different dimensions



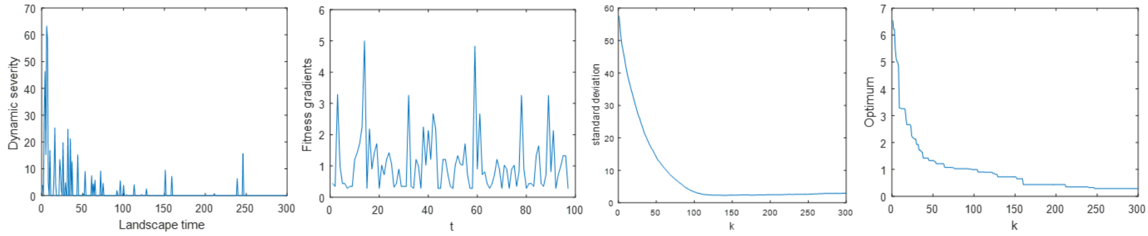
(a) Quadric-2D



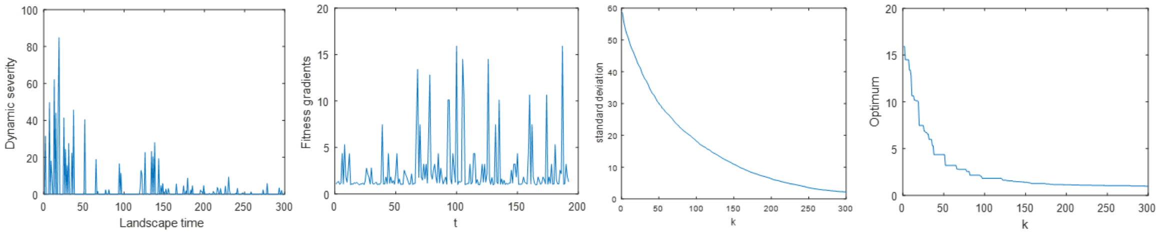
(b) Quadric-3D



(c) Quadric-5D

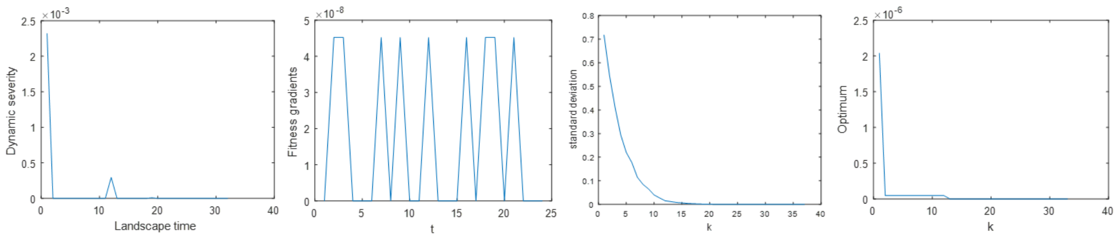


(d) Quadric-15D

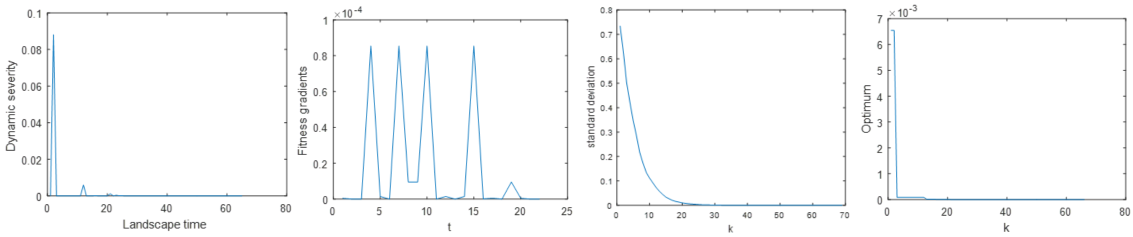


(e) Quadric-30D

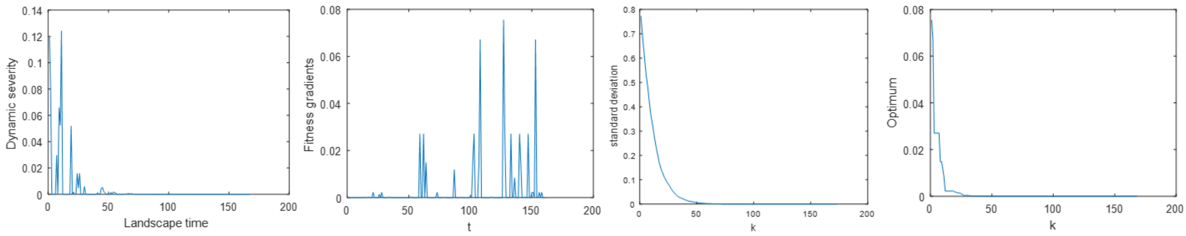
Figure 6. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Quadric function change dynamically based on different dimensions



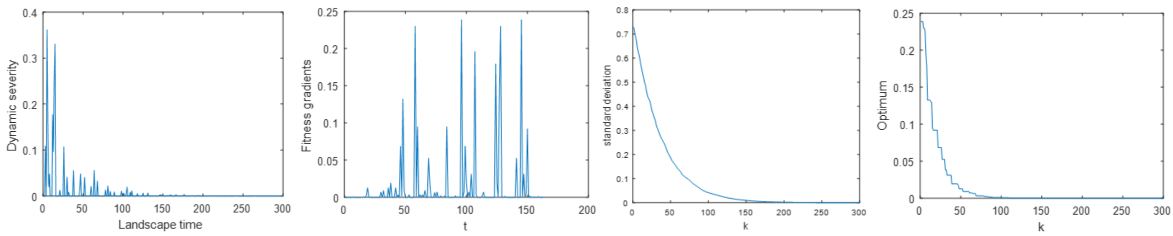
(a) Quartic-2D



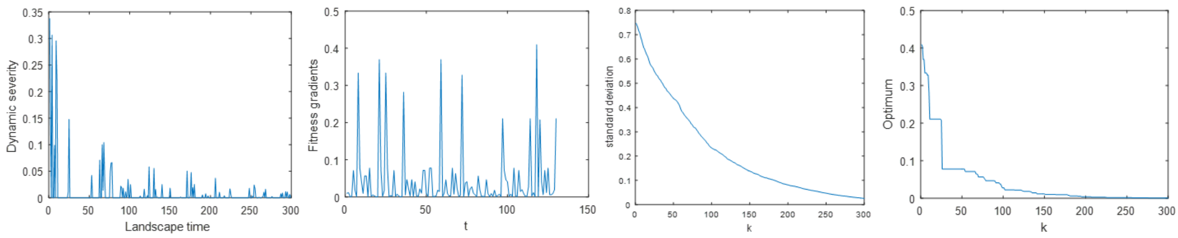
(b) Quartic-3D



(c) Quartic-5D

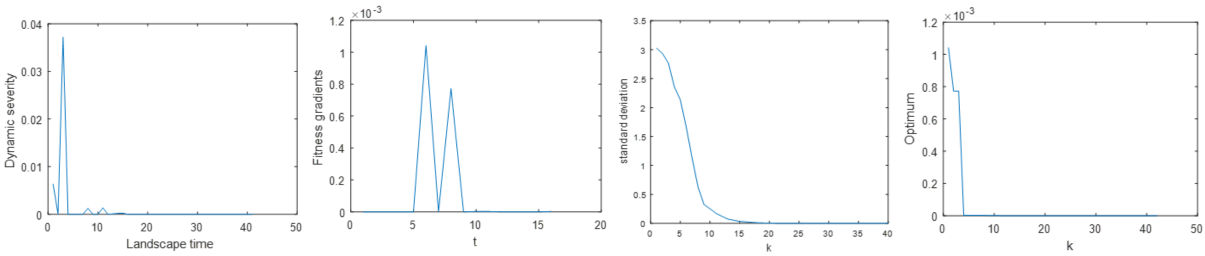


(d) Quartic-15D

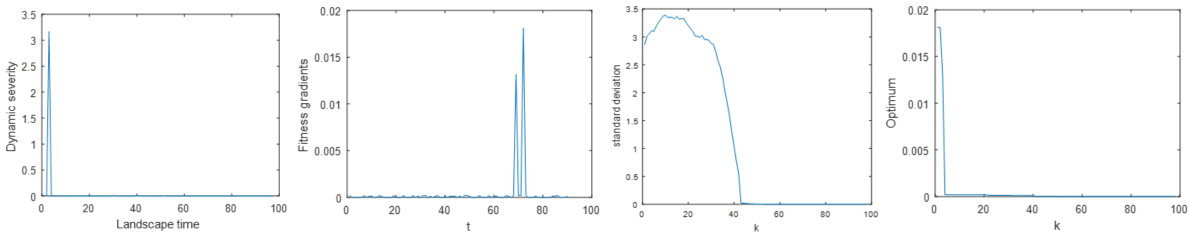


(e) Quartic-30D

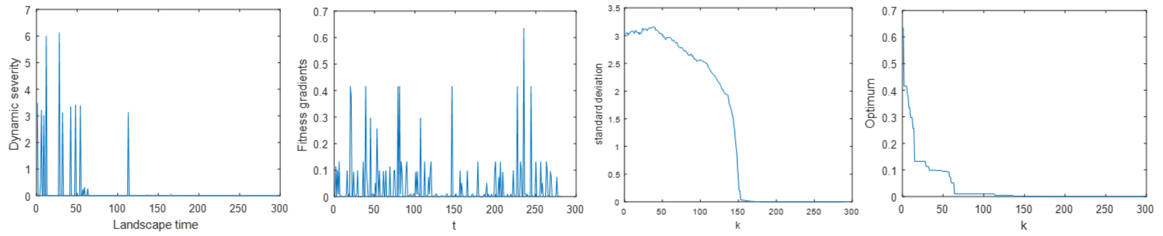
Figure 7. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Quartic function change dynamically based on different dimensions



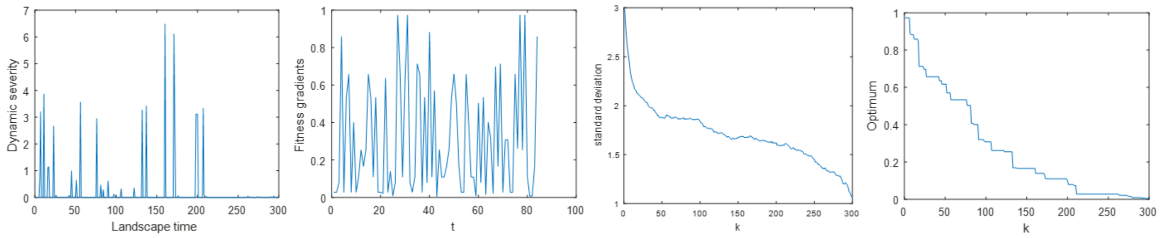
(a) Rastrigin-2D



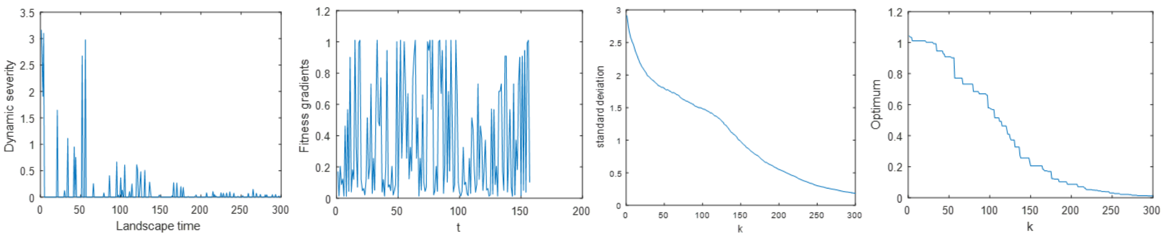
(b) Rastrigin-3D



(c) Rastrigin-5D

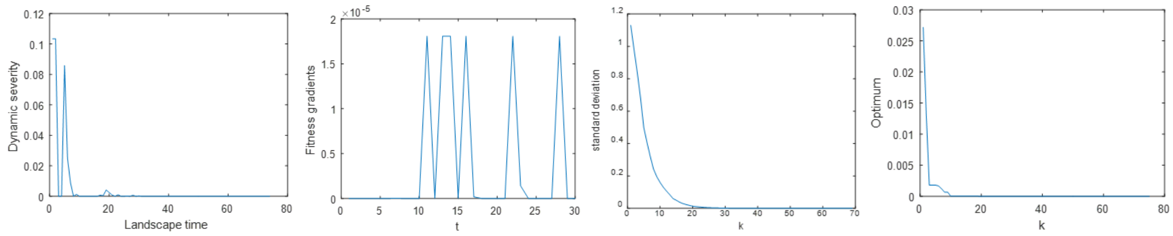


(d) Rastrigin-15D

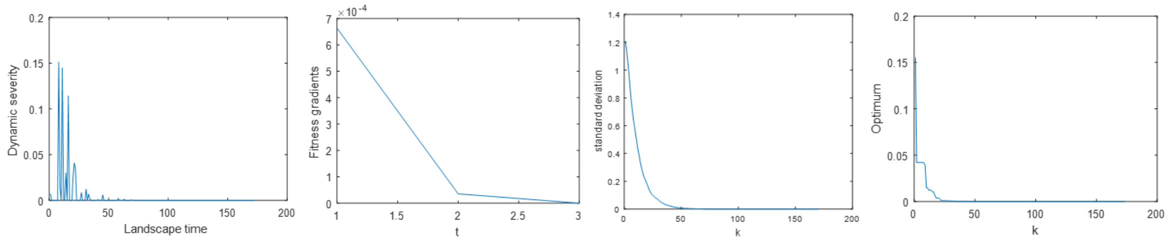


(e) Rastrigin-30D

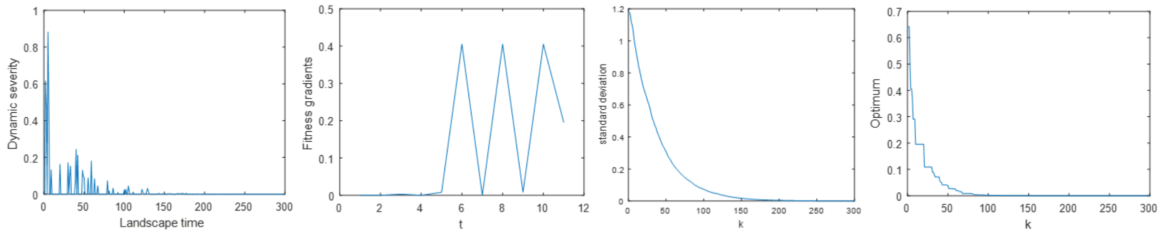
Figure 8. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Rastrigin function change dynamically based on different dimensions



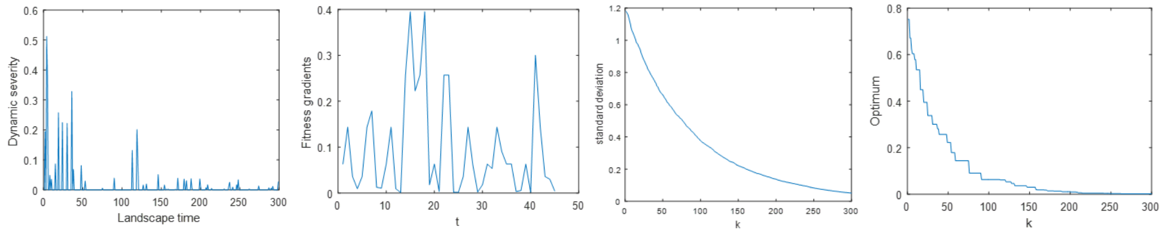
(b) Rosenbrock-3D



(c) Rosenbrock-5D

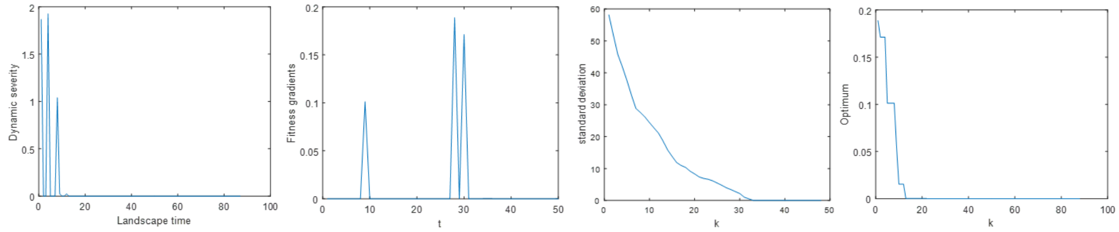


(d) Rosenbrock-15D

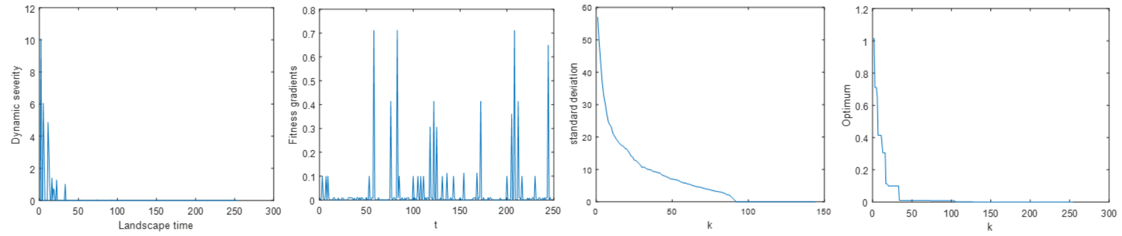


(e) Rosenbrock-30D

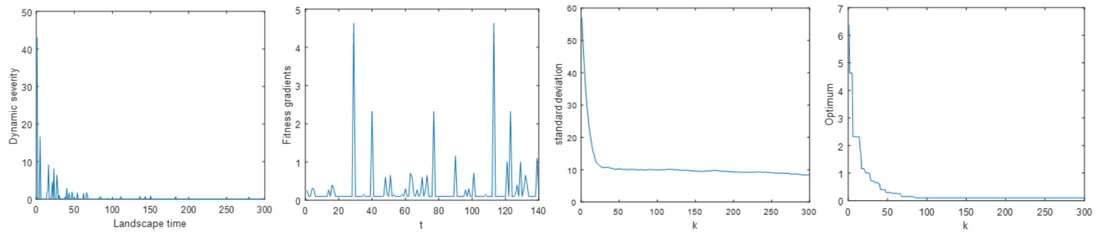
Figure 9. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Rosenbrock function change dynamically based on different dimensions



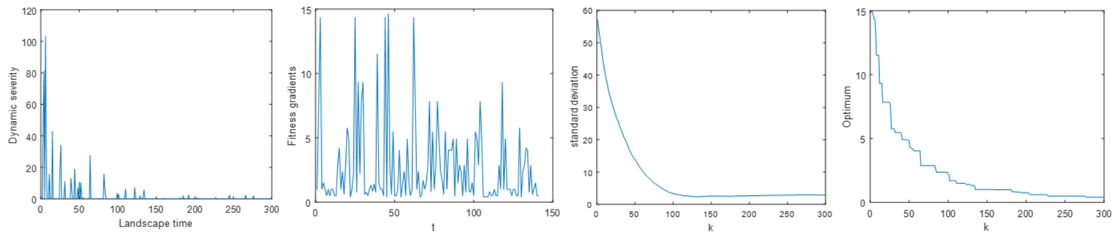
(a) Salomon-2D



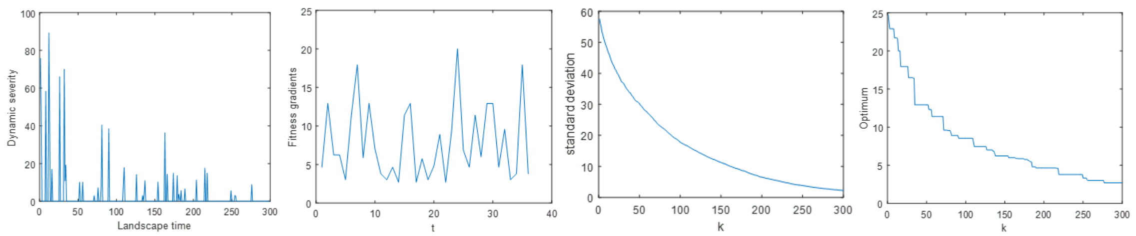
(b) Salomon-3D



(c) Salomon-5D

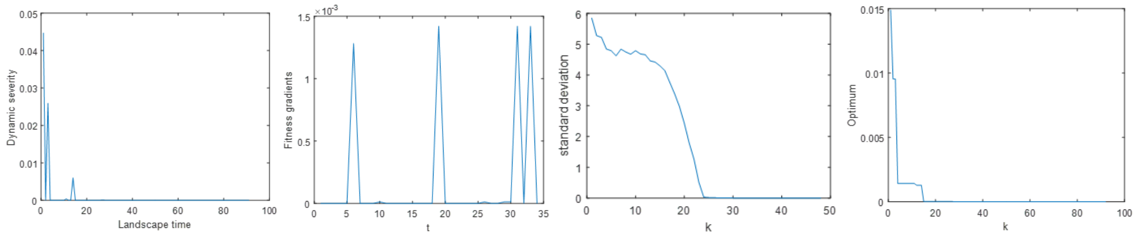


(d) Salomon-15D

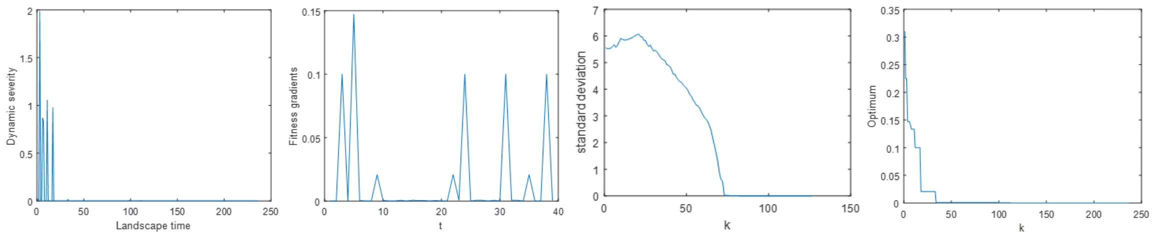


(e) Salomon-30D

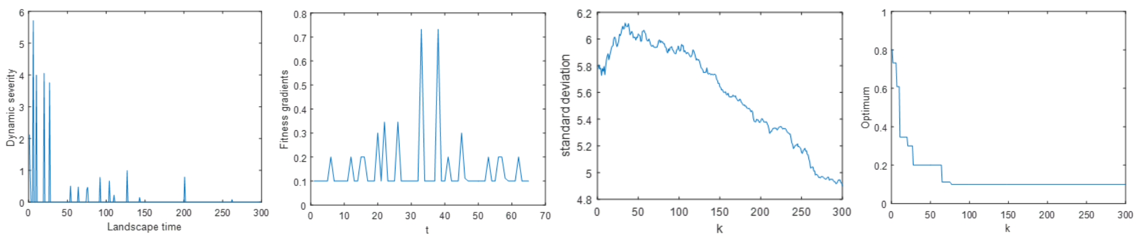
Figure 10. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Salomon function change dynamically based on different dimensions



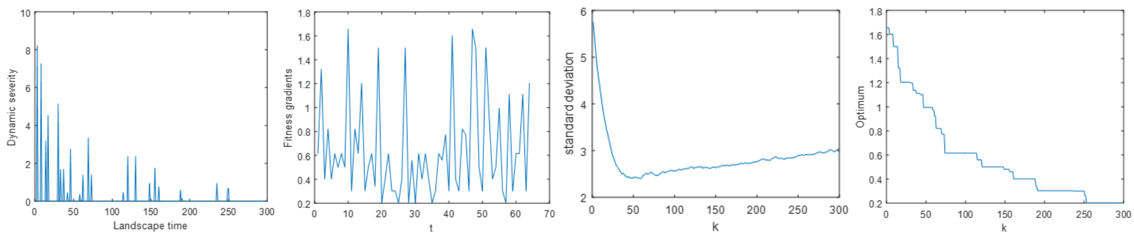
(a) Schwefel 2.22-2D



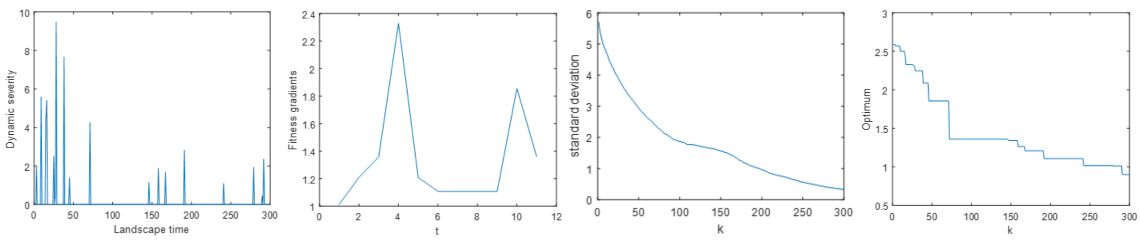
(b) Schwefel 2.22-3D



(c) Schwefel 2.22-5D

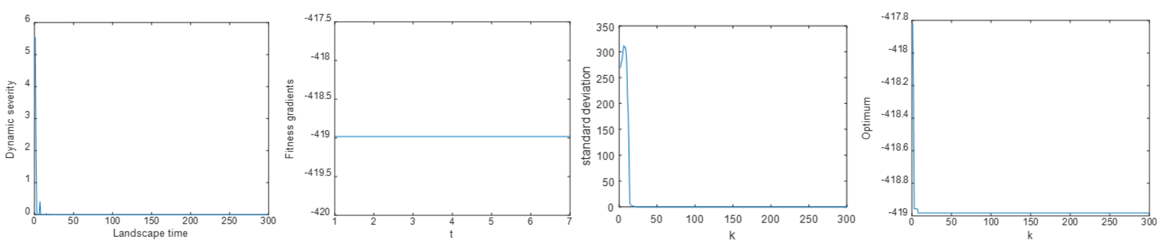


(d) Schwefel 2.22-15D

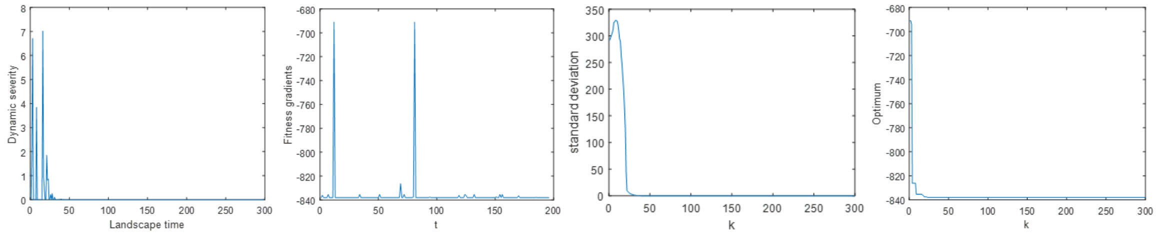


(e) Schwefel 2.22-30D

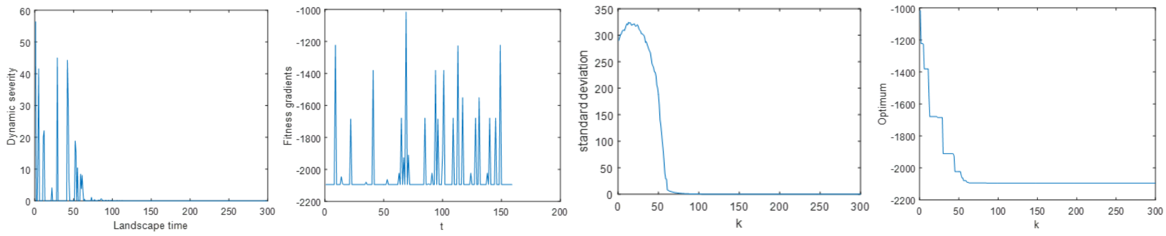
Figure 11. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Schwefel 2.22 function change dynamically based on different dimensions



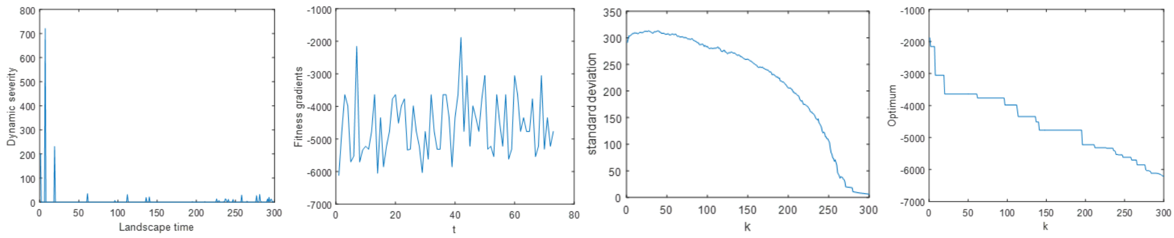
(a) Schwefel 2.26-2D



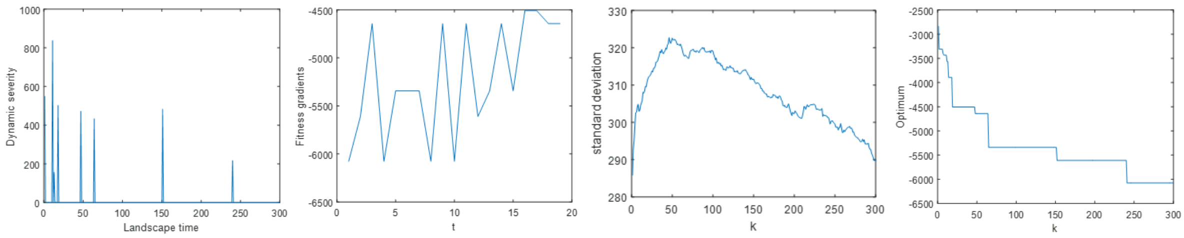
(b) Schwefel 2.26-3D



(c) Schwefel 2.26-5D

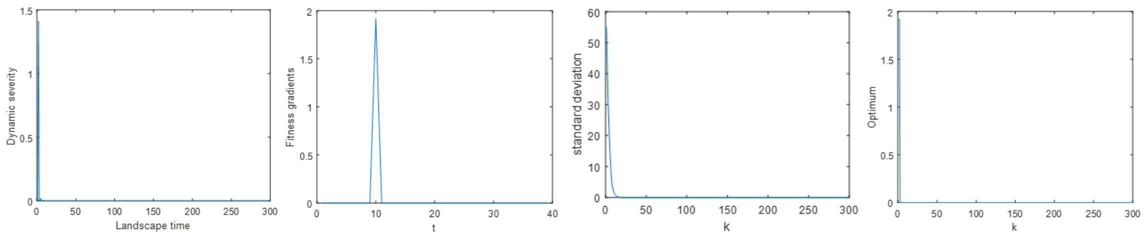


(d) Schwefel 2.26-15D

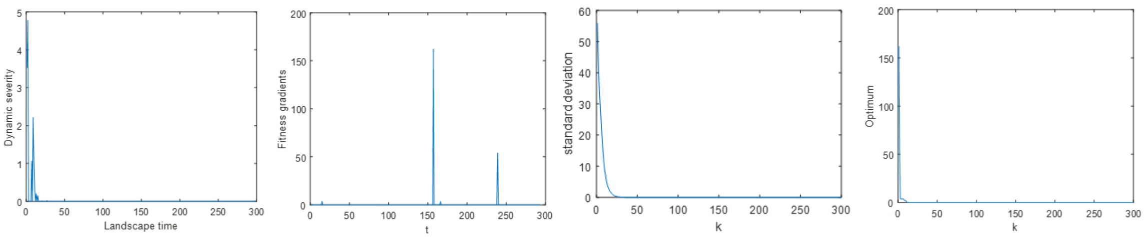


(e) Schwefel 2.26-30D

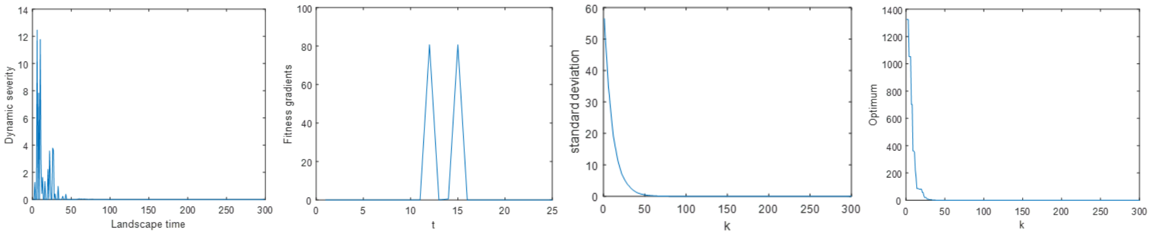
Figure 12. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Schwefel 2.26 function change dynamically based on different dimensions



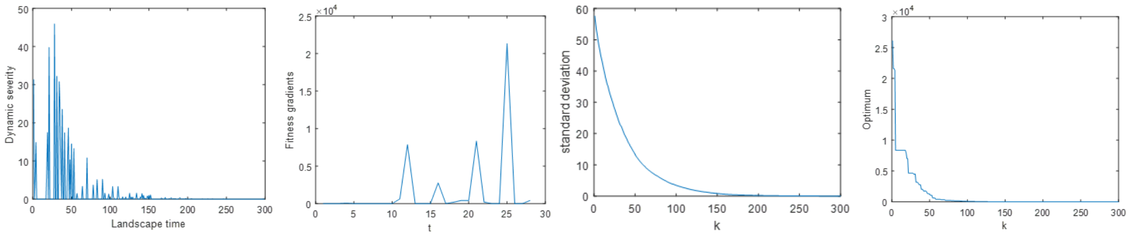
(a) Spherical-2D



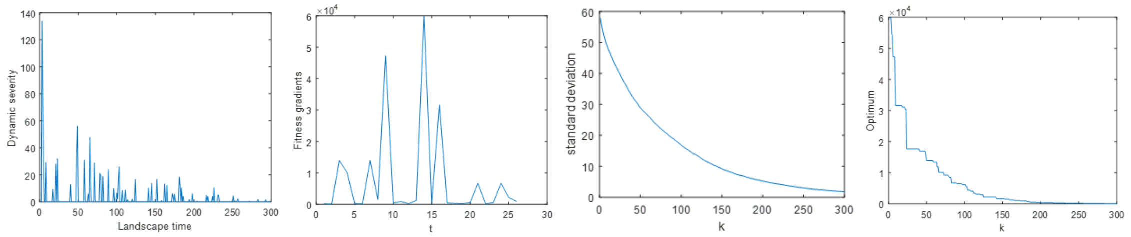
(b) Spherical-3D



(c) Spherical-5D



(d) Spherical-15D



(e) Spherical-30D

Figure 13. Dynamic severity, Fitness Gradients, Standard Gradient and Optimum of Spherical function change dynamically based on different dimensions

VI. CONCLUSION AND FUTURE DIRECTIONS

This paper mainly considers dynamic fitness landscape metrics for differential evolution algorithm: fitness distance correlation as a measure of search difficulty, a ruggedness measure based on entropy, dynamic severity as an evolutionary search measure, fitness cloud and gradients as analyzing micro behaviors of evolutionary algorithms. The basic metrics attempt to calculate and evaluate the performance of evolutionary algorithms. According to experiment results, we can see that differential evolution algorithm can not successfully handle all of benchmark functions which have different complexity. Every result that we obtained from experiment is tried to find correlations. Sometimes, it is really difficult for people to find out correlations through mass data, but it can clearly see differences through pictures. Therefore, we tried to apply fitness landscape to analyze advantages and disadvantages of differential evolution. In the future, we can apply those metrics to analyze other genetic algorithms. Besides, we can also add some new metrics or change some metrics in order to get better performance. It is very clear to see the performance of differential evolution algorithm through dynamic fitness landscape figures. And we can also use those methods in other evolutionary algorithms.

In the future, there are still a lot of work for us to do. First, in this paper, we just focus on results which are calculated by differential evolution algorithm. Actually, we can do the research deeply. For example, we can discuss three operators including mutation, selection and crossover separately. Secondly, we can apply dynamic fitness landscape metrics to analyze other evolutionary algorithms like GA (Genetic Algorithm), PSO (Particle Swarm Optimization Algorithm) and compare with each other. Finally, we can combine dynamic fitness landscape metrics with other metrics such as Markov Chain Model and analyze difference, convergence and equivalence.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China with Grant No. 61573157, the Fund of Natural Science Foundation of Guangdong Province of China with Grant No.2014A030313454 and the Key Project of Natural Statistical Science and Research with the Grant No.2015LZ30.

REFERENCES

Aldeida Aleti, I. Moser, Lars Grunske. Analysing the fitness landscape of search-based software testing problems [J]. Automated Software Engineering. 2016, pp. 1-19.

Branke, J. Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Dordrecht, 2001.

C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution, IEEE Transactions on Evolutionary Computation 8 (1) (2004) 1–13.

Hendrik Richter. Dynamic Fitness Landscape Analysis [J]. Indian Journal of Medical Sciences, 2013, 62(8): 336-339.

J Kennedy, R Eberhart. Particle Swarm Optimization [C]. In Proceedings of IEEE Internatinal Conference on Neural Networks, 1995, Vol 4, 1942-1948.

J Kennedy. The particle swarm: social adaptation of knowledge. Proceedings of IEEE International Conference on Evolutionary Computation. 1997, pp. 303–308.

Jones, T, Forrest S. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. Proceedings of 6th International Conference on Genetic Algorithms [C]. 1995, pp: 184-192.

K. A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1975.

K. M. Malan, A. P. Engelbrecht. Quantifying Ruggedness of Continuous Landscapes using Entropy [C]. in Proceedings of the IEEE Congress on Evolutionary Computation, 2009, pp. 1440-1447.

K.V.Price, R.M.Storn, J.A.Lampinen. Appendix A.1: Unconstrained Uni-modal Test Functions. In Differential Evolution A practical Approach to Global Optimization, ser. Natural Computing Series. Berlin, Germany: Springerr- Verlage, 2005, pp. 514-533.

Katherine M. Malan, Andries P. Engelbrecht. Particle Swarm Optimisation Failure Prediction Based on Fitness Landscape Characteristics [J]. IEEE Symposium on Swarm Intelligence, 2014: 1-9.

Katherine M. Malan, Andries P. Engelbrecht. Ruggedness, Funnels and Gradients in Fitness Landscapes and the Effect on PSO Performance [C]. 2013 IEEE Congress on Evolutionary Computation.

L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley, New York, 1966.

Matthieu Basseur, Adrien Goeffon. Climbing Combinatorial Fitness Landscapes [J]. Applied Soft Computing. 2015, pp. 688-704.

P Rocca, G Oliveri, A Massa. Differential Evolution as Applied to Electromagnetics [J]. Antennas & Propagation Magazine IEEE, 2011, 53(1): 38-49.

P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special on Real-Parameter Optimization. Nanyang Technological University, Singapore, Tech. Rep., May 2005.

Pietro A. Consoli, Yi Mei, Leandro L. Minku, Xin Yao. Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis [J]. Soft Computing. 2016, pp: 1-26.

R. Storn, K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global optimization, vol. 11, pp. 341-359, 1997.

Richter, H. A Study of Dynamic Severity in Chaotic Fitness Landscapes [C]. IEEE Congress on Evolutionary Computation, 2005, pp. 2824-2831.

S. K. Mishra. Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program. Social Science Research Network (SSRN), Tech. Rep. August 2006.

Sebastien Verel, Philippe Collard, Manuel Clergue. Where are Bottlenecks in NK Fitness Landscapes?. Computer Science. 2007, vol. 1, pp. 273-280.

Shayan Poursoltan, Frank Neumann. Ruggedness Quantifying for Constrained Continuous Fitness Landscapes [C]. Evolutionary Constrained Optimization. 2015, pp: 29-50.

Shinichi Shirakawa, Tomoharu Nagao. Bag of local landscape features for fitness landscape analysis [J]. Soft Computing. 2016, pp: 1-16.

Stefan Nowak, Joachim Krug. Analysis of adaptive walks on NK fitness landscapes with different interaction schemes [J]. Journal of Statistical Mechanics: Theory and Experiment. 2015, pp: 1742-5468.

T. Back, D. B. Fogel, Z. Michalewicz. Handbook of evolutionary computation. New York: Inst. Phys. and Oxford Univ. Press, 1997.

V. K. Vassilev, T. C. Fogarty, J. F. Miller. Smoothness, Ruggedness and Neutrality of Fitness Landscape: from Theory to Application [J]. in Advances in Evolutionary Computing: Theory and Applications. Springer-Verlag New York, Inc., 2003, pp. 3-44.

Vitaly Bolshakov, Erik Pitzer, Michael Affenzeller. Fitness Landscape Analysis of Simulation Optimisation Problems with HeuristicLab [C]. 2011 UKSim 5th European Symposium on Computer Modeling and Simulation, 2011.

Weicker, K. An Analysis of Dynamic severity and Population size. International Conference on Parallel Problem Solving from Nature. 2000, 1917: 159-168.

X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

X.Yao, Y.Liu, G.Lin. Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation. 1999, vol. 3, no. 2, pp. 82-102.