A Naïve Five-Element String Algorithm

Yanhong Cui
University of Cape Town, Private Bag, Rondebosch 7701, Cape Town, South Africa
Email: hadisna@hotmail.cn

Renkuan Guo and Danni Guo
University of Cape Town, Private Bag, Rondebosch 7701, Cape Town, South Africa
South African National Biodiversity Institute, Claremont 7735, Cape Town, South Africa
Email: Renkuan.Guo@uct.ac.za ; guo@sanbi.org

***Abstract*—In this paper, we propose a new global optimization algorithm inspired by the human life model in Chinese Traditional Medicine and graph theory, which is named as naïve five-element string algorithm. The new algorithm utilizes strings of elements from member set {0,1,2,3,4} to represent the values of candidate solutions (typically represented as vectors in *n*-dimensional Euclidean space). Except the mathematical operations for evaluating the objective function, sort procedure, creating initial population randomly, the algorithm only involves if-else logical operation. In contrast to existing global optimization algorithms, the five-element algorithm engages the simplest mathematics but reaches the highest searching efficiency.**

***Index Terms*—global optimization, five-element string, genetic algorithm, sort, naïve string algorithm**

I. Introduction

Optimization is one of challenging and active mathematical research branches. Particularly, the topic of global optimization is of critical importance because of the high demand and wide applications in science, business, economy, and industry.

Natural world is always the best teacher since the human society started, for example, fire usage, planting crops, hunting animals, Kungfu exercises, and so on. Many global optimization algorithms are imitating the behavior of biological world, for example, genetic algorithm (abbreviated as GA), ant colony algorithm, monkey algorithm, etc. However, those biological-imitated algorithms are direct "copy" of natural evolution. It is necessary to mention that ancient scientists and philosophers had established many abstract biological models for human life and natural living-beings, for example, Yin-Yang and five elements (Wu-Xing) are among them. Yin-Yang and five elements (Wu-Xing) doctrines are still active in guiding today's Traditional Chinese Medicine (abbreviated as TCM practices.

In today's part-theory dominated western scientific communities, exploration of an ancient Chinese philosophy and its potential scientific value is often regarded as nonsense, pseudo-science or waste of times. Scientists more tend to believe part-theory based genetic engineering rather than TCM. We are not resisting any advancements in science and technology, however, we also have to accept the cruel realities: only 30% of the patients or illness could be cured by modern western medicine, and on other hand, no less than 30% of the patients or illness could be cured by traditional medical treatments. More and more people accept traditional medical treatments because of cost-saving and effectiveness, for example, acupuncture and moxibustion from TCM. The theoretical foundation guiding TCM is ancient Chinese Yin-yang and Wu-Xing doctrines. Yin-yang concept plays roles in other scientific fields too. The link between Yin-Yang representation and binary number system is already well-known, and it is not difficult to reveal certain root of GA in Yin-Yang doctrine. We notice that a common rule guiding TCM doctors: identifying and eliminating factors causing in-balances within patient's body system and strengthening these factors leading the patient to his/her harmonious state according to five-element (Wu-Xing) doctrine. A natural question arises inevitably: is it possible to create an algorithm for searching global optimum with a root in five-element (Wu-Xing) doctrine? A faith inspires us is that TCM medical exercises are nothing but seek human body system optimal state, with the aid of Yin-Yang, five-element (Wu-Xing) system model.

The five-element doctrine (Wu-Xing) is different from Greek theory of four elements in formality, but both are atomic theory of substances. Ancient philosophers believed that five elements: 'metal', 'wood', 'water', 'fire', 'earth', constitute of the world. People started using the features of five-element to explain the changing of the object world in terms of the five-element's generation and deduction relationship for evolving into next sub-balanced state. However, the ever-changing nature of object world would repeatedly evolutions along the direction of generation and deduction until the system reaches its intrinsic harmonious state, even it is temporary but relatively stable. In other words, theory of five elements (Wu-Xing) as a Chinese ancient philosophy was not a merely five-substance constitution of existing objects or systems surround us but more critically the theory of five elements provides the guidance for people

Manuscript received December 2, 2008; revised January 19, 2009; accepted February 4, 2009.

Corresponding author: Renkuan Guo, Department of Statistical Sciences, University of Cape Town, Rondebosch 7701, South Africa.

doi:10.4304/jsw.4.9.925-934

to seek the intrinsic harmonious state of a system under investigation. This is the reason why Traditional Chinese Medicine is using the so-called Five-Element Doctrine as its foundation because an individual human being in good health is nothing but is in a harmonious state.

Definitely, our new five-element global optimization searching algorithm is not simulating ancient five-element objects, instead, we has established a dedicated link between mathematical objective function under investigation and the simulated population of five-element strings help the accomplishment of the optimal solution searching. In other words, the five-element algorithm followed the idea of computer simulation, not only a simple mimicking of some natural phenomenon, but also a creative idea generates from old Chinese traditional Wu-Xing Doctrine. The five-element algorithm treat the object function as a system, by simulating the five-element strings involved in this system followed by cycles of balance to generates a better system or find a better solution of object function.

It is necessary to mention here that many existing global optimization algorithms engage complicated mathematical operations, for example, algebraic operators, derivative operator, integration operator, projection operator (for parameter calibration) and control operator, however, five-element algorithm only engages the simplest logical operator: if-else. This feature greatly saves the computing time. The adjective "naïve" is added for reminding that this new algorithm does not involve complicated.

II. An Inspiring example

The objective function for illustrating purpose is the Rosenbrok function

$$f(x,y)=100\left(y-x^2\right)^2+(1-x)^2. \qquad (1)$$

The global minimal value is 0 of Rosenbrok function at $(x,y)=(1,1)$. The plot of (1) in Figure 1 offers an intuitive view on the features optimality of (1).

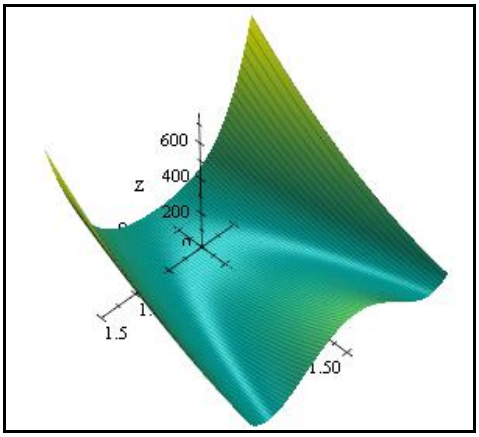

Figure 1. Plot of Rosenbrok function.

We used GA [6] to search the global minimum of Rosenbrok function. However, contrary to the global optimization searched by GA, the "global" minimal value 0.11935 at $(x,y)=(0.681,0.477)$ is reported.

The case of GA's "failure" to search the true global minimum (for given computing time) here inspires us to consider the fundamental weakness of GA. It is noticed that GA, as a global optimization algorithm, differs from many other algorithms.

Let $f(\underline{x})$ be the objective function, where $\underline{x}=(x_1,x_2)^T\in\mathbb{D}\subset\mathbb{R}^2$. In many optimization searching algorithms, a typical exercise is trying to improve the optimality within the neighborhood. It is obvious that the increment in $\underline{x}$ approach typically leads to a local optimum.

GA does not work on system state $\underline{x}\in\mathbb{D}\subset\mathbb{R}^n$ of the objective function $f(\underline{x})$ directly, rather it uses string like $00110011000100111010111100$ for the representing the state $\underline{x}^T=(x_1,x_2,\cdots,x_n)$ and hence may possess better global coverage. However, GA string member set is $\{0,1\}$. Inevitably, the change in string may not change the state $\underline{x}^T=(x_1,x_2,\cdots,x_n)$ efficiently for covering the whole domain because the element change in a string is 1.

III. String Representation of the state of objective function

An improvement strategy is to expand string member set. Now let us formally establish the string representation related concepts.

Definition 3.1 A string is a sequence of integers, denoted by $n_1n_2\cdots n_p$. Number p is called the length of a string.

For operational convenience, a string may be repressed by a row vector, $\underline{n}^T=(n_1,n_2,\cdots,n_p)$.

Definition 3.2 The collection of the elements for constructing a string, denoted by $\{0,1,\cdots,s-1\}$, is termed as an element set for a string. s is called the size of the element set of a string (i.e., the number of elements in the element set).

Conjecture 3.3 The size of the element set of a string used in a naïve string algorithm is a prime number.

In GA, the size of the element set $\{0,1\}$ is prime number 2. Prime number 3, 5, 7, 11, etc can also be used. If the size of the element set is 7, then the element set is $\{0,1,2,3,4,5,6\}$.

The length of a string p should be at least $n(s+1)$.

Definition 3.4 Let $\underline{x}=(x_1,x_2,\cdots,x_n)^T\in\mathbb{D}\subset\mathbb{R}^2$ denote system state, which is also representing the candidate solution. Then the length of the string representing $\underline{x}$ is $p>nu$ if the size of the string element set is s, $u>s$, u is called the basic unit size of a string. The string representation for $(x_1,x_2,\cdots,x_n)^T$ is

$$e_1 e_2 \cdots e_u e_{u+1} \cdots e_{2u} \cdots e_{(n-1)u+1} \cdots e_{nu} \tag{2}$$

An intuitive correspondence between the state $\underline{x}$ and the representing string is

$$\underbrace{e_1 e_2 \cdots e_u}_{x_1} \underbrace{e_{u+1} \cdots e_{2u}}_{x_2} \cdots \underbrace{e_{(n-1)u+1} \cdots e_{nu}}_{x_n} \tag{3}$$

Lemma 3.5 Let the system state be $\underline{x} \in \mathbb{D} \subset \mathbb{R}^n$, and $\underline{e}$ be a string representation (of the system state) with element set size s and string length $n(s+1)$. Let $u_r = \max_{1 \le i \le n} \{u_{\max,i} - u_{\min,i}\}$, where $\underline{u}_{\min} \le \underline{x} \in \mathbb{D}$, $\underline{u}_{\max} \ge \underline{x} \in \mathbb{D}$. The weight matrix $O = (o_{ij})_{n \times nu}$ with the i^{th} row vector $\underline{o}_i^T$ having a form

$$\left(0, 0, \cdots, 0, \cdots, \frac{s^s}{s^{s+1}}, \frac{s^{s-1}}{s^{s+1}}, \cdots, \frac{s^0}{s^{s+1}}, \cdots, 0, 0, \cdots, 0\right) \tag{4}$$

where the nonzero weights are located at the i^{th} segment. Then the system state is a linear transformation of the s-element string representation

$$\underline{x} = \underline{u}_{\min} + u_r O \underline{e} \tag{5}$$

Definition 3.6 If e is an element of a string with element set $\{0,1,2,3,\cdots,s-1\}$, then the value changing rule is

$$e = \begin{cases} e+1 & \text{if } e \in \{0,1,2,\cdots s-2\} \\ 0 & \text{if } e = s-1 \end{cases} \tag{6}$$

In the remaining sections of this paper, we will use 5-element string for illustration and the establishment of the naïve string algorithm.

IV. FIVE-ELEMENT STRING REPRESENTATION

For clarity, we will use numerical examples for illustrating the necessity and advantages of string representation.

Example 4.1 Let $\mathbb{D} \triangleq [u_{\min}, u_{\max}] \times [u_{\min}, u_{\max}]$ be the domain for an objective function $f(x_1, x_2)$. Assume that a string 1 2 4 3 0 1 2 4 3 2 11 represents (x_1, x_2): the first 6 elements, i.e., 1 2 4 3 0 1, in the string stand as x_1 and the second 6 elements, i.e., 2 4 3 2 11, stand as x_2. The element set is $\{0,1,2,3,4\}$, the size of element set is 5, the basic unit $u = 5+1 = 6$. The length of the string 1 2 4 3 0 1 2 4 3 2 11 is $2u$ =12, which is the number of units occupied in computer.

Mathematically, the linear system linking the five-element string and the system state can be expressed by

$$\begin{cases} x_1 = u_{\min} + (u_{\max} - u_{\min}) \sum_{j=1}^{6} e_j \dfrac{5^{6-j}}{5^6} \\ x_2 = u_{\min} + (u_{\max} - u_{\min}) \sum_{j=7}^{12} e_j \dfrac{5^{12-j}}{5^6} \end{cases} \tag{7}$$

Let

$$O_{2\times 2u} = \begin{bmatrix} \frac{5^5}{5^6} & \cdots & \frac{5^0}{5^6} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{5^5}{5^s} & \cdots & \frac{5^0}{5^s} \end{bmatrix}, \underline{e}_{2u\times 1} = \begin{bmatrix} e_1 \\ \vdots \\ e_6 \\ e_7 \\ \vdots \\ e_{12} \end{bmatrix} \tag{8}$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \underline{u}_{\min} = \begin{bmatrix} u_{\min} \\ u_{\min} \end{bmatrix}, u_r = u_{\max} - u_{\min}$$

Then a matrix equation for string to state vector transformation is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u_{\min} \\ u_{\min} \end{bmatrix} + (u_{\max} - u_{\min}) \begin{bmatrix} \frac{5^5}{5^6} & \cdots & \frac{5^0}{5^6} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{5^5}{5^6} & \cdots & \frac{5^0}{5^6} \end{bmatrix} \begin{bmatrix} e_1 \\ \vdots \\ e_6 \\ e_7 \\ \vdots \\ e_{12} \end{bmatrix} \tag{9}$$

Matrix O is actually a weighting system which promotes the changes in (x_1, x_2) according to the location of an individual member in the string as well as the changing size of the member.

In other words, the mechanism underlying the usage of string lies on that the weighting system, i.e., $\left\{\frac{5^5}{5^6}, \frac{5^4}{5^6}, \frac{5^3}{5^6}, \frac{5^2}{5^6}, \frac{5^1}{5^6}, \frac{5^0}{5^6}, 0, 0, 0, 0, 0, 0\right\}$, assigned to the 6 members in the first half of the string and $\left\{0, 0, 0, 0, 0, 0, \frac{5^5}{5^6}, \frac{5^4}{5^6}, \frac{5^3}{5^6}, \frac{5^2}{5^6}, \frac{5^1}{5^6}, \frac{5^0}{5^6}\right\}$, the weighting system assigned to the 6 members in the second half of the string create the possibility that change in the member of the string will have different impacts.

A string, denoted by $e_1 e_2 \cdots e_6 e_7, e_8 \cdots e_{12}$, the blue-color members are the first half of the string, representing x_1, the red-color members are the second half of the string, representing x_2. Logically, changes in e_1 and e_7 will result in largest changes in x_1 and x_2 respectively, because the highest weight 0.2 is assigned to them, while changes in e_6 and e_{12} will result in the smallest changes in x_1 and x_2 respectively, because the lowest weight 0.000064 is assigned to them. Therefore, a well-constructed string element change scheme will have a balanced global searching capability as well as local fine-tune capacity.

Example 4.2 (Continued) Define $u_{\min} = -10^{10}$, $u_{\max} = +10^{10}$, then $u_r = u_{\max} - u_{\min} = 2\times 10^{10}$. String 1: 1 2 4 3 0 1 2 4 3 2 11 used in Example 3.4 is the base for observing the impacts from string member changes. String 2 changes the first element of the String 1 by adding 1 and the seventh element of the String 1 by adding 1, which is the smallest shift in size at highest weight 0.2. The change in x_1 and x_2 is quite large with distance 5656854249.5. However, String 3 changes the sixth element of the String 1 by adding 3 and the seventh element of the String 1 by adding 3, which is the largest shift in size at highest weight 0.000064. The change in x_1 and x_2 is much small with distance 202276452.4. Table I summaries the changes and impacts.

TABLE I.
THE IMPACTS OF WEIGHTS IN GLOBAL SEARCHING AND LOCAL TUNE-UP

String	x_1	x_2	$\lvert\Delta x\rvert$
1 2 4 3 0 1 2 4 3 2 1 1	-3662720000	1751680000	
2 2 4 3 0 1 **3** 4 3 2 1 1	337280000	5751680000	5656854249.5
1 2 4 3 0 **4** 2 4 3 2 1 **4**	-3460480000	1755520000	202276452.4

It is important to emphasize here that the value of a string depends on three factors: (1) value of individual element in a string from {0, 1, 2, 3, 4}; (2) the location (or position) of a specific element e_i; (3) the combination of all elements appeared in the given string. Formally, let us define the five-element if-else operator, called as λ operator.

Definition 4.3 (**λ operator**) Let $e \in \{0,1,2,3,4\}$, then

$$\lambda[e] = \begin{cases} e+1 & \text{if } e \in \{0,1,2,3\} \\ 0 & \text{if } e = 4 \end{cases} \tag{10}$$

$\lambda^{(l)}[\ ]$ is l^{th} order λ operator, which repeats λ operation *m* times.

Definition 4.4 (Modulo operator) Let d be an positive integer, q be the quotient and r remainder *r* satisfying

$$d = nq + r \tag{11}$$

Then we write the modulo operation as

$$d \bmod(q) = r \tag{12}$$

Definition 4.5 Let $\underline{e} = (e_1, e_2, \cdots, e_g)$ be a five-element string, then the λ operation on a string is a component-wise operation, i.e.,

$$\lambda[\underline{e}] = (\lambda[e_1], \lambda[e_2], \cdots, \lambda[e_g]) \tag{13}$$

Furthermore, let $A = (e_{ij})_{h\times g}$ be a five-element matrix, i.e., $e_{ij} \in \{0,1,2,3,4\}$, then

$$\lambda[A] = (\lambda[e_{ij}])_{h\times g} \tag{14}$$

Proposition 4.6 $\lambda^{(l)}[e] = \lambda^{(l \bmod (4))}[e]$, where $\lambda^{(0)}[e] \triangleq e$.

Proof: Note that $e \in \{0,1,2,3,4\}$, the number *e* only has five choices. For example, $e = 0$,

$$\lambda^{(5)}[e] = \lambda^{(4)}[1] = \lambda^{(3)}[2] = \lambda^{(2)}[3] = \lambda^{(1)}[4] = 0 \tag{15}$$

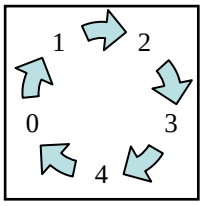

Figure 2. $\lambda[\ ]$ operation cycle

For any element $e \in \{0,1,2,3,4\}$, one-time $\lambda[\ ]$ operation shifts the element *e* from current position into the next *1st* position along the cycle shown in Figure 2. Hence *l*-time $\lambda[\ ]$ operation shifts the element *e* from current position into the next l^{th}-position along the cycle. Further, due to the fact that five-element member set {0,1,2,3,4} only has five members in it, the period of the cycle is 5. Therefore, $\lambda^{(l)}[e] = \lambda^{(l \bmod(4))}[e]$ since 0 is the first member of the element set.

Proposition 4.7 For any given five-element string $\underline{e}$, the five-time $\lambda[\ ]$ operated strings form a string cycle. In other words, $\{\underline{e}, \lambda^{(1)}[\underline{e}], \lambda^{(2)}[\underline{e}], \lambda^{(3)}[\underline{e}], \lambda^{(4)}[\underline{e}],\}$ is a string cycle.

Definition 4.8 Let

$$\underline{x}^{(k)} = \underline{u}_{\min} + u_r O\lambda^{(k)}[\underline{e}], \quad k = 0,1,2,3,4 \tag{16}$$

be the corresponding system state of $\lambda^{(k)}[\underline{e}]$. Then $\{\underline{x}, \underline{x}^{(1)}, \underline{x}^{(2)}, \underline{x}^{(3)}, \underline{x}^{(4)}\}$ is the system state cycle and $\{f(\underline{x}), f(\underline{x}^{(1)}), f(\underline{x}^{(2)}), f(\underline{x}^{(3)}), f(\underline{x}^{(4)})\}$ is the objective function value cycle respect to the string cycle $\{\underline{e}, \lambda^{(1)}[\underline{e}], \lambda^{(2)}[\underline{e}], \lambda^{(3)}[\underline{e}], \lambda^{(4)}[\underline{e}],\}$.

Proposition 4.9 Let

$$\begin{aligned} f_{\min} &= \min\{f(\underline{x}), f(\underline{x}^{(1)}), f(\underline{x}^{(2)}), f(\underline{x}^{(3)}), f(\underline{x}^{(4)})\}, \\ f_{\max} &= \max\{f(\underline{x}), f(\underline{x}^{(1)}), f(\underline{x}^{(2)}), f(\underline{x}^{(3)}), f(\underline{x}^{(4)})\} \end{aligned} \tag{17}$$

Then the objective function cycle will demonstrate three patterns: (i) $f(\underline{x}) = f_{\min}$, i.e., the remaining four objective function values are above the cycle starting value $f(\underline{x})$; (ii) $f(\underline{x}) = f_{\max}$, i.e., the remaining four objective function values are below the cycle starting value $f(\underline{x})$; (iii) $f_{\min} \le f(\underline{x}) \le f_{\max}$, i.e., the cycle

starting value $f(\underline{x})$ falls between cycle minimum and maximum.

Remark 4.10 The weight matrix O in string and system state linking equation $\underline{x} = \underline{u}_{\min} + u_r O \underline{e}$ reveals the ever-changing and controllable character of five element string representation. And the three cycle patterns of objective function values with respect to string cycles reveal that $\lambda[\ ]$ operations guarantee the chance for global optimum searching.

Now it is ready to state the scheme of the naïve five-element string algorithm.

V. A NAÏVE GLOBAL OPTIMUM SEARCH SCHEME

A few terms are defined first.

***Stopping time*:** The algorithm stops after running for an amount of time in seconds, which is specified as stopping time.

Population size*:** The population size defines numbers of rows of matrices, denoted by ***N.

***String length*:** The string length defines the number of elements in each five-element string.

n: the dimension of objective function.

$u_{\min}$: the lower bound value of input variables.

$u_{\max}$: the upper bound value of input variables.

Before the searching scheme enters algorithm loop the naïve nature of the scheme requires the creation of a candidate solution string population. Randomly select numbers from member set {0,1,2,3,4} uniformly and independently and put them into strings until the string population is established. It is obvious that the discrete uniform random number nature eliminates any possible bias for the starting the algorithm.

Stochastic initialization: Randomly generate ***2N,*** say, *N=100,* five-element strings as candidate solutions, then divide the candidate solutions into two string vectors (two matrices of elements), The first string vector is denoted by $Q_{\min}$ and the second by $Q_{\max}$. The searching range for the i^{th} component of system state $\underline{x}$ is $[u_{\min}, u_{\max}]$, i.e., $u_{\min} \le x_i \le u_{\max}$.

Searching loop:

Step 1: 2*N* string cycles creation. By applying $\lambda[\ ]$ to $Q_{\min}$ and $Q_{\max}$ respectively, ten string vectors (including $Q_{\min}$ and $Q_{\max}$), denote them by Q_i, $i = 1,\cdots,5,6,\cdots,10$. Note that $Q_1 = Q_{\min}$ and $Q_6 = Q_{\max}$. Mathematically,

$$\begin{aligned} Q_i &= \lambda^{(i-1)}[Q_{\min}],\ i = 1,2,3,4,5 \\ Q_i &= \lambda^{(i-6)}[Q_{\max}],\ i = 6,7,8,9,10 \end{aligned} \tag{18}$$

Mathematically, **step 1** is creating 200 (*2N in general*) string cycles according to **Proposition 4.7**, which paves the way toward the global optimum searching.

Step 2: Rank the strings. Fitness checking and best-worst string vectors creation. It is divided into three sub-steps:

(1) Combine Q_i, $i = 1,2,\cdots,10$ into a super string vector, denoted by Q.

(2) Sort the 1000 strings in Q by ascending order according to objective function values with respect to the 1000 strings, and denote the ranked string vectors as Q'.

(3) Define the top 100 strings of Q' as $Q'_{\min}$ and the bottom 100 strings of Q' but reverse them in descending order as $Q'_{\max}$.

Mathematically, **Step 2** is utilizing the 200 cycles of objective function values in which 200 minimum candidate solutions and maximum candidate solutions are constructed according to **Proposition 4.9**.

Step 3: Best element select and worst element remove.

Intuitively, this step utilizes genetic engineering ideas: for seeking the best healthy gene combinations it is necessary to keep the best individual gene in the particular position within the gene sequence and also remove the worst individual gene from the particular position within the gene sequence. What we will act is just an imitation to gene selecting and removing in the five-element string sequences created in **Step 2**, i.e., $Q'_{\min}$ and $Q'_{\max}$ in terms of $\lambda[\ ]$ operation. This is divided into two sub-steps.

(1) **Packed-Rolling operation**. This sub-step performs operations within $Q'_{\min}$ and $Q'_{\max}$ respectively.

If we aim at search global minimum of the given objective function, strings in $Q'_{\max}$ will be regarded as worse gene sequences and thus the first string corresponding to the maximum objective function value is the worst one. Similarly, strings in $Q'_{\max}$ will be regarded as better gene sequences and thus the first string corresponding to the minimum objective function value is the best one.

The Matlab pseudo-code of packed rolling operation is listed as follows.

Assume ranked candidate solutions denote as matrix Q, the matrix size is row multiply column.

```
for i=1:1: row-4
for j=1:1: column
if  Q(i,j)== Q(i+1,j) && Q(i,j)~=4
Q(i+1,j)=Q(i+1,j)+1;
elseif Q(i,j)== Q(i+1,j) && Q(i,j)==4
Q(i+1,j)=0;
elseif Q(i,j)== Q(i+2,j) && Q(i,j)~=4
Q(i+2,j)= Q(i+2,j)+1;
elseif Q(i,j)== Q(i+2,j) &&Q(i,j)==4
Q(i+2,j)=0;
elseif Q(i,j)== Q(i+3,j) && Q(i,j)~=4
Q(i+3,j)= Q(i+3,j)+1;
elseif Q(i,j)== Q(i+3,j) && Q(i,j)==4
Q(i+3,j)=0;
```

```
elseif Q(i,j)== Q(i+4,j) && Q(i,j)~=4
Q(i+4,j)= Q(i+4,j)+1;
elseif Q(i,j)== Q(i+4,j) && Q(i,j)==4
Q(i+4,j)=0;
end
end
end
```

Verbally, Packed-Rolling operation can explained as follows: Defined five strings as a "package", within the selected package, the best string is the first item of the package. Then examining the first element (location) in the second string, if the element repeats the first element of the best string, $\lambda[\ ]$ operator should be applied to the repeated element one-time. Next, the second element (position) of the second string is examined, if it repeats the second element of the best string, $\lambda[\ ]$ operator should be applied. Keep on the checking every individual element of the second string until the last one (position). Repeat the check and replacement operations with respect to the third, fourth and the fifth string in the package. Then we select the second package, in which the second string in the string vector Q will be defined as the first string of this package. Perform the check and replacement operations within the second package until finished. Then the third package is defined where the third string in the string vector Q, and perform the check and replacement operations within the third package, and so on until the ROW-4th package is defined and checked.

At the end of Packed-Rolling, all the strings are re-evaluated via $f(\underline{u}_{min} + u_r O\underline{e})$, and accordingly re-ranked in ascending order for forming new string vector $Q_{\min}$ and in descending order for $Q_{\max}$.

(2) **Excise worst elements.** Different from Packed-Rolling sub-step, this operation is performed by comparing the corresponding elements between $Q_{\min}$ and $Q_{\max}$. Intuitively, excising the worst elements with respect to the best strings from the opposite string vector is similar to excising bad gene from the gene sequence by comparing to a healthy gene sequence.

In this sub-step, two corresponding strings (candidate solutions) from $Q_{\min}$ and $Q_{\max}$ each are selected and compare their corresponding elements sequentially. If we are seeking global minimum, then the strings from $Q_{\max}$ will be "sick" ones while the strings from $Q_{\min}$ will be regarded as "healthier" ones. For the same location, if the healthier string contains element being the same as the element at the same location in the "sick" string, this individual element at this location should be excised and replaced by the element at the same location from the best string (i.e., the first string in $Q_{\min}$).

The pseudo-code of Matlab describes how to excise unhealthy elements from relevant the strings.

Assume string vector Q is for generating global minimum, and string vector $Q1$ is for generating global maximum.

```
for i=1:1:row-1
for j=1:1:column
if Q1(1,j)== Q(i+1,j)
Q(i+1,j)= Q(1,j);
end
if Q(1,j)== Q1(i+1,j)
Q1(i+1,j)= Q1(1,j);
end
end
end
```

In the excising operation, the first strings in Q and $Q1$'s are defined as the best elements and the worst elements respectively. If for a given location the element in Q repeats the element at the same location in $Q1$, this particular element should be excised and replaced by the element at the same location of the first string in Q.

At the beginning of scheme running, the excising operation might cause the convergence too quick (such that trap into local optimum), and during the whole algorithm running period, it also might cause some healthy elements been excised. However **Proposition 4.9** guarantees the success of the scheme as what we pointed in **Remark 4.10**.

At the end of **Step 3**, new string vector $Q''_{\min}$ in ascending order and $Q''_{\max}$ in descending order will be generated.

The flow chart of the naïve string optimization searching scheme is shown in Figure 3.

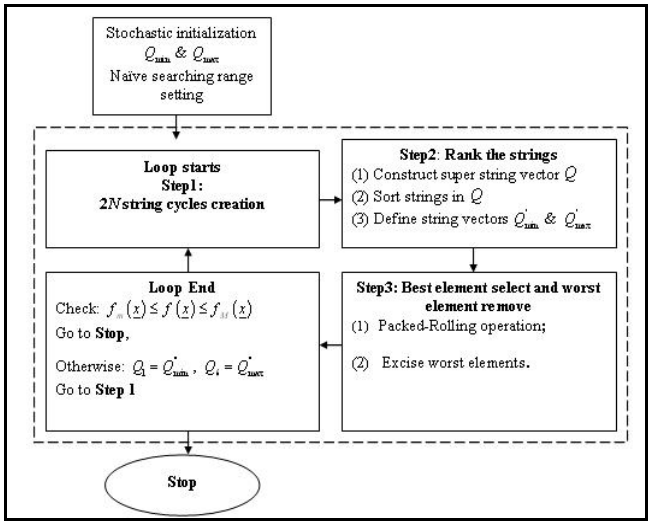

Figure 3. Flow chart of naïve five-element string algorithm.

The naïve five-element string algorithm can be stated as following:

Initialization (generating string population $Q_{\min}$ and $Q_{\max}$ stochastically).

Start loop
1. 2*N* string cycles creation;
2. Rank the strings;
3. Best element select and worst element remove;
4. Check the loop stop criteria: (yes, GoTO 1, yes, Loop Stops);

End loop

VI. ILLUSTRITIVE EXAMPLES

We use five-element naïve string algorithm to search the global optimum for three objective functions: Rosenbrok function, Rastrigin function and Griewank function. Also, we use GA performing the three functions as comparison.

A. Rosenbrok function

$$f(x_1,x_2)=100\left(x_2-x_1^2\right)^2+(1-x_1)^2 \qquad (19)$$

The naïve string algorithm searching by 52 loops gives $f_{\min}=0.0015$, and the global minimum state $\left(x_1^m,x_2^m\right)=(1.0117,1.0199)$. The searching area is $\mathbb{D}\triangleq\left[-10^6,10^6\right]\times\left[-10^6,10^6\right]$.

B. Rastrigin function

$$f(x_1,x_2)=2+x_1^2-\cos(18x_1)-\cos(18x_2) \qquad (20)$$

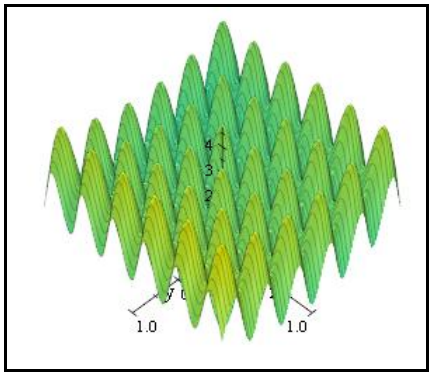

Figure 4. 3D-plot of Rastrigin function.

The global minimum is 0 at $\left(x_1^m,x_2^m\right)=(0,0)$ and it is well-known that in area $[-1,1]\times[-1,1]$ there are more than 50 local minima spreading as a lattice around the global minimum.

The naïve string algorithm searching in the area $\mathbb{D}\triangleq\left[-10^6,10^6\right]\times\left[-10^6,10^6\right]$ by 26 loops gives the global minimum 0 at $\left(x_1^m,x_2^m\right)=(0,0)$.

C. Griewank function

This function is 10-dimensional. In the cube $[-600,600]^{10}$, there are thousands of local minima around and the global minimum 0 at the origin.

Using naïve string algorithm to search in the cube $\mathbb{D}\triangleq\left[-10^6,10^6\right]^{10}$, by 77 loops, the algorithm locates $\left(x_1^m,\cdots x_8^m,x_9^m,x_{10}^m\right)=(-0.0041,\cdots,-0.0041,0.0041,0.0041)$ which gives global minimum 0.00010846.

TABLE II.
COMPARISONS BETWEEN GA AND FIVE-ELEMENT NAÏVE STRING ALGORITHM

Function	algorithm	Searching Cube	Loops	Global min
(1)	GA	$\left[-10^6,10^6\right]^2$	57	0.11935
	NSA	$\left[-10^6,10^6\right]^2$	52	0.0015
(2)	GA	$\left[-10^6,10^6\right]^2$	51	1.2178
	NSA	$\left[-10^6,10^6\right]^2$	26	0.000
(3)	GA	$\left[-10^6,10^6\right]^{10}$	52	0.12506
	NSA	$\left[-10^6,10^6\right]^{10}$	77	0.000
(3)*	GA	$[-600,600]^{10}$	35	0.03243650
	NSA	$[-600,600]^{10}$	7	0.0001085

VII. CONCLUDING REMARKS

It is quite promising that the naïve five-element string algorithm has demonstrated its excellent global searching capability with competitive speed (measured by loop number) and competitive quality (in terms of the global minimum). The naïve sting algorithm offers global minimum and maximum at the same time. It is also exciting that when the search "cube" is reduced, the searching loops decreases greatly and the search quality increases without any doubts. However, the "reduced" search cube implies a constrained optimization or more information is required for the objective function. Such a demand is often impossible to be satisfied in GIS modelling exercises.

The algorithm has three fundamental features: (1) The states of the system is represented by strings of 5 elements {0,1,2,3,4} and hence the search of the optimal state(s) is realized by string manipulations; (2) A weighting system is created for a balanced global and local search to avoid the scheme trapping in local optimum; (3) The string operation is a pseudo-linear transformation, which involves if-else logical operator, such that the searching the optimum of a nonlinear multivariate objective function is essentially linear.

Finally, there is a trend in scientific research – complication. It is true that real world is complicated. However, any complicated phenomenon can be decomposed into simple ones. It is fair to sat that to pursue simple one, rather, complicated should be the basic goal of scientists. Our naïve five-element string algorithm is the simplest one with high efficiency and worth to be promoted.

APPENDIX: MATLAB CODES FOR NFESA

```
function [Bestfitness,variables]=FE
```

```
% The Naïve Five-Element Algorithm is initialed by Yanhong Cui and Professor Renkuan Guo
% University of Cape Town, Statistical Science Department
% The Matlab codes are written by Yanhhong Cui (Copyright reserved)

% Bestfitness=Best fitness value of objective function (Should close or equal to Zero)
% variables=Response values of input variables of Best fitness of objective function
% HE is function name,abbreviate of Harmony Elements algorithm

%Operation Example:

% [Bestfitness,variables]=HE
% Please enter the function name(must enter):example @function name
% @rastriginsfcn
% Please enter population size you want(could be empty):default 100
%
% Please enter loop times you want(could be empty):default 100
%
% Please enter String length of each variables(could be empty):default 12
% 30
% please enter upper bound of variables you want(could be empty):default 10^6
%
% please enter lower bound of variable you want(could be empty):default -10^6
%
% Please enter number of variables(must enter):example 2
% 2
%
% Bestfitness =
%
%     0
%
%
% variables =
%
%   1.0e-008 *
%
%    0.0349    0.3260

name=input('Please enter the function name(must enter):example @function name\n');
%Setup objective function name: example @function name
Size=input('Please enter population size you want(could be empty):default 100\n');
%Setup population matrix size: default 100.(100 candidate solutions of objective function)
if isempty(Size)
   Size=100;
end
%Default setup:100
G=input('Please enter loop times you want(could be empty):default 100\n');
%Setup loop times: default 100.(HEA will end the optimization of objective function after 100 loop times)
if isempty(G)
   G=100;
end
%Default setup:100
Codel=input('Please enter String length of each variables(could be empty):default 12\n');
%Setup String length of each input variables: default 12.If the obejctive
%Function have 2 variables, then the length of String is 12*2=24.
if isempty(Codel)
   Codel=12;
end
%Default setup:12
umax=input('please enter upper bound of variables you want(could be empty):default 10^6\n');
%Setup upper bound of variables: default 10^6. (X1,X2,X3,...<=upper bound value)
if isempty(umax)
   umax=10^6;
end
%Dault setup:10^6
umin=input('please enter lower bound of variable you want(could be empty):default -10^6\n');
%Setup lower bound of variables:default -10^6.(X1,X2,X3,...>=lower bound value)
if isempty(umin)
   umin=-10^6;
end
%Dault setup:-10^6
n=input('Please enter number of variables(must enter):example 2\n');
%Setup number of variables i.e. if your equation with 2 variables then
%Input with 2.
mm=cell(1,n);
mmb=cell(1,n);
%Program setup (Please don't modify)
Q=cell(1,5);
%Setup 5 closed cell for store Q{1,1};Q{1,2};Q{1,3};Q{1,4};Q{1,5} five matrix, every rows in matrix
%standard as indiviudal solutions of all variables.
Q{1,1}=round(4*rand(Size,n*Codel));
% Generate a random population matrix with numbers
% 1's,2's,3's,4's,5's.Matrix min initialization
for j=1:1:n*Codel
   pp(j)=0;
end
%Initial best string(best candidate solution)checking string.The string
%will note the change of each element of the best string with 1,unchange
%with 0.
```

```
EX=round(4*rand(Size,n*Codel));
% Generate a random population matrix with numbers
% 1's,2's,3's,4's,5's.Matrix max initialization

%Initial best string expanding matrix
TempE(1,:)=Q{1,1}(1,:);
for k=1:1:G
% start loop
   time(k)=k;
   for j=1:1:n*Codel
     TT1(j,:)=Q{1,1}(1,:);
   end
TT2=TT1;TT3=TT1;TT4=TT1;
QQ1=round(4*rand(n*Codel,n*Codel));
QQ2=round(4*rand(n*Codel,n*Codel));
QQ3=round(4*rand(n*Codel,n*Codel));
QQ4=round(4*rand(n*Codel,n*Codel));
% counting looping times

   for j=1:1:n*Codel
   if Q{1,1}(1,j)==0
      TT1(j,j)=1;TT2(j,j)=2;TT3(j,j)=3;TT4(j,j)=4;
   elseif Q{1,1}(1,j)==1
      TT1(j,j)=2;TT2(j,j)=3;TT3(j,j)=4;TT4(j,j)=0;
   elseif Q{1,1}(1,j)==2
      TT1(j,j)=3;TT2(j,j)=4;TT3(j,j)=0;TT4(j,j)=1;
   elseif Q{1,1}(1,j)==3
      TT1(j,j)=4;TT2(j,j)=0;TT3(j,j)=1;TT4(j,j)=2;
   elseif Q{1,1}(1,j)==4
      TT1(j,j)=0;TT2(j,j)=1;TT3(j,j)=2;TT4(j,j)=3;
   end
   end

%Followed rule with pp(1) remaining
%the element of best string,pp(0) expanding the elements
%with cycle:0->1->2->3->4->0
   for i=1:1:Size
      for j=1:1:n*Codel
      if Q{1,1}(i,j)==0

Q{1,2}(i,j)=1;Q{1,3}(i,j)=2;Q{1,4}(i,j)=3;Q{1,5}(i,j)=4;
      elseif Q{1,1}(i,j)==1

Q{1,2}(i,j)=2;Q{1,3}(i,j)=3;Q{1,4}(i,j)=4;Q{1,5}(i,j)=0;
      elseif Q{1,1}(i,j)==2

Q{1,2}(i,j)=3;Q{1,3}(i,j)=4;Q{1,4}(i,j)=0;Q{1,5}(i,j)=1;
      elseif Q{1,1}(i,j)==3

Q{1,2}(i,j)=4;Q{1,3}(i,j)=0;Q{1,4}(i,j)=1;Q{1,5}(i,j)=2;
      elseif Q{1,1}(i,j)==4

Q{1,2}(i,j)=0;Q{1,3}(i,j)=1;Q{1,4}(i,j)=2;Q{1,5}(i,j)=3;
      end
      if  EX(i,j)==0

Q{1,6}(i,j)=1;Q{1,7}(i,j)=2;Q{1,8}(i,j)=3;Q{1,9}(i,j)=4;
      elseif EX(i,j)==1

Q{1,6}(i,j)=2;Q{1,7}(i,j)=3;Q{1,8}(i,j)=4;Q{1,9}(i,j)=0;
      elseif EX(i,j)==2

Q{1,6}(i,j)=3;Q{1,7}(i,j)=4;Q{1,8}(i,j)=0;Q{1,9}(i,j)=1;
      elseif EX(i,j)==3

Q{1,6}(i,j)=4;Q{1,7}(i,j)=0;Q{1,8}(i,j)=1;Q{1,9}(i,j)=2;
      elseif EX(i,j)==4

Q{1,6}(i,j)=0;Q{1,7}(i,j)=1;Q{1,8}(i,j)=2;Q{1,9}(i,j)=3;
      end
      end
   end
%From matrix Q{1,1} to generate
Q{1,2},Q{1,3},Q{1,4},Q{1,5} followed rule:0->1->2-
>3->4->0
%From matrix EX to generate
Q{1,6};Q{1,7};Q{1,8};Q{1,9}followed rule:0->1->2->3-
>4->0
if k>40

E=[Q{1,1};Q{1,2};Q{1,3};Q{1,4};Q{1,5};Q{1,6};Q{1,7
};Q{1,8};Q{1,9};EX;TT1;TT2;TT3;TT4];
else

E=[Q{1,1};Q{1,2};Q{1,3};Q{1,4};Q{1,5};Q{1,6};Q{1,7
};Q{1,8};Q{1,9};EX;QQ1;QQ2;QQ3;QQ4];
end
% Let E equal to combination of 10 expanding Matrix
and a best string expanding Matrix TT.

   for s=1:1:10*Size+4*n*Codel
      m=E(s,:);
      for v=1:1:n
      y(v)=0;
      mm{1,v}=m(Codel*(v-1)+1:1:v*Codel);
      for i=1:1:Codel
         y(v)=y(v)+mm{1,v}(i)*5^(Codel-i);
      end
      x(v)=(umax-umin)*y(v)/(5^Codel)+umin;
      r(1,v)=x(v);
      end
      F(s)=name(r);
   end
% Evaluate the objective function value of each
string(candidate solution)
% of cobination Matrix E
   Ji=1./F;
% Setp up inverse measure (image plot using)
   BestJ(k)=max(Ji);
% Setp up inverse measure of best fitness (image plot
using)
   fi=F;
   [Oderfi,Indexfi]=sort(fi);
% Ranking Strings of Matrix E from Best fitness to worst
one.(From min value of objective function to max)
   [Oderfi1,Indexfi1]=sort(fi,'descend');
% Ranking Strings of Matrix E from Worst fitness to best
one.(From max value of objective function to min)
   Bestfitness=Oderfi(1);
```

```
%Record best fitness value.It will showing as a result. The best fitness
%value will close or equal to Zero
   for i=1:1:Size
      TempE(i,:)=E(Indexfi(i),:);
      TempE1(i,:)=E(Indexfi1(i),:);
   end
%Rank Matrix E from best to worst fitness and record first 100 (population
%matrix size) as a new matrix TempE
%Rank Matrix E from worst to best fitness and record first 100 (population
%matrix size) as a new matrix TempE1
%P.S Delete other strings don't record
   BestS=TempE(1,:);
%Record Best fitness string( not value) as string BestS
   bfi(k)=Bestfitness;
   BS(k,:)=BestS;
%Record Best fitness string of each loop and combined as BS
   for j=1:1:n*Codel
   if TempE(1,j)==Q{1,1}(1,j)
   pp(j)=1;
   end
   end
%The string note the change of each element of the best string with 1,unchange with 0.

   for i=1:1:Size-4
   for j=1:1:n*Codel
         if TempE(i,j)==TempE(i+1,j) && TempE(i,j)~=4
            TempE(i+1,j)=TempE(i+1,j)+1;
         elseif TempE(i,j)==TempE(i+1,j) && TempE(i,j)==4
            TempE(i+1,j)=0;
         elseif TempE(i,j)==TempE(i+2,j) && TempE(i,j)~=4
            TempE(i+2,j)=TempE(i+2,j)+1;
         elseif TempE(i,j)==TempE(i+2,j) && TempE(i,j)==4
```

ACKNOWLEDGMENT

This research is partially supported by South African National Research Foundation Grant FA2006042800024.

REFERENCES

[1] C.T. Benson and N.E. Losey, "On a Graph of Hoffman and Singleton," *J. Combin. Th. Ser. B*, 11, pp. 67-79, 1971.

[2] J.L. Crosby, *Computer Simulation in Genetics*, John Wiley & Sons, London, 1973.

[3] A. Fraser, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Aust. J. Biol. Sci.*, 10, pp. 484-491, 1957.

[4] A. Fraser and B. Donald, *Computer Models in Genetics*, McGraw-Hill, New York, 1970.

[5] T. Weise, *Global Optimization Algorithms - Theory and Application*, 2nd Edition, 2008. http://it-weise.de/

[6] Matlab help Example: Rastrigin's function: Getting Started with Genetic algorithm (Genetic Algorithm and Direct Search Toolbox).

Mr Yanhong Cui is registered as a PhD student at the Department of Statistical Sciences, University of Cape Town, Cape Town, South Africa, since 2008. His research interest is statistical computations, particularly, in small sample statistics.
He has published fourteen peer-reviewed papers in various international conferences.

Dr Guo is a full Professor in the Department of Statistical Sciences, University of Cape Town. His research interests is statistical data analysis in random and fuzzy environments with applications including financial market modeling, spatial modeling and quality control and reliability data modeling. Professor Guo pays special attention to small sample analysis and DEAR (differential equation associated regression) modeling theory recently. He published more than 100 papers at international conferences and international journals.

Dr Danni Guo is a Specialist Scientist in the Global Change Research Group at the South African National Biodiversity Institute (SANBI). Her current research focus is on spatial data modeling and GIS in biodiversity under sparse data uncertainty.