

# Ruggedness Quantifying for Constrained Continuous Fitness Landscapes

Shayan Poursoltan and Frank Neumann

Optimisation and Logistics, School of Computer Science  
University of Adelaide, Adelaide, SA 5005, Australia  
shayan.poursoltan@adelaide.edu.au  
frank.neumann@adelaide.edu.au

**Abstract.** Constrained optimisation problems appear frequently in important real world applications. In this chapter, we study algorithms for constrained optimisation problems from a theoretical perspective. Our goal is to understand how the fitness landscape influences the success of certain types of algorithms. One important feature for analysing and classifying fitness landscape is its ruggedness. It is generally assumed that rugged landscapes make the optimisation process by bio-inspired computing methods much harder than smoothed landscapes which give clear hints towards an optimal solution. We will introduce different methods for quantifying the ruggedness of a given constrained optimisation problem. They, in particular, take into account on how to deal with infeasible regions in the underlying search space.

**Keywords:** constrained optimisation, continuous optimisation, fitness landscapes, ruggedness

## 1 Introduction

Constrained optimisation problems (COP)s, especially non-linear ones, are very important and widespread in many real world applications such as chemical engineering, VLSI chip design and structural design [3]. Various algorithmic approaches have been introduced to tackle constrained optimisation problems. The major component of these optimisation algorithms is devoted to the handling of the involved constraints.

Different types of evolutionary algorithms such as evolutionary strategies [16], differential evolution [19], and particle swarm optimisation (PSO) [2] have been applied to constrained continuous optimisation problems. Constraint handling mechanisms that are frequently used include penalty functions, decoder based methods, and special operators that separate the treatment of the objective function and the constraints. We refer the reader to [13] for an overview on the different types of methods. Among various types of optimisation algorithms, penalty methods are well-known as one of the most successful and popular approaches for dealing with constraints. They penalise the violation of constraints by adding penalty values to the fitness value of a given solution. Effectively, this transforms the constrained problem into an unconstrained one. Turning constrained optimisation problems into unconstrained ones by using penalty functions makes the

problem easily accessible to a wide range of methods for unconstrained optimisation and can be regarded as one of the major reasons for the popularity of penalty functions.

There is a wide range of optimisation algorithms for constrained continuous optimisation problems and their performances are usually evaluated based on the results of popular benchmark problems [9, 6]. These benchmark problems are designed to impose different types of difficulties for optimisation algorithms. As evolutionary algorithms make heavy use of random decisions, it is hard to understand the behaviour of these algorithms from an analytic perspective. More importantly, it is very hard to predict which algorithm would perform best for a newly given real-world optimisation problem. Mersmann et al [12] have proposed the following steps to select the best possible algorithm from a given suite of algorithms. First, one has to extract important problem properties from the class of problems that is under investigation. Secondly, it is necessary to analyse the performance of different algorithms based on the problem properties and build a prediction model which allows to select the best possible algorithm based on problem characteristics.

There are various problem properties associated with the fitness landscape. In other words, analysing the fitness landscape helps us to classify them with related characteristics that make problems easy or hard to solve by certain types of algorithms. In recent years, fitness landscape analysis has become very popular to describe the characteristics of optimisation problems. Important attributes that are associated with fitness landscapes and that impact the optimisation process of evolutionary algorithms include the smoothness, multi-modality, feasibility rate and variable separability of the landscape and the considered problem [14].

Among several characteristics associated with fitness landscapes, the notion of *fitness landscape ruggedness* plays a vital role in determining the problem difficulty. If the objective function is unsteady and goes up and down frequently, choosing the right direction to continue becomes difficult for many solvers. Since ruggedness and problem difficulty are closely related to each other, many studies have been conducted to analyse this feature. For discrete landscapes, one important approach is to consider autocorrelations by calculating the correlation of fitness values of search points that are visited by a random walk on the landscape [22]. Furthermore, there have been many studies that extend the basic autocorrelation approach to provide additional insights on fitness landscapes [1, 5]. One of the drawbacks of using autocorrelation by these statistical analysis techniques is that the calculated value is a vague notion that does not clearly reflect the landscape ruggedness. Thus, Vassilev proposed a new technique which is based on the assumption that each landscape is an ensemble of different objects (the nodes seen by a random walk on the fitness landscape), which can be grouped by their form, size and distribution [20]. Vassilev's approach was applicable to discrete problems. For real parameter landscapes, Malan [8] used Vassilev's information theoretic analysis to measure the fitness landscape ruggedness in the continuous domain. So far, these landscape analysis techniques have been conducted only for unconstrained or discrete problems. Measuring the landscape ruggedness for constrained continuous problems imposes additional challenges and we will propose how to tackle them in this chapter.

We propose an approach to measure the fitness landscape ruggedness of constrained continuous optimisation problems. The quantification of ruggedness combined with other analytical problem characteristics can help to build an algorithm

selection model based on the relation of different algorithms and problem properties. This chapter includes a methodology for quantifying fitness landscape ruggedness of constrained continuous problems. In order to do this, we extend Malan’s approach to quantify the fitness landscape ruggedness of constrained continuous problems. The information obtained by using simple random walks on constrained problems landscape is not useful enough since it is mostly related to infeasible areas which are unlikely to be seen by the solver. To cope with constraints in nearly infeasible problems, our approach replaces Malan’s random walk with a biased one. The obtained samples are used to quantify the ruggedness of landscapes by using the approach of Vassilev [20]. We evaluate our approach on well-known benchmarks taken from the recent CEC competitions [9] and discuss the benefits and drawbacks of our new approach.

The remainder of this chapter is organized as follows: In Section 2, we introduce constrained continuous optimisation and discuss approaches that have been used to analyse the ruggedness of unconstrained fitness landscapes. We present our approach for quantifying ruggedness of constrained continuous fitness landscapes in Section 3 and the results of our experimental investigations in Section 4. Finally, we finish our research with some concluding remarks.

## 2 Preliminaries

In this section, we introduce basic notations and summarize previous works on measuring the ruggedness of fitness landscapes.

### 2.1 Constrained continuous optimisation problem

Constrained continuous optimisation problems are optimisation problems where a function on real-valued variables should be optimized with respect to a given set of constraints. Constraints are usually given by a set of inequalities and/or equalities. Without loss of generality, we present our approach for minimization problems. Formally, we consider single-objective functions  $f: S \rightarrow \mathbb{R}$ , with  $S \subseteq \mathbb{R}^n$ . The constraints impose a feasible subset  $F \subseteq S$  of the search space  $S$  and the goal is to find an element  $x \in S \cap F$  that minimizes  $f$ .

We consider problems of the following form:

$$\text{Minimize } f(x), \quad x = (x_1, \dots, x_n) \in \mathbb{R}^n \quad (1)$$

such that  $x \in S \cap F$ .

The feasible region  $F \subseteq S$  of the search space  $S$  is defined by

$$l_i \leq x_i \leq u_i, \quad 1 \leq i \leq n \quad (2)$$

where  $l_i$  and  $u_i$  are lower and upper bounds on the variable  $x_i$ ,  $1 \leq i \leq n$ . Additional constraints are given by the functions

$$\begin{aligned} g_i(x) &\leq 0, \quad 1 \leq i \leq q, \\ h_i(x) &= 0, \quad q+1 \leq i \leq p \end{aligned}$$

In order to work with iterative optimisation algorithms for these problems, it is common to relax the equality constraints

$$h_i(x) = 0, \quad q + 1 \leq i \leq p$$

to

$$|h_i(x)| \leq \varepsilon, \quad q + 1 \leq i \leq p \quad (3)$$

where  $\varepsilon$  is a very small positive value that determines how much the original constraints can be violated. In our experimental study, we will work with  $\varepsilon = 0.0001$  which is the same setting as used in [9].

## 2.2 Fitness landscape ruggedness analysis using the entropy measure

A fitness landscape (see [18]) is given by a search space  $S$ , a fitness function  $f : S \rightarrow \mathbb{R}$  which assigns a value  $f(s)$  to each search point  $s \in S$ , and a function  $v : S \rightarrow 2^S$  that assigns to each search point  $s$ , a set  $v(s) \subseteq S$  of search points. The elements in  $v(s)$  are called the neighbours of  $s$ .

Various techniques have been used for the statistical analysis of fitness landscapes. Popular techniques measure the correlation of the search points visited by a random walk algorithm [22, 7, 10]. However, it has been shown that this information is very basic and not very useful to reflect problem difficulty [11]. Vassilev [20] conducted an information theoretic approach to quantify fitness landscape ruggedness. The difference between Vassilev's and the previous approaches is that his technique focuses on the relation between ruggedness and neutrality of the problem landscape. Vassilev's method performs a random walk on a fitness landscape to generate a sequence of fitness values  $\{f_i\}_{i=0}^n$ . This random walk starts from a random position on a discrete landscape and moves to its neighbor using bit flips. The aim of this method is to extract an ensemble of objects from a sequence of fitness values. These objects can be classified into 3 categories:

- Flat objects: The fitness value of each point is similar to its two visited neighbours (predecessor and successor).
- Isolated objects: Each point has higher or lower fitness value comparing to its two neighbours.
- Points which do not belong to the former two groups.

The aim of the approach is to extract the ensemble of objects mentioned above from the values in a sequence of fitness values. The following function represents the time series as a set of objects. The ensemble is defined as a string  $S(\varepsilon) = (s_1 s_2 s_3 \dots s_n)$  with  $s_i \in \{\bar{1}, 0, 1\}$  given by

$$s_i = \Psi_{f_i}(i, \varepsilon) = \begin{cases} \bar{1}, & \text{if } f_i - f_{i-1} < -\varepsilon \\ 0, & \text{if } |f_i - f_{i-1}| \leq \varepsilon \\ 1, & \text{if } f_i - f_{i-1} > \varepsilon \end{cases} \quad (4)$$

where the parameter  $\varepsilon$  is the real positive number that represents the accuracy of the calculation of the string  $S(\varepsilon)$ . According to the function, if  $\varepsilon = 0$  then the function will be very sensitive to the differences of points. It can be observed that increasing the value of  $\varepsilon$  reduces the sensitivity of the function. Therefore, if the value of  $\varepsilon$  equals to the difference of highest and lowest points in the walk, then the fitness sequence will only consist of zeros.

Table 1: Various sub blocks in  $S_i$  considered as rugged objects

Sub block	$\bar{1}0$	$\bar{1}\bar{1}$	$01$	$10$	$1\bar{1}$	$0\bar{1}$
Object Type	Rugged	Rugged	Rugged	Rugged	Rugged	Rugged
Object Figure						

Table 2: Various sub blocks in  $S_i$  considered as flat objects

Sub block	$00$	$\bar{1}\bar{1}$	$11$
Object Type	Flat	Flat	Flat
Object Figure			

To measure the ruggedness, the entropy of the string  $S(\epsilon)$  is calculated as follows:

$$H(S(\epsilon)) = - \sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]} \quad (5)$$

where  $pq$  is a substring of the string  $S(\epsilon)$  consisting of two elements. Furthermore,  $H(S(\epsilon))$  is the information content which is an estimation of variety in different shapes within the string of  $S(\epsilon)$ . This measurement is used to characterise the landscape ruggedness with respect to the flat areas where neutrality is present.  $P_{[pq]}$  refers to the frequency of the blocks where  $p$  and  $q$  have different values ( $p \neq q$ ):

$$P_{[pq]} = \frac{n_{[pq]}}{n} \quad (6)$$

In other words, in order to measure ruggedness with respect to neutrality, it is necessary to include the rugged block in our estimation ( $p \neq q$ ). Thus, sub blocks with two similar elements are excluded (case  $p = q$ ) in this function. The formula calculates the frequencies of sub block with different symbols. As it is discussed above, since there are six different possibilities of rugged sub blocks in the string (according to Table 1), the logarithm base is set to 6. The different possibilities of rugged objects are considered as isolated area which each point has different values. Table 1 and 2 show different possibilities of rugged and flat sub blocks of  $pq$  in the string of  $S(\epsilon)$ .

As discussed earlier, the variable  $\epsilon$  controls the sensitivity of the function  $\Psi$  (see Equation 4). It can be observed that greater values for  $\epsilon$  lead to more neutrality in the measurement. It is suggested that using smaller values of  $\epsilon$  makes the behaviour of  $H(S(\epsilon))$  significant for characterising the ruggedness with respect to the landscape neutrality [21]. Therefore, for comparing various problems with different fitness ranges, the smaller values of  $\epsilon$  are used for  $H(S(\epsilon))$ . The values of  $\epsilon$  used in [8] are:

$$\epsilon = 2^{-k} \epsilon^* \quad (k = 1, 2, \dots, 8). \quad (7)$$

in which,  $\epsilon^*$  is the smallest value that generates all sub blocks as zeros and consequently the landscape becomes flat. Also,  $k$  is considered 1 to 8 to calculate smaller values for  $\epsilon$ s. Note that the parameter  $\epsilon^*$  can be calculated as the difference of highest and lowest fitness that has been found in the random walk.

1. Choose a random place within the bound as a starting the point
2. Generate all the neighbours of the chosen point using permutation
3. Choose one neighbour randomly and save its value
4. Go back to step 2

**Algorithm 1: Random Walk**

1. Input: Problem domain (*domain*), number of the dimensions (*dimension*) and number of steps (*MaxStepNumber*) for the walk
2. Calculate the maximum step size
 
$$\text{MaxStepSize} = \frac{\text{Range of the problem domain}}{100}$$
3. Set *counter* = 0 and create an array *steps* to save the steps in the walk
4. Assign a random position to *steps[0]* within the boundaries of the problem
5. **Repeat**
6.   **For** every dimension *i* of the problem
7.     *currentStep* = random(0,MaxStepSize);
8.     *steps(counter)* = *steps(counter-1)*+*currentStep*;
- 9.
10.   **If** *steps(counter)* > boundaries
11.     *steps(counter)* = *steps(counter-1)*-(Range of the problem domain);
12.   **Endif**
13. **Endfor**
14. **Until** (*counter* < *MaxStepNumber*)

**Algorithm 2: Random increasing walk algorithm.**

An entropic measure  $H(S(\epsilon))$  requires a sequence of search points  $S(\epsilon)$ . In order to generate a set of time series, a simple random walk on a landscape path can be used (see Algorithm 1).

The above method was used for measuring the ruggedness of discrete problems. The major issue of using this approach for continuous problems is that (unlike the discrete problems) it is not possible to generate or access all possible neighbours of visited individual. Thus, Malan and Engelbrecht [8] modified the approach to use it for unconstrained continuous problems. The proposed approach adopts a random increasing walk which increases the step size over time. Furthermore, the step size is decreased if the algorithm produces a solution that is not within the boundaries given by the constraints. The algorithm for the random increasing walk proposed in [8] is given in Algorithm 2. Here, we assume that the variable range is the same for all dimensions which implies that the maximum step size is the same for all dimensions. The algorithm can be easily adjusted to problems with different variable ranges by using a maximum step size for each variable.

### 3 Ruggedness quantification for constrained continuous optimisation

In this section, we present a new approach for quantifying the ruggedness of a fitness landscape of a constrained continuous optimisation problem. Since we are

working on constrained optimisation problems, dealing with infeasible areas is the important and challenging part. Often in these problems, the infeasibility rate is high and it might be even very hard to find one feasible solution. This implies that random walk methods are usually not that helpful as they would produce infeasible solutions most of the time. Most constraint handling methods direct the search process to feasible regions of the search space and therefore often allow to optimize in the feasible region of the search space which might be a very small proportion of the size of the overall space.

### **3.1 Ruggedness Quantification**

In the following, we discuss drawbacks of applying previous approaches for ruggedness quantification when dealing with constrained continuous optimisation problems. Afterwards, we explain the solution to these issues following by our new approach. As mentioned in the previous section, random walk algorithms have been used to measure the ruggedness of fitness landscapes. However, random walk algorithms are often not useful when it comes to constrained optimisation problems. We discuss the different reasons in the following.

A random walk algorithm is not accurate enough to reflect the fitness landscape as a whole which is already true for unconstrained optimisation but becomes even more evident when dealing with constrained problems. Random walk algorithms cannot discriminate accurately between two different search spaces (feasible and infeasible space) since they do not make decision based on the fitness values. Experiments show that the statistics obtained by random walks on landscapes are biased to areas with low fitness [17]. Hence, various landscapes with different high fitness value areas and same low areas generate similar data for walks and consequently the obtained ruggedness measures are within the same range when using previous methodologies. To address this issue, we introduce methods which take into account the individual fitness values in the sampling process. Using this method forces the algorithm to explore higher fitness values in landscape which is more interesting for optimisation algorithms. Therefore, the calculated fitness landscape ruggedness is more interesting as it reflects the landscape structure in regions of the search space that are crucial for optimisation.

The chance of finding even a few feasible individuals when using random walk algorithms is likely to be very low for highly infeasible landscapes. Since the majority of constrained optimisation problems are nearly infeasible, it is more likely to have more infeasible individuals when using a random walk to explore the landscape. Optimisation algorithms prefer to move and search in feasible regions. In order to solve this problem, the sampling method for exploring fitness landscapes of constrained optimisation problems needs to move towards feasible areas in the search space. Our remedy for this issue is that we introduce methods that have the ability to distinguish between feasible and infeasible individuals when choosing the next step in the walk. Our method is flexible and can be tuned such that the walk contains more or less feasible individual in it.

### **3.2 Biased sampling using evolution strategies**

We use a biased walk in our approach to quantify the ruggedness of a constrained problem fitness landscape. Considering the fitness values of individuals in the

1. Initialize the strategy parameters, set generationCounter = 0
2. Initialize and create the population of solution of  $x$  using uniform  $n$  dimensional probability distribution on problem search space ( $\mu$  individuals)
3. Evaluate the fitness of population
4. **Repeat**
5.   Generate offspring using Equation 3.2 and 3.2 (**mutation**)
6.   Evaluate the fitness of offspring
7.   Apply the selection process to select from offspring individuals for next generation (**selection**).
8.   generationCounter = generationCounter + 1
9. **Until** stopping condition is true

**Algorithm 3:**  $(\mu, \lambda)$ -ES used as biased walking

sampling process improves the reliability of the calculated measure. Our biased walk is using a simple evolution strategy [16]. Since the adjacent steps in the walk should be different, we use a  $(\mu, \lambda)$ -ES. This means that the selection is performed among the  $\lambda$  offspring and their parents are excluded from new generation.

In the  $(\mu, \lambda)$ -ES, each individual (both parents and offspring) is a vector  $(x_i, \sigma_i)$  consisting of the coordinates of the search point and the step sizes for the different coordinates. The initial population is generated by choosing  $\mu$  solutions uniformly at random from the search space and the initial step size of variable  $j$  in individual  $i$  is given as:

$$\sigma_{i,j}^{(0)} = \frac{\Delta x_{i,j}}{\sqrt{n}}$$

in which  $\sigma_{i,j}$  refers to the  $j$ th component of vector  $\sigma_i$  and  $\Delta x_{i,j}$  is the difference of upper and lower bounds on  $\sigma_{i,j}$  [16, p. 117]. It is noteworthy that the calculated strategy parameters for each generation are used in the next generation. The step sizes for each generation are as follows:

$$\sigma'_{ij}(t+1) = \sigma_{ij}(t) e^{\tau' N(0,1) + \tau N_j(0,1)}$$

where  $\tau' = \frac{1}{\sqrt{2n}}$  and  $\tau = \frac{1}{\sqrt{2\sqrt{n}}}$  are learning rates and  $N(0, 1)$  is normally distributed random variable and  $N_j(0, 1)$  denotes that there is a new value for each component of  $\sigma$ .

By calculating the next generation strategy parameters (as above), each parent produces new individuals as:

$$x'_{h,j} = x_{i,j}(t) + N_j(0, 1) \sigma'_{h,j}(t)$$

where  $h \in \{1, \dots, \lambda\}$  and  $i \in \{1, \dots, \mu\}$ . The pseudo-code for  $(\mu, \lambda)$ -ES is shown in Algorithm 3. In this chapter, we are using  $\mu = 1$ , i.e. a  $(1, \lambda)$ -ES. This implies that each search point in the sequence we are generating is an offspring of the previous point in this sequence.

1. Initialize probability of  $P_f$
2.  $I_j = \{1, \dots, \lambda\}$
3. **For**  $i$  starts at 1,  $i < N$ , increment  $i$
4.   **For**  $j$  starts at 1,  $h < \lambda - 1$ , increment  $j$
5.     Generate a random number ( $U$ ) in the range of (0,1)
6.     **If** ( $\phi(I_j) = \phi(I_{j+1}) = 0$ ) **or** ( $U < P_f$ )
7.       **If**  $f(I_j) > f(I_{j+1})$
8.          $swap(I_j, I_{j+1})$
9.       **End if**
10.     **else**
11.       **If**  $\phi(I_j) > \phi(I_{j+1})$
12.          $swap(I_j, I_{j+1})$
13.       **End if**
14.     **End if**
15.   **End for**
16. **Break** if no changes occurred within a complete sweep
17. **End for**

**Algorithm 4:** Stochastic ranking for dealing with infeasible areas.  $N$  is the number of sweeps needed for whole population,  $\lambda$  is the number of individuals which are ranked by at least  $\lambda$  sweeps and  $\phi$  is a real-valued function that imposes penalty.

### 3.3 Dealing with infeasible areas

Among all categories of constraint handling methods, it has been shown that penalty methods in general have a good performance [9]. Some methods calculate the constraint violation as a sum of violation of all constraints and integrate them into the objective function.

When integrating constraint violations into the objective function, the main problem is to choose an appropriate penalty coefficient that determines how strongly the constraint violation influences the objective value. There are also penalty methods which use the constraint violation and objective functions separately. In this case, the optimise the constraint violation and objective function in lexicographic order so that the main goal is to obtain a feasible solution.

As discussed earlier, to deal with nearly infeasible problems, there is a need to use a walk with the ability to distinguish between feasible and infeasible individuals. We choose the stochastic ranking method proposed by Runarsson [15] as our constraint handling mechanism to sample and collect individuals for the time series  $S(\epsilon)$ . It has been observed that there should be a balance between accepting infeasible individuals and preserving feasible ones. Hence, neither over nor under penalising infeasible solution is a proper choice as constraint handling method [4]. It is worth noting that all penalty methods try to adjust the balance between the objective and penalty function. The proposed stochastic ranking method adjusts this balance in a direct way. By using this method, the walk is directed towards feasible areas of the search space.

The stochastic ranking method is used to rank  $\lambda$  offspring in evolutionary strategy we discussed earlier (see Algorithm 4). The ranking is achieved by comparing adjacent individuals in at least  $\lambda$  sweeps. The ranking is terminated once no change occurs during a whole sweep. To determine the balance of offspring selection, the

1. Parameter setting:  $P_f=0.4$ , MaxStepNumber=5000
2. Set  $counter=0$  and create an array of  $steps$  to save the steps in the walk
3. **Repeat**
4. Produce new individuals by using evolutionary strategy (ES) in algorithm es
5. Rank generated offspring by employing stochastic ranking method in Algorithm 4
6. Save the highest ranking individual fitness (infeasible/feasible) in array of  $steps[counter]$
7.  $counter = counter + 1$
8. **Until** ( $counter < \text{MaxStepNumber}$ )
9. Set  $\epsilon^* = \max(steps[]) - \min(steps[])$
10. Generate ensemble of objects (Equation 4)
11. Calculate the entropic measure  $H(S(\epsilon))$  (Equation 5)

**Algorithm 5:** Ruggedness quantifying for constrained continuous fitness landscape problem

probability of  $P_f$  has been introduced in [15]. In other words,  $P_f$  is the probability of comparing two adjacent individuals based on their objective function. It is obvious, if two comparing individuals are feasible then  $P_f$  is 1.

### 3.4 Ruggedness quantifying method using constrained handling biased walk

We already explained how we use a biased walk which can distinguish the feasible and infeasible individuals. In order to obtain more interesting individuals, we need to use a biased walk that moves through good regions of the fitness landscape. It is necessary to have feasible solution within the walk steps in order to obtain an effective ruggedness measure. Therefore, our approach uses a biased walk by constraint handling methods which make it possible to have feasible individuals in the path. In the algorithm, the individuals that are found by the simple evolutionary strategy are ranked by the stochastic ranking method. Afterwards, the highest rank individual is selected for the step walk. The pseudo-code of our methodology to quantify the ruggedness of constrained continuous fitness landscapes is given in Algorithm 5.

As mentioned in the previous section,  $P_f$  controls the probability of comparing two adjacent individual  $x$  and  $y$  based on their objective function. According to [15], the probability of winning for  $x$  is given by

$$P_w = P_{f\omega}P_f + P_{\phi\omega}(1 - P_f) \quad (8)$$

where  $P_{f\omega}$  is the probability that individual  $x$  is winning when  $x$  and  $y$  are compared according to their objective function value and  $P_{\phi\omega}$  is the probability that  $x$  wins when they are compared according to the penalty function.

As discussed in Section 3.1, the walking algorithm should consider both feasible and infeasible areas. Thus,  $P_f$  determines whether the comparison is based on the objective or the penalty function. Of course, the impact of this parameter setting depends on the fitness landscape under investigation. By adjusting the parameter  $P_f$ , we can control the number of feasible or infeasible individuals in the walk and consequently the calculated ruggedness measure is more likely based on the feasible or infeasible regions.

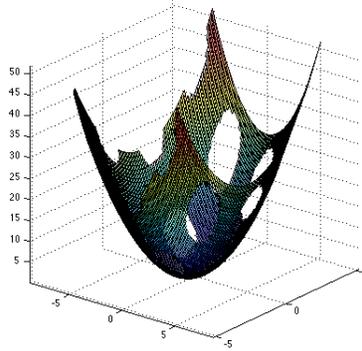


Fig. 1: Two dimensional constrained sphere function using the functions

## 4 Experimental studies

In this part, we describe experimental studies to evaluate our approach for measuring the ruggedness of a constrained continuous fitness landscape. We carry out experimental investigations on two different types of problems. The first consists of a constrained version of the classical Sphere function. Imposing constraints that lead to different infeasible areas, we examine our approach with respect to the number of feasible solutions that are obtained during the run of the algorithm and compare it to the other approaches outlined in Section 2.2. Then, we examine our approach on different benchmark functions taken from the special session on single objective constrained real parameter optimisation [9] at CEC 2010.

### 4.1 Constrained sphere function

In order to investigate the proposed method, we first consider the following constrained version of the 2 dimension classical Sphere function:

$$\min Sphere(x) = \sum_{i=1}^n x_i^2 \quad -5.12 \leq x_i \leq 5.12$$

subject to  $g(x) \leq 0$

where  $g(x)$  imposes the constraints of the 2 dimensional sphere function. We construct 3 different problems that differ from each other by using each of the following constraints:

- $g_1(x) = 10(\sum_{i=1}^n |\cos^3(x_i - 40)|) - 4$ ,
- $g_2(x) = 10(\sum_{i=1}^n |\cos^3(x_i - 40)|) - 8$ ,
- $g_3(x) = 10(\sum_{i=1}^n |\cos^3(x_i - 40)|) - 12$

In this experiment different optimisation problems ( $Sphere_{g_1}$ ,  $Sphere_{g_2}$ ,  $Sphere_{g_3}$ ) have low, medium and high feasibility rate. In this experiment, we consider 2 dimension sphere function to analyse the results more accurately. Figures 1,2 show the feasible areas in these 3 functions ( $n = 2$ ).

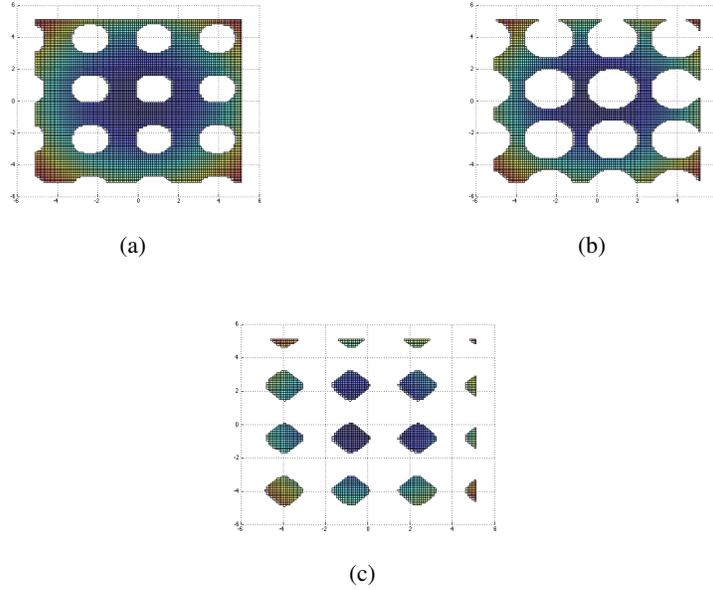


Fig. 2: Two dimensional space of the constrained sphere functions with infeasible areas marked white: (a)  $\text{Sphere}_{g_1}$ , (b)  $\text{Sphere}_{g_2}$ , (c)  $\text{Sphere}_{g_3}$  having low, medium and high infeasibility rate

Table 3: Percentage of feasible individuals in the walks

	$\text{Sphere}_{g_1}$	$\text{Sphere}_{g_2}$	$\text{Sphere}_{g_3}$
Random increasing walk	71.3	55.8	28.7
Biased walk	75.8	68.1	48.7

We apply and compare the random increasing walk (see Algorithm 2) with our methodology on these problems with different feasibility rate. In this experiment, we use (1,7)-ES algorithm and  $P_f = 0.4$  that means the ES has a tendency to focus on feasible solutions. We did 20 independent runs consisting of 1000 steps each and for each problem the percentage of feasible solutions is represented in Table 3.

Due to stochastic nature of an evolutionary optimisation, the above test is repeated 10 times and the two-tail  $t$ -test significance is performed. In all tests, the significant level  $\alpha$  is assigned as 0.05. The  $p$ -values for each function are represented in Table 4. The results shows that the difference of means are significant and less than 0.05.

Clearly, our methodology is less influenced by increasing the infeasibility rate of the problem. Also, comparing both walks shows that using our biased walk is more likely to obtain feasible individuals (steps) in the walk see Table 3). The

Table 4:  $p$ -values for significance of a difference between two means for running Random increasing and Biased walk over 3 functions.

	Sphere <sub><math>g_1</math></sub>	Sphere <sub><math>g_2</math></sub>	Sphere <sub><math>g_3</math></sub>
$p$ -value	0.0043	$7.0834E - 06$	$9.4817E - 06$

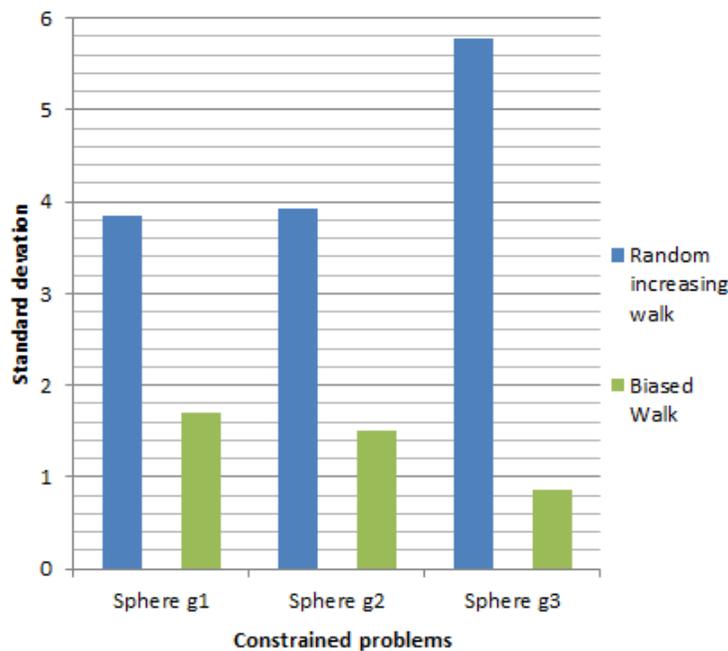


Fig. 3: Standard deviation for average percentage of feasible individuals in walks using Random increasing and Biased walks

standard deviation of feasible individuals in both walks are shown in Figure 3. It is clear that the standard deviation of feasible individuals is higher for random walks.

Thus, the obtained ruggedness measure is related to the feasible parts which is more likely to be seen by the solver.

#### 4.2 CEC benchmark problems

Also, we investigate our new method on benchmark problems from CEC 2010 competition [9]. First, we compare our method with random increasing walk in terms of number of feasible individuals (steps) in the walk. In order to do this, we use (1,7)-ES in this experiment and  $P_f$  is considered as 0.4 which forces the walk towards feasible areas (see Equation 8). We calculate the number of feasible steps (individuals) taken by the walking algorithm within 5000 steps for nearly infeasible problems. Figure 4 shows the results of 30 independent runs on CEC

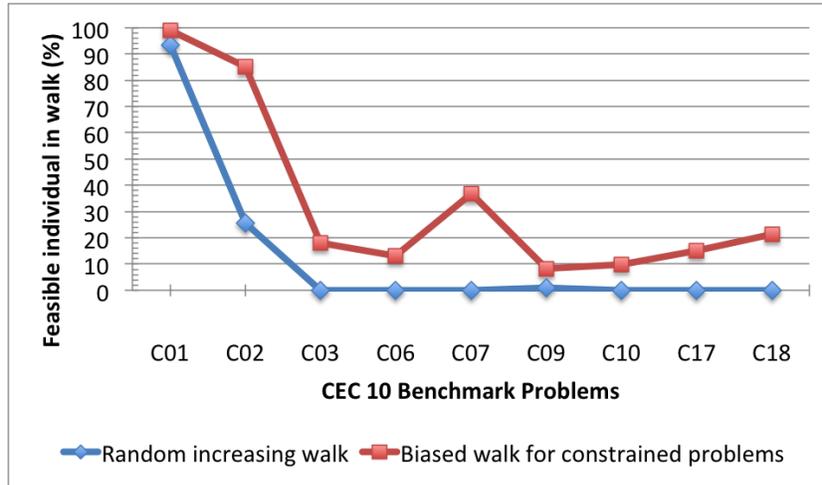


Fig. 4: Percentage of feasible individuals in walks for nearly infeasible CEC benchmark problems

problems. It can be observed that for nearly infeasible problems [9], our method performs better to include more feasible individual in the steps (see Figure 4).

Also, to test the ability of our methodology in ruggedness quantifying, we used different CEC benchmark problems with  $D = 10$ . To quantify the ruggedness, we calculate the entropic measure  $H(S(\epsilon))$  for different values of  $\epsilon$  (Equation 7). Table 5 shows our experimental results. The results indicate the mean value of  $H(S(\epsilon))$ s for different values of  $\epsilon$ s over 30 runs. Based on [8], the ruggedness feature of problem is considered as the maximum value of  $H(S(\epsilon))$  among all different  $\epsilon$ s. These numbers are values describing the ruggedness of each problem fitness landscape with respect to neutrality. Also, the standard deviation for different  $\epsilon$ 's is shown in Table 6.

To interpret this table, it is convenient to classify the problems based on their objective functions. Problems C17 and C18 are similar according to their objective functions and present close values for their ruggedness. For problems C03, C07, C09 and C10 (with the same objective function), the ruggedness measure is in the same range. C02 and C06 with the same objective function have different ruggedness measures compared to C01 which has the largest value in ruggedness. Therefore, it can be concluded that it is more likely that similar problems have similar ruggedness measures. Based on the table, we can conclude that C01 is more rugged than other categories.

## 5 Conclusions

In this chapter, we have reviewed the literature on measuring ruggedness of fitness landscapes and discussed the drawbacks of the current methods when dealing with constrained problems. In order to address constrained continuous optimisation problems, we have presented a new technique to quantify the ruggedness of

Table 5: Ruggedness results for functions in CEC 2010 benchmarks (10D). The values for different  $\epsilon$ 's are mean values in 30 independent runs.

Function (10D)	$\epsilon^*$	$\frac{\epsilon^*}{2}$	$\frac{\epsilon^*}{4}$	$\frac{\epsilon^*}{8}$	$\frac{\epsilon^*}{16}$	$\frac{\epsilon^*}{32}$	$\frac{\epsilon^*}{64}$	$\frac{\epsilon^*}{128}$	$\frac{\epsilon^*}{256}$	Ruggedness
C01	0	0.001	0.005	0.013	0.024	0.035	0.060	0.102	0.153	<b>0.153</b>
C02	0	0.001	0.003	0.004	0.006	0.010	0.015	0.023	0.035	<b>0.035</b>
C03	0	0.000	0.001	0.004	0.009	0.011	0.014	0.014	0.013	<b>0.014</b>
C06	0	0.006	0.010	0.012	0.014	0.018	0.023	0.035	0.027	<b>0.027</b>
C07	0	0.001	0.004	0.006	0.007	0.009	0.012	0.013	0.015	<b>0.015</b>
C09	0	0.001	0.002	0.003	0.005	0.006	0.009	0.012	0.014	<b>0.014</b>
C10	0	0.002	0.002	0.003	0.004	0.006	0.007	0.01	0.012	<b>0.012</b>
C17	0	0.002	0.003	0.005	0.008	0.013	0.015	0.011	0.019	<b>0.019</b>
C18	0	0.001	0.002	0.003	0.004	0.007	0.009	0.012	0.017	<b>0.017</b>

Table 6: Standard deviation values for different  $\epsilon$ 's in 30 independent runs.

Function (10D)	$\epsilon^*$ STD	$\frac{\epsilon^*}{2}$ STD	$\frac{\epsilon^*}{4}$ STD	$\frac{\epsilon^*}{8}$ STD	$\frac{\epsilon^*}{16}$ STD	$\frac{\epsilon^*}{32}$ STD	$\frac{\epsilon^*}{64}$ STD	$\frac{\epsilon^*}{128}$ STD	$\frac{\epsilon^*}{256}$ STD
C01	0	0.002	0.005	0.006	0.009	0.0160	0.028	0.044	0.058
C02	0	0.002	0.003	0.003	0.005	0.008	0.0140	0.022	0.035
C03	0	0.000	0.000	0.000	0.001	0.002	0.003	0.004	0.009
C06	0	0.013	0.016	0.016	0.017	0.019	0.024	0.035	0.028
C07	0	0.001	0.002	0.003	0.004	0.006	0.007	0.009	0.009
C09	0	0.001	0.001	0.002	0.002	0.004	0.006	0.010	0.011
C10	0	0.001	0.001	0.002	0.003	0.004	0.005	0.007	0.009
C17	0	0.002	0.002	0.005	0.011	0.022	0.041	0.008	0.009
C18	0	0.001	0.001	0.002	0.002	0.004	0.004	0.006	0.010

constrained continuous problem landscapes. The modification is based on replacing the random sampling data by a biased walk using a  $(1, \lambda)$ -evolution strategy which can distinguish the feasible and infeasible individuals. We evaluated our approach on different benchmark functions and have shown that it produces more feasible solutions during its run. Furthermore, we evaluated our method on CEC 2010 benchmark problems and discussed the results.

## References

1. Box, G.E., Jenkins, G.M., Reinsel, G.C.: Time series analysis: forecasting and control. Wiley. com (2013)
2. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. pp. 39–43. IEEE (1995)
3. Floudas, C.A., Pardalos, P.M.: A collection of test problems for constrained global optimization algorithms, vol. 455. Springer (1990)
4. Gen, M., Cheng, R.: Genetic algorithms and engineering optimization, vol. 7. John Wiley & Sons (2000)

5. Hordijk, W.: A measure of landscapes. *Evolutionary Computation* 4(4), 335–360 (1996)
6. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.C., Deb, K.: Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics* 41 (2006)
7. Lipsitch, M.: Adaptation on rugged landscapes generated by local interactions of neighboring genes. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA (1991)
8. Malan, K.M., Engelbrecht, A.P.: Quantifying ruggedness of continuous landscapes using entropy. In: *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. pp. 1440–1447. IEEE (2009)
9. Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Nanyang Technological University, Singapore (2010)
10. Manderick, B., de Weger, M., Spiessens, P.: The genetic algorithm and the structure of the fitness landscape. In: *Proceedings of the fourth international conference on genetic algorithms*. pp. 143–150. San Mateo, CA: Morgan Kaufman (1991)
11. Mattfeld, D.C., Bierwirth, C., Kopfer, H.: A search space analysis of the job shop scheduling problem. *Annals of Operations Research* 86, 441–453 (1999)
12. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. pp. 829–836. ACM (2011)
13. Mezura-Montes, E., Coello Coello, C.A.: Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation* 1(4), 173–194 (2011)
14. Naudts, B., Kallel, L.: A comparison of predictive measures of problem difficulty in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on* 4(1), 1–15 (2000)
15. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on* 4(3), 284–294 (2000)
16. Schwefel, H.P.P.: *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc. (1993)
17. Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. *Evolutionary computation* 10(1), 1–34 (2002)
18. Stadler, P.F., et al.: Towards a theory of landscapes. In: *Complex systems and binary networks*, pp. 78–163. Springer (1995)
19. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11(4), 341–359 (1997)
20. Vassilev, V.K., Fogarty, T.C., Miller, J.F.: Information characteristics and the structure of landscapes. *Evolutionary Computation* 8(1), 31–60 (2000)
21. Vassilev, V.K., Fogarty, T.C., Miller, J.F.: Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In: *Advances in evolutionary computing*, pp. 3–44. Springer (2003)
22. Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics* 63(5), 325–336 (1990)

## Appendix

The experimented benchmark functions described in [9] are summarised here. In this experiment  $\epsilon$  is considered as 0.0001.

### C01

Minimize

$$f(x) = - \left| \frac{\sum_{i=1}^D \cos^4(z_i) - 2 \prod_{i=1}^D \cos^2(z_i)}{\sqrt{\sum_{i=1}^D i z_i^2}} \right| \quad z = x - o$$

subject to

$$g_1(x) = 0.75 - \prod_{i=1}^D z_i \leq 0$$

$$g_2(x) = \sum_{i=1}^D -0.75D \leq 0$$

$$x \in [0, 10]^D$$

### C02

Minimize

$$f(x) = \max(z) \quad z = x - o, y = z - 0.5$$

subject to

$$g_1(x) = 10 - \frac{1}{D} \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10] \leq 0$$

$$g_2(x) = \frac{1}{D} \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10] - 15 \leq 0$$

$$h(x) = \frac{1}{D} \sum_{i=1}^D [y_i^2 - 10 \cos(2\pi y_i) + 10] - 20 \leq 0$$

$$x \in [-5.12, 5.12]^D$$

### C03

Minimize

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad z = x - o$$

subject to

$$h(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 = 0$$

$$x \in [-1000, 1000]^D$$

### C06

Minimize

$$f(x) = \max(z) \quad z = x - o, y = (x + 483.6106156535 - o)M - 483.6106156535$$

subject to

$$h_1(x) = \frac{1}{D} \sum_{i=1}^D (-y_i \sin(\sqrt{|y_i|})) = 0$$

$$h_2(x) = \frac{1}{D} \sum_{i=1}^D (-y_i \cos(0.5\sqrt{|y_i|})) = 0$$

$$x \in [-600, 600]^D$$

### C07

Minimize

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad z = x + 1 - o, y = x - o$$

subject to

$$g(x) = 0.5 - \exp(-0.1\sqrt{\frac{1}{D} \sum_{i=1}^D y_i^2}) - 3 \exp(\frac{1}{D} \sum_{i=1}^D \cos(0.1y)) + \exp(1) \leq 0$$

$$x \in [-140, 140]^D$$

### C09

Minimize

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad z = x + 1 - o, y = x - o$$

subject to

$$h_1(x) = \sum_{i=1}^D (y_i \sin(\sqrt{|y_i|})) = 0$$

$$x \in [-500, 500]^D$$

### C10

Minimize

$$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad z = x + 1 - o, y = (x - o)M$$

subject to

$$h_1(x) = \sum_{i=1}^D (y_i \sin(\sqrt{|y_i|})) = 0$$

$$x \in [-500, 500]^D$$

### C17

Minimize

$$f(x) = \sum_{i=1}^D (z_i - z_{i+1})^2 \quad z = x - o$$

subject to

$$g_1(x) = \prod_{i=1}^D z_i \leq 0$$

$$g_2(x) = \sum_{i=1}^D z_i \leq 0$$

$$h(x) = \sum_{i=1}^D (z_i \sin(4\sqrt{|z_i|})) = 0$$

$$x \in [-10, 10]^D$$

**C18**

Minimize

$$f(x) = \sum_{i=1}^D (z_i - z_{i+1})^2 \quad z = x - o$$

subject to

$$g(x) = \sum_{i=1}^D (-z_i \sin(\sqrt{|z_i|})) \leq 0$$

$$h(x) = \sum_{i=1}^D (z_i \sin(\sqrt{|z_i|})) = 0$$

$$x \in [-50, 50]^D$$