Cui Yanhong

Guo Renkuan
University of Cape Town, Cape Town, South Africa

Guo Danni
South African National Biodiversity Institute, Cape Town, South Africa

Lambda algorithm and maximum likelihood estimation

Keywords

lambda algorithm, genetic algorithm, reliability, repairable system, likelihood-lambda procedure, reliability

Abstract

In this paper, a new global optimization algorithm by imitating ancient Chinese human body system model, named as lambda algorithm, is introduced. The lambda algorithm utilizes five-element multi-segment string to represent the *n*-dimensional Euclidean point and hence the string based operation rules for expansion, comparison and sorting candidate strings. The algorithm enjoys the simplest mathematical operations but generates highest searching speed and accuracy. We furthermore explore to merge the lambda algorithm with maximum likelihood procedure for creating a non-derivative scheme - likelihood- lambda procedure. A illustrative example is given.

1. Introduction

Safety and reliability optimization problems are a fundamental components intrinsically in the sense that the statistical theory underlying them is built up by a pile of relevant mathematical optimal theories and methodologies. Particularly, in safety and reliability modelling practices the maximum likelihood estimation plays an important role. Therefore, it is necessary to improve the efficiency in searching the optimal solution of a likelihood function.

Different searching schemes have different efficiency. The standard derivative-orient scheme, the Newton-Raphson procedure is the commonly engaged, see [1], [10], [11]. However, more and more searching schemes are utilizing non derivative-orient schemes, for example, merging likelihood and genetic algorithm to avoid derivative computations.

The lambda algorithm is created by imitating an ancient human body system, see [4], [6], [7]. In its searching scheme, except the necessary mathematical computations for evaluating the objective function and the creation of the initial "searching population" randomly, the algorithm only involves if-else logical operation and sort procedure. In contrast to existing global optimization algorithms, particularly GA, the lambda algorithm engages the simplest mathematics but reaches the highest searching efficiency.

The remaining structure of the paper is stated as following: Section two serves the introduction of 5-element string and the presentation of an Euclidean vector; Section three will features of the lambda algorithm; Section four will discuss the operators engaged in the lambda algorithm and identify the Markovian features of the searching scheme; Section five reserves for testing the new algorithm, Section six proposes the merging of lambda algorithm and likelihood searching procedure and a reliability application and Section Seven conclude this paper.

2. Five-element string and Euclidean vector presentation

In this section we utilize the Rosenbrock's function as an example for introducing the lambda algorithm.

Assume that the two-dimensional Rosenbrock's function, $f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$, which is the objective function under investigation. There are

many local optimal solutions, but we are interested in obtain the global optimal solution of *f*.

Similar to the genetic algorithm (GA) binary string representation, whose element set is Θ={0,1}, the λ-algorithm takes elements from a 5 element-set, which takes a membership set Θ={0, 1, 2, 3, 4}, to construct a string candidate solution which represents the candidate solution, Euclidean vector, $\underline{x}$, in the *2*-dimensional Euclidean space R^2 in term of a linear transformation.

Definition 2.1. A string in an algorithm, denoted by $\underline{e} = e_1 e_2 \cdots e_l$, is a sequence of l elements from the membership set Θ. The total number of the elements, l, composed of the string $\underline{e}$ is called the length of the string.

Definition 2.2. In a string algorithm, in order to represent an *n*-dimensional Euclidean point $\underline{x}' = (x_1, x_2, \cdots, x_n)$, a string $\underline{e}$ is typically constituted by *n* segmental strings, whose length are *u*, i.e., the string $\underline{e} = \underline{e}_1 \underline{e}_2 \cdots \underline{e}_n$ with length $l = nu$, where the i^{th} segment of the string, or the i^{th} segmental string, $\underline{e}_i = e_{i,u(i-1)+1} e_{i,u(i-1)+2} \cdots e_{i,iu}$, is of length u, (i=1,2,$\cdots$,n).

Definition 2.3. A triple (m,u,n) is called string configuration, where m = total number of elements in the membership set Θ, u = the length of segmental string $\underline{e}_i = e_{i,u(i-1)+1} e_{i,u(i-1)+2} \cdots e_{i,iu}$, and n = total number of segmental strings composing of the string $\underline{e} = \underline{e}_1 \underline{e}_2 \cdots \underline{e}_n$. Let $\mathrm{S} = \{\underline{e} : e_i \in \Theta\}$ denote the string space generated from *m*-element membership set Θ_m and $\mathrm{S}_{(m,u,n)} = \{\underline{e}_1 \underline{e}_2 \cdots \underline{e}_n : e_{(i,j_i)} \in \Theta_m\}$ the (m,u,n) configuration string space on Θ_m.

Example 2.4. A string with configuration triple (5,6,2) $\underline{e} = \underline{e}_1 \underline{e}_2$ represents a 2-dimensional candidate solution (X_1 , X_2) for Rosenbrock's function with two segmental strings, $\underline{e}_1$ and $\underline{e}_2$. For example,

1	3	4	1	4	0	0	1	2	2	3	0

The first segmental string, $\underline{e}_1$, is constituted by the first 6 elements, for example, $\underline{e}_1$ = 1 3 4 1 4 0, which represents the variable X_1, while the second segmental string, $\underline{e}_2$, is constituted by the seventh element to the twelfth element, i.e., the second six elements in the string, for example, $\underline{e}_2$ =0 1 2 2 3 0, which represents the variable X_2. The string length $l = 2 \times 6 = 12$. The correspondence between each segment string and X can be also labelled as following:

$$\underbrace{1\ 3\ 4\ 1\ 4\ 0}_{X_1}\quad \underbrace{0\ 1\ 2\ 2\ 3\ 0}_{X_2}$$

Let

$D_1 \equiv \left[u^1_{\min}, u^1_{\max}\right]$ be the searching domain of X_1;

$D_2 \equiv \left[u^2_{\min}, u^2_{\max}\right]$ be the searching domain of X_2;

$u^1_{\min}, u^1_{\max}$ be the lower bound and upper bound of X_1;

$u^2_{\min}, u^2_{\max}$ be the lower bound and upper bound of X_2.

Then, equation (1) and (2) specify the relationship between (X_1, X_2) and string $\underline{e} = \underline{e}_1 \underline{e}_2 = e_1 e_2 \cdots e_6 e_7 e_8 \cdots e_{12}$:

$$X_1 = u^1_{\min} + \left(u^1_{\max} - u^1_{\min}\right) \sum_{j=1}^{6} e_j \frac{5^{6-j}}{5^6} \tag{1}$$

$$X_2 = u^2_{\min} + \left(u^2_{\max} - u^2_{\min}\right) \sum_{j=7}^{12} e_j \frac{5^{12-j}}{5^{12}} \tag{2}$$

where e_j, $j = 1,2,\cdots,12$ denote the elements taking numbers 0, 1, 2, 3, 4 in the string.

Definition 2.5. A configuring (i, j_i) indicates the i^{th} segmental string and the j^{th} element (position) in the i^{th} segmental string.

Once we setup values of $D_1 \equiv \left[u^1_{\min}, u^1_{\max}\right]$ and $D_2 \equiv \left[u^2_{\min}, u^2_{\max}\right]$, the fitness value of a objective function $f(\underline{x})$ fitness value is readily to calculate according to equation (1) and (2).

A natural question will be raised: why does GA not offer the convergent power as high as the lambda algorithm does? In traditional binary strings represented algorithms, the chance of appearance of 0 or 1 element in the strings is 50% to 50% of each. The element repeated and unrepeated chances are equal, which let us feel difficult to draw any useful information from both repeated and unrepeated events respectively.

In the *Table 1*, we use configuration triple (2,6,2) to specify the string. In other words, (2,6,2) represents membership set Θ={0,1}, segmental string length 6, and 2 segmental strings.

Table 1. The binary element strings (2,6,2)

1	0	1	1	0	1	0	0	1	1	0	1
1	1	0	1	0	0	0	0	1	0	1	0

Similarly, in *Table 2*, (5,6,2) represents membership set Θ={0,1,2,3,4}, unit string length 6, and 2-segmental string.

Table 2. The five-element (5,6,2) strings

2	3	4	4	2	1	3	3	2	1	0	0
2	1	1	3	1	0	4	3	2	3	0	1
2	2	4	3	0	1	2	4	3	2	1	1
1	2	4	3	0	1	2	4	3	2	1	1
0	1	2	3	4	1	0	2	2	3	3	0

Assume that a column of strings is ranked according to their fitness values (from minimum to maximum), if we select top 5 ranked strings as a sample, see *Table 2*, the first knowledge we learn from it is that their fitness values are less than any other strings. On the other hand, in the lambda algorithm, the repeated chance of each element is 20%, see *Table 2*. That means that in the sample, if 3 out of 5 are repeated with a particular element at any position, then the repeated chance of the element in this position is 60%, which is much higher than 20% (in binary string cases). This phenomenon is the second knowledge we would like to know. Even we might interpret this phenomenon as a consequence of randomization. However, under a perfect circulation, we might consider the extra 40% chance would be induced by their smaller fitness values. This phenomenon shows us the convergent tendency toward the optimal solution is higher than that of GA. To highlight the global convergence tendency role in element numeration in the lambda algorithm, we divide element events into 3 categories: repeated one time element events, repeated two times element events and unrepeated element events. The lambda algorithm draws useful information from all three categories, to construct an intrinsic scheme towards global optimization.

3. Features of string representation

We should be aware that in searching and selecting candidate solution the λ-algorithm utilizes the information contained in the value of an element, the element position in the segmental string, and the sequential order of the segmental string. Note that the transformation matrix plays a vital role for linking a string of length nu, $\underline{e} = \underline{e}_1\underline{e}_2\cdots\underline{e}_n$, where the i^{th} piece of the string, or the i^{th} segmental string, with length u, $\underline{e}_i = e_{i,u(i-1)+1}e_{i,u(i-1)+2}\cdots e_{i,iu}$, ($i$=1,2,$\cdots$,n), and an n-dimensional Euclidean point $\underline{x}' = (x_1, x_2, \cdots, x_n)$. In other words, the global search strength mechanism 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 segmental 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 segmental string create the possibility that change in the element j_i (position) of the i^{th} segmental string will have different impacts because different position has different weight. A $(5,6,2)$ string, denoted by $e_1e_2\cdots e_6e_7,e_8\cdots e_{12}$, the blue-color elements are the first segmental string, representing x_1, the red-color elements are the second segmental string, representing x_2. Logically, changes in elements e_1 and e_7 will result in large 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 shift scheme will have a balanced global searching capability as well as local fine-tune capacity.

Example 3.1. Define $u_{\min}$ = - 10^{10} , $u_{\max}$ = + 10^{10} , then u_r = $u_{\max}$ - $u_{\min}$ = 2´ 10^{10}. String 1 in *Table 3*: 1 2 4 3 0 1 2 4 3 2 11 is the base for observing the impacts from string element changes. String 2 changes the first element of the first segmental string in String 1 by adding 1 and the first element of the second segmental string 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 first segmental string by adding 3 and the sixth element of the second segmental string 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 3* lists the changing effects for comparisons.

Table 3. 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 critical to emphasize here that the global feature of a string algorithm is also materialized by the range parameter setting, i.e., the setting of $(u_{\min}, u_{\max})$. The larger u_r = $u_{\max}$ - $u_{\min}$, the better global coverage is intended.

Examining the i^{th} component of $\underline{x}$, x_i, which is a real number, for example, $x_i = 7.5673$, traditional optimization schemes may be shift $x_i = 7.5673$ by a

small increment, say, 0.005, it is intuitive that by changing at digit level of a real, the scheme remains at a local algorithm nature. Therefore, the globally and locally simultaneous searching feature of the string scheme will definitely speed up the global optimization algorithm. By repeated and unrepeated (string) elements numerated from high weight position (i.e. element of a segmental string) to lower weight position within the strings (the candidate solution strings), in terms of sort and pack scheme repeatedly until the final optimized solution is found. Our computation experiences show that in searching the final global optimal solution, it is often the case that the first 3 or 4 positions (high weight elements) in a string (if the string size is 6) could adequately secure the main searching path toward the final optimal solution globally. Nevertheless, the lower cell elements are able to change for fine-tuning x_i in small increment manner, which allows the searching scheme to access the final optimal solution with amazingly high precision.

4. Operators in lambda algorithm

In this section we will systematically explore the mathematical operators in the lambda algorithm. For this purpose, it is necessary to introduce more notations.

4.1. String vector

A string vector, denoted by $\overline{e^1}$, is a column vector taking strings as its components. The dimensionality indicates how many strings are used to construct a string vector. It is obvious that for (5,6,2) configuration string vector, $\overline{e^1}$ of dimensionality 100, it represents as 100 pairs of variables X_1 and X_2. As matter of fact, a string vector $\overline{e^1} = (e_{ij})_{N\times l}$ is a matrix of elements $e_{ij} \in \Theta_5$ with size $N \times l$. *Figure 1* intuitively a string vector of $(e_{ij})_{100\times 12}$:

$$\overline{e^1} = \begin{bmatrix} \underline{e}_1^1 \\ \underline{e}_2^1 \\ \vdots \\ \underline{e}_{100}^1 \end{bmatrix} = \begin{bmatrix} e_{1,1} & \cdots & e_{1,6} & e_{1,7} & \cdots & e_{1,12} \\ e_{2,1} & \cdots & e_{2,6} & e_{2,7} & \cdots & e_{2,12} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e_{100,1} & \cdots & e_{100,6} & e_{11,7} & \cdots & e_{100,12} \end{bmatrix} \tag{3}$$

4.2. λ operator and λ^{-1} operator

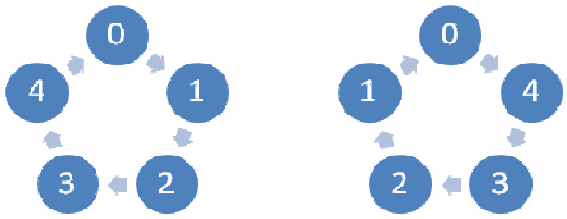


Figure 1. Cyclic behavior of operator $\lambda[e_i]$ (Left) and operator $\lambda^{-1}[e_i]$ (Right)

Definition 4.1. λ operator for (5,*u*,*n*) configuration string element is defined by

$$\lambda[e_i] = \begin{cases} e_i + 1 \ if \ \ e_i \in \{0,1,2,3\} \\ 0 \qquad if \ \ e_i \in \{4\} \end{cases} \tag{4}$$

where e_i is an element in a string $\underline{e}$.

Definition 4.2. λ^{-1} operator for (5,*u*,*n*) configuration string element is defined by

$$\lambda^{-1}[e_i] = \begin{cases} e_i - 1 \ if \ \ e_i \in \{1,2,3,4\} \\ 4 \qquad if \ \ e_i \in \{0\} \end{cases} \tag{5}$$

where e_i is an element in a string $\underline{e}$.

4.3. Cyclic behavior of a vector string

Definition 4.3. Let $\underline{e} = e_1 e_2 \cdots e_l$ be a (5,*u*,*n*) configuration string, the λ operation on string $\underline{e}$ is

$$\lambda[\underline{e}] \triangleq \lambda[e_1]\lambda[e_2]\cdots\lambda[e_l]. \tag{6}$$

Theorem 4.4. Let $\underline{e} = e_1 e_2 \cdots e_l$. be a $(5,u,l/u)$ configuration string of length $l = nu$, where *u* is the length of segmental string. Then

$$\lambda^{(n)}[\underline{e}] = \lambda^{(\mathrm{mod}_4(n))}[\underline{e}], \tag{7}$$

where

$$\lambda^{(n)}[\underline{e}] \triangleq \lambda\Big[\underset{\text{repeat } n \text{ times}}{\lambda[\cdots\lambda[\underline{e}]]}\Big], \qquad \lambda^{(0)}[\underline{e}] \equiv \underline{e}. \tag{8}$$

(Note that $\mathrm{mod}_4(\cdot)$ is a modulo operator using 4 as its quotient.)

Proof 4.5. According to Definition 4.1, applying λ operator to a string $\underline{e}$ is just applying the operator λ to each individual element in string $\underline{e}$, e_i, without any disturbance on the sequential order, *i*. Also, recall that $\lambda^{(n)}[e_i]$ is cyclic as *n* increases by step size

1, e.g., $e_i = 0$, $\lambda^{(5)}[e_i] = \lambda^{(0)}[e_i] = e_i = 0$. Hence, $\lambda^{(n)}[\underline{e}] = \lambda^{(\mathrm{mod}_4(n))}[\underline{e}]$.

Corollary 4.6. Let $\overline{e^1}$ be a string vector. Then

$$\lambda^{(n)}\left[\overline{e^1}\right] = \lambda^{(\mathrm{mod}_4(n))}\left[\overline{e^1}\right], \quad \lambda^{(0)}\left[\overline{e^1}\right] \equiv \overline{e^1}. \tag{9}$$

Remark 4.7. Define $\overline{e^1_{(0)}} \square \overline{e^1}$, $\overline{e^1_{(1)}} \square \lambda\left[\overline{e^1}\right] = \lambda\left[\overline{e^1_{(0)}}\right]$, $\overline{e^1_{(2)}} = \lambda\left[\lambda\left[\overline{e^1}\right]\right] = \lambda^{(2)}\left[\overline{e^1}\right]$, $\overline{e^1_{(3)}} = \lambda^{(3)}\left[\overline{e^1}\right]$, and $\overline{e^1_{(4)}} = \lambda^{(4)}\left[\overline{e^1}\right]$. We treat the string vectors within one cycle as the 5 states of $\overline{e^1}$: { $\overline{e^1_{(0)}}$, $\overline{e^1_{(1)}}$, $\overline{e^1_{(2)}}$, $\overline{e^1_{(3)}}$, $\overline{e^1_{(4)}}$ }, see *Figure 2.* Then if we prepare *M* string vectors, then in terms of λ operator, we will immediately expand to 5*M* string vectors. In such a sense, we call it as λ expansion operation on string vectors.

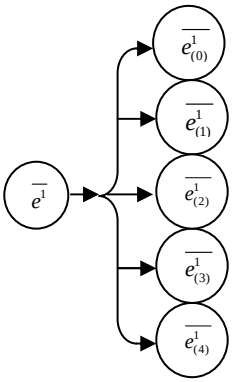


Figure 2. A string vector $\overline{e^1}$ and its cyclic vectors

Example 4.8. Given a string $\underline{e^1_1}$ in strings vector $\overline{e^1}$:

1	3	4	1	4	0	0	1	2	2	3	0

The string $\underline{e^1_1}$ will be expanded by applying λ , the cyclic vectors with respect to $\underline{e^1_1}$ are : String $\underline{e^1_{1(0)}}$ of (0) state in the string vector $\overline{e^1_{(0)}}$,

1	3	4	1	4	0	0	1	2	2	3	0

String $\underline{e^1_{1(1)}}$ of (1) state in string vector $\overline{e^1_{(1)}}$,

2	4	0	2	0	1	1	2	3	3	4	1

String $\underline{e^1_{1(2)}}$ of (2) state in the string vector $\overline{e^1_{(2)}}$,

3	0	1	3	1	2	2	3	4	4	0	2

String $\underline{e^1_{1(3)}}$ of (3) state in the string vector $\overline{e^1_{(3)}}$,

4	1	2	4	2	3	3	4	0	0	1	3

String $\underline{e^1_{1(4)}}$ of (4) state in the string vector $\overline{e^1_{(4)}}$,

0	2	3	0	3	4	4	0	1	1	2	4

All the strings generated will be in the same row position in relevant string vectors.

4.4 . λ comparison operation

In the candidate solution search procedure, we need to compare the objective function $f(\underline{x}), \underline{x} \in \mathrm{R}^n$ at different values of candidates $\underline{x}$ and thus it is inevitably to compare strings, at which *f* will be optimal.

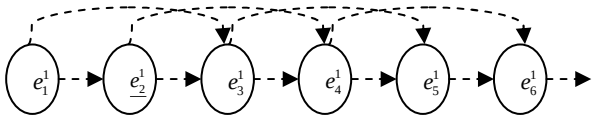


Figure 3. λ comparison operation of string vector $\overline{e^1}$

There are two kinds of λ comparison operations, but we only engage the first kind of comparison operation within a strings vector.

Definition 4.9. λ comparison operation of the first kind means that the value of the element e_i follows $0 \to 1 \to 2 \to 3 \to 4$ criterion to change.

Example 4.10. Let $\underline{e^1_1}, \underline{e^1_2}, \cdots\cdots, \underline{e^1_{100}}$ are strings in string vector $\overline{e^1}$. Let $e^1_{n(1)}, e^1_{n(2)}, \cdots\cdots, e^1_{n(12)}$ are elements in any string $\underline{e^1_n}$, respectively, $n = 1, 2, \cdots, 100$. Then

For n=1:1:100-2
For i=1:1:12
If $e^1_{n(i)} \equiv e^1_{n(i+1)}$
$e^1_{n(i+1)} \leftarrow \lambda\left[e^1_{n(i)}\right]$
ELSE IF $e^1_{n(i)} \equiv e^1_{n(i+2)}$
$e^1_{n(i+2)} \leftarrow \lambda\left[e^1_{n(i)}\right]$
END

Definition 4.11. λ comparison operation of the second kind means that the value of the element e_i follows $0 \to 4 \to 3 \to 2 \to 1$ criterion to change.

Example 4.12. Let, $\underline{e^1_1}, \underline{e^1_2}, \cdots\cdots, \underline{e^1_{100}}$ are strings in string vector $\overline{e^1}$. Let $e^1_{n(1)}, e^1_{n(2)}, \cdots\cdots, e^1_{n(12)}$ are elements in any string $\underline{e^1_n}$, respectively, $n = 1, 2, \cdots, 100$. Then

For n=1:1:100-2
For i=1:1:12
If $e^1_{n(i)} \equiv e^1_{n(i+1)}$
$e^1_{n(i+1)} \leftarrow \lambda^{-1}\left[e^1_{n(i)}\right]$
ELSE IF $e^1_{n(i)} \equiv e^1_{n(i+2)}$
$e^1_{n(i+2)} \leftarrow \lambda^{-1}\left[e^1_{n(i)}\right]$
END

Furthermore, we state the assumption on the initial set of string vectors.

Initialization Assumption: Let the initial string vector be $\overline{e^0} = \left(e_{ij}^{(0)}\right)_{M\times l}$ such all the elements in the i^{th} string $\underline{e}_i$ are mutually independent, $i = 1, 2, \cdots, M$.

Let $D \equiv [u_{\min}, u_{\max}]^n$ be the searching domain for an objective function $f(\underline{x})$ defined in n-dimensional Euclidean space $\square^n$. It is obvious D determines the scope of searching globally. Mathematically, the linear system linking the strings $\underline{e}$ and the system state $\underline{x}$ can be expressed by

$$\begin{cases} x_1 = u_{\min} + (u_{\max} - u_{\min}) \sum_{j=1}^{u} e_j \dfrac{5^{u-j}}{5^u} \\ x_2 = u_{\min} + (u_{\max} - u_{\min}) \sum_{j=u+1}^{2u} e_j \dfrac{5^{2u-j}}{5^{2u}} \\ \vdots \\ x_n = u_{\min} + (u_{\max} - u_{\min}) \sum_{j=(n-1)u+1}^{nu} e_j \dfrac{5^{nu-j}}{5^{nu}} \end{cases} \tag{10}$$

Let the weight matrix be

$$O_{n\times nu} = \begin{bmatrix} \frac{5^{u-1}}{5^u} & \cdots & \frac{5^0}{5^u} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{5^{u-1}}{5^u} & \cdots & \frac{5^0}{5^u} & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{5^{u-1}}{5^u} & \cdots & \frac{5^0}{5^u} \end{bmatrix} \tag{11}$$

and further, let

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \ \underline{u}_{\min} = \begin{bmatrix} u_{\min} \\ u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}; \ u_r = u_{\max} - u_{\min} \tag{12}$$

and write string $\underline{e} = e_1 e_2 \cdots e_l$ in column vector form, (i.e., $nu\times 1$ column vector of element e_i), that is

$$\underline{e}'_{nu\times 1} = \begin{bmatrix} \underline{e}'_1 \\ \vdots \\ \underline{e}'_u \\ \underline{e}'_{u+1} \\ \vdots \\ \underline{e}'_{2u} \\ \vdots \\ \underline{e}'_{nu-1} \\ \vdots \\ \underline{e}'_{nu} \end{bmatrix} \tag{13}$$

Then the candidate solution is a linear transformation of the (5,*u*,*nu*) configured string representation

$$\underline{x} = \underline{u}_{\min} + u_r O_{n\times nu} \underline{e}'_{nu} \tag{14}$$

Definition 4.13. Let $\underline{x}^{(q)} = \underline{u}_{\min} + u_r \lambda^{(q)}[\underline{e}]O', q \in \{0,1,2,3,4\}$, then,

$$\left\{ f\left(\underline{x}^{(0)}\right), f\left(\underline{x}^{(1)}\right), f\left(\underline{x}^{(2)}\right), f\left(\underline{x}^{(3)}\right), f\left(\underline{x}^{(4)}\right) \right\}_{\underline{e}} \tag{15}$$

is called the cyclic set of objective function values with respect to string $\underline{e}'_{nu\times 1}$.

Definition 4.14. (λ comparison operation on two strings) If we compare 2 strings $\underline{e}_1$ and $\underline{e}_2$. Assume that String $\underline{e}_1$ (candidate solution)'s fitness value is better than string $\underline{e}_2$'s, and then we managed some change to $\underline{e}_2$. Let L be the length of strings $\underline{e}_1$, $\underline{e}_2$. Let e_{1i} and e_{2i} be one of the element of $\underline{e}_1$ and $\underline{e}_2$ respectively, and $g_1(\cdot)$ and $g_2(\cdot)$ be 2 different kind of λ comparison operation functions.

$$g(e_{2i}) = \begin{cases} \lambda[e_{2i}], & if \ e_{1i} = e_{2i} \\ e_{2i}, & if \ e_{1i} \neq e_{2i} \end{cases} \tag{16}$$

$$g_2(e_{2i}) = \begin{cases} \lambda^{-1}[e_{2i}], & if \ e_{1i} = e_{2i} \\ e_{2i}, & if \ e_{1i} \neq e_{2i} \end{cases} \tag{17}$$

Because λ comparison operation have 2 different kind of functions, and each function apply on only one single string vector $\bar{e}$, so we may only describes $g(\cdot)$'s applications as a sample for other similar function.

In the definition 4.14, we only describe one kind λ comparison function $g(\cdot)$. The other one function is similar to $g(\cdot)$ operation in string vector $\bar{e}$.

Definition 4.15. Given a string vector $\underline{e_{vu\times N}}$. A string $\underline{e} = e_1 e_2 \cdots e_u e_{u+1} \cdots e_{2u} \cdots e_{(v-1)u+1} \cdots e_{vu}$ represent a candidate solution has *n* components. By ranking of the fitness values from best to worst, we have a sorted string vector $\overline{e'}$, where e_{ni} denotes any element in $\overline{e'}$ at n^{th} row, l^{th} column. Then λ Comparison operation in strings vector defined as:

If $n \geq 3$

$$g(e_{nl}) = \begin{cases} \lambda[e_{nl}], & if \ e_{nl} = e_{(n-1)l} \neq e_{(n-2)l} \\ \lambda[e_{nl}], & if \ e_{nl} \neq e_{(n-1)l} = e_{(n-2)l} \\ \lambda^{(2)}[e_{nl}], & if \ e_{nl} = e_{(n-1)l} = e_{(n-2)l} \\ e_{nl}, & if \ e_{nl} \neq e_{(n-1)l} \neq e_{(n-2)l} \end{cases} \tag{18}$$

where

$$\begin{aligned} p(\lambda[e_{nl}]) &= p\left(e_{nl} = e_{(n-1)l} \neq e_{(n-2)l}\right) + p\left(e_{nl} \neq e_{(n-1)l} = e_{(n-2)l}\right) \\ &= 0.16 + 0.16 = 0.32; \end{aligned} \tag{19}$$

$p\left(\lambda^{(2)}[e_{nl}]\right)=0.04, p(e_{nl})=0.64$

If $n=2$

$$g(e_{nl})=\begin{cases}\lambda[e_{nl}], & if\ \mathrm{e}_{nl}=e_{(n-1)l}\\ e_{nl}, & if\ \mathrm{e}_{nl}\neq e_{(n-1)l}\end{cases} \tag{20}$$

where

$$p\left(\lambda[e_{nl}]\right)=0.2 \text{ and } p(e_{nl})=0.8 \tag{21}$$

If $n = 1$

$$g(e_{nl})=e_{nl} \tag{22}$$

Note that at each looping time t , a λ comparison operation on whole string vector $\bar{e}$ will result in a new conditional random variable. If denote it as $\overline{e_t}$, $t=0,2,\cdots,n^0$, then $\left\{\overline{e_t}, t=0,2,\cdots,n^0\right\}$ is a stochastic process and furthermore it is a Markov Markov (decision) process due to the independent Initialization Assumption. Because the decision for choosing actions (λ comparison operation) does not only depend on the present state but also concerning prior states, so the process is not a simple Markov decision process, but more complicated.

4.5. λ expansion operation

Definition 4.16. Given a string $\underline{e}'_{nu\times 1}$ in column vector form, then the set of strings after an expansion

$$\lambda^{\text{expansion}}\left[\underline{e}'\right]\square\left\{\lambda^{(0)}\left[\underline{e}'\right],\lambda^{(1)}\left[\underline{e}'\right],\lambda^{(2)}\left[\underline{e}'\right],\lambda^{(3)}\left[\underline{e}'\right],\lambda^{(4)}\left[\underline{e}'\right]\right\} \tag{23}$$

is called the λ expansion set.

Now we would like to examine the string state change in λ expansion set after a λ comparison operation executed in string vector.

If $n\geq 3$

$$\lambda^{\text{expansion}}g(e_{nl})=\begin{cases}\lambda^{(k)}[\underline{e}_n]\lambda[e_{nl}], & if\ e_{nl}=e_{(n-1)l}\neq e_{(n-2)l}\\ \lambda^{(k)}[\underline{e}_n]\lambda[e_{nl}], & if\ e_{nl}\neq e_{(n-1)l}=e_{(n-2)l}\\ \lambda^{(k)}[\underline{e}_n]\lambda^{(2)}[e_{nl}], & if\ e_{nl}=e_{(n-1)l}=e_{(n-2)l}\\ \lambda^{(k)}[\underline{e}_n], & if\ e_{nl}\neq e_{(n-1)l}\neq e_{(n-2)l}\end{cases} \tag{24}$$

If $n=2$

$$\lambda^{\text{expansion}}\left(g(e_{nl})\right)=\begin{cases}\lambda^{(k)}[\underline{e}_{nl}]\lambda[\underline{e}_{nl}], & if\ \mathrm{e}_{nl}=e_{(n-1)l}\\ \lambda^{(k)}[\underline{e}_{nl}], & if\ \mathrm{e}_{nl}\neq e_{(n-1)l}\end{cases} \tag{25}$$

If $n = 1$

$$\lambda^{\text{expansion}}\left(g(e_{nl})\right)=\lambda^{(k)}[\underline{e}_n] \tag{26}$$

where $k=0,1,2,3,4$; $n=1,2,\cdots N$; $l=1,2,\cdots L$. L is the length of string, N represent size of strings in string vector.

Both λ expansion and λ comparison operations in string vector are taken after ranking the string vector according to the value of objective function $f(\underline{x})$. After ranking, the fitness values corresponding to strings e_n, e_{n-1}, e_{n-2} are supposed to be very close to those corresponding to whole vector strings.

Therefore, what we need to find out are whether or not some same elements exist in each of $\underline{e}_n, \underline{e}_{n-1}, \underline{e}_{n-2}$ (three strings) to ensure those repeated elements in the strings are the reason why the fitness values are similar. According to equation (20), we can see the repeated elements e_{nl} already separated from the unrepeated elements, by taken an extra $\lambda[\cdot]$ operation, the twice time repeated elements also separated from the unrepeated elements by taken two times $\lambda[\cdot]$ operation. Then one time and two times repeated elements rejoin with other elements in $\lambda^{(k)}[\underline{e}_n]$, *k=0, 1,2,3,4* respectively.

Consequently, we can select only one string from rejoined 5 states $\lambda^{(k)}\left[\underline{e}_n^{new}\right]$, *k=0,1,2,3,4* of strings.

After carrying on the above process recursively, the sequence of the fitness values of objective function will be convergent. The recursive procedure is shown in *Figure 4*, which demonstrates a dynamic Bayesian network pattern.

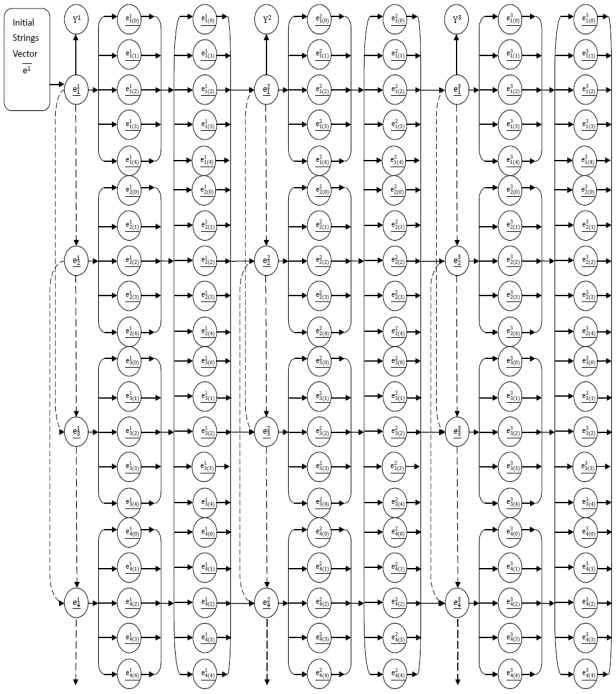


Figure 4. A dynamic Bayesian networks (DBNs) representation of λ algorithm

Figure 5 gives the flow chart to express the operations process of λ algorithm.

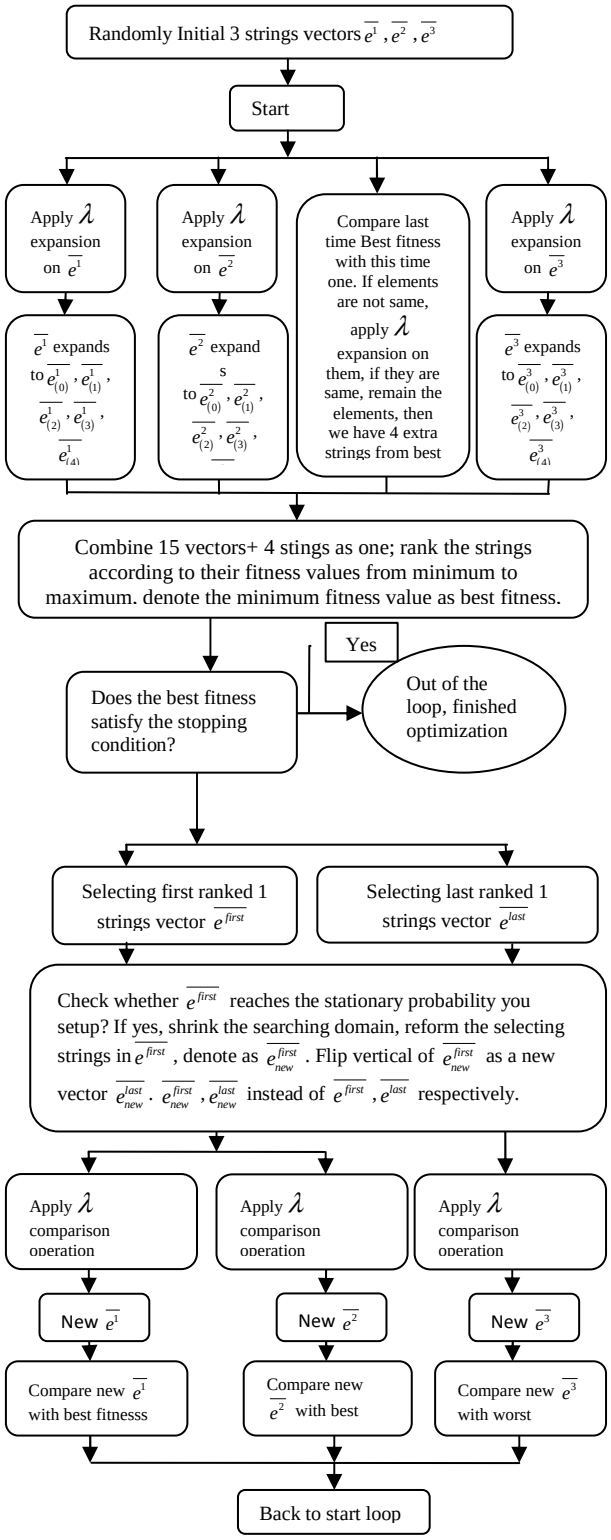


Figure 5. Flow chart to express the operations process of λ algorithm

Bayesian networks (BNs) is a probabilistic graphical model (GM), where an individual node in the GM represents a random variable, while those edges between the nodes represent the conditional probabilities among the corresponding random variables. A GM enjoys certain degree of Markov property. See [2], [3], [8].

A dynamic Bayesian networks (DBNs) is a general state-space model as an extension of Kalman Filter Models and Hidden Markov model. General speaking, a state-space model first specifies a prior $p(X_0)$ and a state-transition function, $p(X_t \mid X_{t-1})$, and an observation function, $p(Y_t \mid X_{t-1})$

It is critical that the observations are conditional first-order Markov $p(Y_t \mid X_t, Y_{t-1}) = p(Y_t \mid X_t)$. The Markovian character of DBNs essentially guarantees the existence of the stationary probability of the steady state.

It is fundamental to recognize that λ algorithm engages a mechanism of the DBNs. Such a recognition drove out the long-time bothering issue, why a λ algorithm converges almost sure and the global optimization can be achieved.

5. Testing examples

As a conventional step to bring in a new global optimization algorithm, we utilize the new algorithm to search the optima of four 30-dimensional testing functions and three 10-dimensional test functions. In addition, we use two extreme challengeable testing functions. The string configuration for the lambda algorithm engaged for the first three testing is (5,4,120), but for Levy function is (5,3,90).

Table 4 lists conventional test indices for the four 30-dimensional test functions.

Table 4. Algorithm efficiency indices

Search indices	Ackley	Dixon & Price	Griewank	Levy
Domain	$[-15,30]^{30}$	$[-10,10]^{30}$	$[-600,600]^{30}$	$[-10,10]^{30}$
Time (sec.)	180.32	167.12	74.41	104.94
Loop	144	289	125	187
Probab. control	0.9	0.8	0.8	0.8

As to the function specifications and searched optima for the four 30-dimensional test functions, we list them as following:

1. Ackley function: Number of variables: n = 30. The minimum is 0 when $x_i = 0,\ \ i = 1,\cdots,30$. Ackley function in general takes the form:

$$f(x_1,\cdots,x_n) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \qquad (27)$$

The searched minimum =7.9936E-015;

Optimal $\underline{x}^o$ =1.0E-014 × (0.0391 0.0857 0.1666 0.1666 0.1767 -0.1808 0.1631 0.1768 0.1790 -0.5370 0.2282 0.4305 -0.6646 0.1693 0.1778 -0.1060 0.0743 0.1775 0.0127 0.1605 -0.0093

0.1357 0.0338 0.1007 0.0285 -0.3655 -0.0357 -0.3171 -0.1896 0.0224)

2. Dixon and Price Function: Number of variables: n = 30, the minimum is 0 when $x_i = 0,\ i = 1,\cdots,30$. The function is defined by

$$f(x_1,\cdots,x_n) = (x_1 - 1)^2 + \sum_{i=2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2 \qquad (28)$$

Searched minimum =0.7463, optimal solution is

$\underline{x}^o$ =(0.2399 0.0866 -0.0012 -0.0004 -0.0050 -0.0004 -0.0026 -0.0046 -0.0116 -0.0029 0.0030 0.0023 0.0055 -0.0011 -0.0014 0.0005 0.0000 0.0017 0.0225 -0.0161 0.0004 -0.0011 -0.0002 0.0289 0.0161 -0.0001 -0.0009 -0.0005 -0.0023 0.0003)

3. Griewank Function: Number of variables: n =30. The minimum is 0 when $x_i = 0$. The n -dimensional Griewank function takes the form:

$$f(x_1,\cdots,x_n) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(x_i/\sqrt{i}\right) + 1 \qquad (29)$$

Searched minimum =0

Optimal $\underline{x}^o$ =1.0E-007 *(0.0007 -0.0075 -0.0229 0.0295 0.0282 0.0106 -0.0046 0.0347 -0.0426 -0.0098 -0.0229 0.0636 -0.0007 -0.0098 0.0374 -0.0033 0.0111 0.0754 -0.0033 0.0164 0.0004 0.1540 0.2458 -0.0885 0.0360 0.1475 0.0020 -0.3008 0.0492 0.0557)

4. Levy function: Number of variables: n =30. The minimum is 0 when $x_i = 1$

$$\begin{aligned} f(x_1,\cdots,x_n) = &\sum_{i=0}^{n-2} (y_i - 1)^2 \left(1 + 10\sin^2(\pi y_i + 1)\right) \\ &+ \sin^2(\pi y_0) + + (y_{n-1} - 1)^2 \left(1 + \sin^2(2\pi x_{n-1})\right) \end{aligned} \qquad (30)$$

where

$$y_i = 1 + \frac{x_i - 1}{4},\ i = 1,\cdots,n \qquad (31)$$

Searched minimum =0.5840, optimal solution is

$\underline{x}^o$ = (1.0166 0.9980 0.9982 0.9919 0.9978 0.2377 0.3999 0.9944 1.0103 0.9965 0.9992 1.0105 0.9773 1.0031 0.3994 1.0407 1.0140 -0.0792 1.0009 0.9985 0.9957 1.0060 1.0487 0.9937 0.3999 1.0037 0.9966 0.3933 0.3999 1.0108)

Table 5 summarizes three 10-dimensional test functions. The search scheme utilizes (6,4,40) string configuration.

Table 5. Algorithm efficiency indices

Search indices	Michalewics	Rastrigin	Rosenbrock
Domain	$[0,\pi]^{10}$	$[-5,5]^{10}$	$[-5,5]^{10}$
Time (sec.)	90.66	37.82	24.47
Loop	200	174	100
Probability control	0.8	0.4	0.98

5. Michalewics Function: Number of Variables: n = 10. The theoretical minimum value is -9.66015

$$f(x_1,\cdots,x_n) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{20} \qquad (32)$$

The searched minimum value = - 9.2562, the optimal value is

$\underline{x}^o$ = (2.1987 1.5692 2.2179 1.9225 0.9947 1.5733 1.4516 1.7603 1.6588 1.2171)

6. Rastrigin Function: Number of Variables: n =10

The theoretical minimum value is 0 when $x_i = 0$, $i = 1,\cdots,10$.

$$f(x_1,\cdots,x_n) = -10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right) \qquad (33)$$

The searched minimum =0, the optimal solution is

$\underline{x}^o$ =1.0E-008 × (0.4096 -0.0819 -0.0819 0.2458 0.2458 -0.2130 0.1147 -0.1802 -0.4096 0.2458)

7. Rosenbrock Function: Number of Variables: n = 10. The general form of Rosenbrock Function is

$$f(x_1,\cdots,x_n) = \sum_{i=1}^{n-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right] \qquad (34)$$

The theoretical minimum is 0, when $x_i = 1$, $i = 1,\cdots,10$. The searched minimum = 0.000194808, the optimal

$\underline{x}^o$ = (0.9998 0.99969 0.99974 0.99933 0.99928 0.9991 0.998 0.99572 0.99137 0.98285)

The last two testing functions are extremely challengeable. *Table 6* summarizes the two testing functions testing results. The lambda algorithm searching scheme utilizes (5,4,120) and (5, 4, 400) respectively.

Table 6. Algorithm efficiency indices for 30-dimensional Rosenbrock function and 100-dimensional $\sin^{20}$ function

Search indices	Rosenbrock	$\sin^{20}$
Domain	$[-2.408, 2.408]^{30}$	$[-10,10]^{100}$
Time (sec.)	90.95	291.06

Loop	100	100
Probability control	0.60	0.3

8. Rosenbrock Function: Number of Variables: n =30, the minimum is 0 when $x_i = 1$, $i = 1, \cdots, 30$. The string configuration for searching lambda scheme is (5,4,120).
The searched minimum = 2.5183

Optimal $\underline{x}^o$ =(0.9994 0.9973 0.9994 1.0024 1.0014 1.0003 1.0035 1.0007 1.0015 1.0007 1.0004 1.0015 0.9971 0.9966 1.0001 0.9995 0.9973 0.9997 0.9952 0.9991 0.9927 0.9856 0.9707 0.9473 0.9014 0.9520 0.9744 0.9714 0.9491 0.9008)

9. $\sin^{20}$ function: Number of Variables: n =100. The theoretical maximum is n when $x_i = j\pi + \pi/2$, $j = 0, 1, 2, \cdots$.

$$f(x_1, \cdots, x_n) = \sum_{i=1}^{n} \sin^{20}(x_i) \tag{35}$$

The searched maximum value = 91.0671, the optimal

$\underline{x}^o$ =(1.4857 -7.8411 -4.6816 -4.7119 -1.5208 -4.6401 1.5676 7.8522 1.6444 7.8734 -7.8784 4.7178 -1.5417 -7.8683 1.5590 7.7531 7.8484 -1.5301 -4.7494 -4.8971 1.5430 -4.7074 -1.3356 -1.5826 4.7689 2.9632 -1.5465 1.6441 -1.5685 1.4859 1.5135 1.5899 1.5212 0.1591 -0.9106 1.5329 -4.7451 4.6183 4.7783 7.8759 1.5594 1.5520 -4.8942 -1.5857 -1.5533 4.6714 1.5529 1.6801 1.5276 1.5989 -7.8975 -7.8473 -4.7436 -1.5183 4.7857 7.8521 4.7119 -1.5925 -1.6175 -1.6763 4.7724 -4.7648 -4.6491 1.5808 -7.8505 -7.8527 4.6484 3.7575 4.7969 -1.5098 1.4933 7.8250 1.7861 -1.4387 -1.5806 7.8622 -1.4867 -4.6982 -4.3371 1.5879 1.7464 -1.4736 -7.8447 1.5872 1.5873 -1.5733 -7.9322 4.7021 1.5871 1.5891 -4.7056 1.6080 4.8077 -1.5593 1.5689 -4.7446 -1.5889 -1.4475 -4.7321 1.7987)

In summary, the algorithm testing demonstrates satisfactory result in accuracy and efficiency.

6. Likelihood-lambda procedure

Likelihood function and procedure plays important role in safety and reliability modelling, see [1], [10], and [11]. In this section, we will investigate the scheme to utilize the lambda algorithm for searching the numerical solution to a likelihood function.

6.1. Log-likelihood function

Let $L(\underline{\theta} \mid \mathrm{K}) = f(x_1, \vartheta_1, x_2, \vartheta_2, \cdots, x_N, \vartheta_N; \underline{\theta})$ with $f(\cdot; \underline{\theta})$ representing the joint distribution of data K. This is then called the likelihood function with respect to parameter set $\underline{\theta}$, $\underline{\theta} \in \Theta$.

Definition 6.1. Let $\mathrm{K} = \{(x_i, \vartheta_i), i = 1, 2, \cdots, N\}$ be a failure-censoring data record, i.e.,

$$\vartheta_i = \begin{cases} 0 & x_i \text{ is a natural failure} \\ 1 & x_i \text{ is a censored event} \end{cases} \tag{36}$$

then

$$L(\underline{\theta} \mid \mathrm{K}) = \prod_{i=1}^{N} \left(f^{1-\vartheta_i}(x_i; \underline{\theta}) R^{\vartheta_i}(x_i; \underline{\theta}) \right) \tag{37}$$

where f is the failure density function and R is the reliability function.

Definition 6.2. The function then

$$l(\underline{\theta} \mid \mathrm{K}) = \ln\left(L(\underline{\theta} \mid \mathrm{K})\right) \tag{38}$$

is called the log-likelihood function.

Lemma 6.3. $\underline{\theta}_0$ is an optimal point for $l(\underline{\theta} \mid \mathrm{K})$ if and only if it is an optimal point for $L(\underline{\theta} \mid \mathrm{K})$.

Note that $\ln(\cdot)$, whose base is $e > 1$, is monotone increasing. Therefore the patterns in $L(\underline{\theta} \mid \mathrm{K})$ will be well-maintained by $l(\underline{\theta} \mid \mathrm{K})$ and the converse is also true: then

$$\begin{aligned} & l(\underline{\theta}_0 \mid \mathrm{K}) = \max\{l(\underline{\theta} \mid \mathrm{K})\} \\ & \Leftrightarrow \\ & L(\underline{\theta}_0 \mid \mathrm{K}) = \max\{L(\underline{\theta} \mid \mathrm{K})\} \end{aligned} \tag{39}$$

Turning our attention now to wave-like lifetime distribution of Type I, (see [7], [8]), it has a form:

$$F(x) = 1 - \exp\left(-\int_0^x \left(\gamma + \frac{\sin^2 \alpha s}{s^2} \right) ds \right) \tag{40}$$

with two-parameter hazard function:

$$\begin{aligned} & h(x) = \gamma + \frac{\sin^2 \alpha x}{x^2} \\ & x \in [0, +\infty), \ \alpha > 0, \ \gamma \geq 0 \end{aligned} \tag{41}$$

Theorem 6.4. ([7], [8]) For the Type I wave-like distribution, the log-likelihood function is:

$$\begin{aligned} l(\alpha, \gamma \mid \mathrm{K}) = & \sum_{i=1}^{N} (1 - \vartheta_i) \ln\left(\gamma + \frac{\sin^2 \alpha x_i}{x_i^2} \right) \\ & - \sum_{i=1}^{N} \int_0^{x_i} \left(\gamma + \frac{\sin^2 \alpha s}{s^2} \right) ds \end{aligned} \tag{42}$$

The first-order partial derivatives are

$$\frac{\partial l(\alpha,\gamma|\mathrm{K})}{\partial\alpha}=\sum_{i=1}^{N}\frac{\sin(2\alpha x_i)(1-\vartheta_i)x_i}{\gamma x_i^2+\sin^2\alpha x_i}-\sum_{i=1}^{N}\int_0^{x_i}\frac{\sin(2\alpha s)}{s}ds$$
$$\frac{\partial l(\alpha,\gamma|\mathrm{K})}{\partial\gamma}=\sum_{i=1}^{N}\frac{x_i^2(1-\vartheta_i)}{\gamma x_i^2+\sin^2\alpha x_i}-\sum_{i=1}^{N}x_i \quad (43)$$

and the second-order order partial derivatives are

$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial^2\alpha}=\sum_{i=1}^{N}\frac{2\cos(2\alpha x_i)\left(\gamma x_i^2+\sin^2\alpha x_i\right)-\sin^2(2\alpha x_i)}{\left(\gamma x_i^2+\sin^2\alpha x_i\right)^2}(1-\vartheta_i)x_i^2$$
$$-\frac{1}{\alpha}\sum_{i=1}^{N}\sin(2\alpha x_i)$$
$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial\alpha\partial\gamma}=-\sum_{i=1}^{N}\frac{x_i^3(1-\vartheta_i)\sin(2\alpha x_i)}{\left(\gamma x_i^2+\sin^2\alpha x_i\right)^2} \quad (44)$$
$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial\gamma^2}=-\sum_{i=1}^{N}\frac{x_i^4(1-\vartheta_i)}{\left(\gamma x_i^2+\sin^2\alpha x_i\right)^2}$$

Theorem 6.5. For the Type II wave-like lifetime distribution with 2 parameters, and a hazard function of the form $h(x)=\gamma+\sin(\alpha x)/x$, the log-likelihood function in the presence of both failures and censored data is

$$l(\alpha,\gamma|\mathrm{K})=\sum_{i=1}^{N}(1-\vartheta_i)\ln\left(\gamma+\frac{\sin(\alpha x_i)}{x_i}\right)$$
$$-\gamma\sum_{i=1}^{N}x_i-\sum_{i=1}^{N}\int_0^{x_i}\frac{\sin(\alpha s)}{s}ds \quad (45)$$

The first-order partial derivatives are

$$\frac{\partial l(\alpha,\gamma|\mathrm{K})}{\partial\alpha}=\sum_{i=1}^{N}(1-\vartheta_i)\frac{x_i\cos(\alpha x_i)}{\gamma x_i+\sin(\alpha x_i)}$$
$$+\frac{1}{\alpha}\sum_{i=1}^{N}\sin(\alpha x_i) \quad (46)$$
$$\frac{\partial l(\alpha,\gamma|\mathrm{K})}{\partial\gamma}=\sum_{i=1}^{N}(1-\vartheta_i)\frac{x_i}{\gamma x_i+\sin(\alpha x_i)}-\sum_{i=1}^{N}x_i$$

and the second-order partial derivatives are

$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial\alpha^2}=-\sum_{i=1}^{N}(1-\vartheta_i)x_i^2\frac{\gamma x_i\sin(\alpha x_i)+1}{\left(\gamma x_i+\sin(\alpha x_i)\right)^2}$$
$$-\frac{1}{\alpha^2}\sum_{i=1}^{N}\sin(\alpha x_i)+\frac{1}{\alpha}\sum_{i=1}^{N}x_i\cos(\alpha x_i)$$
$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial\alpha\partial\gamma}=-\sum_{i=1}^{N}\frac{(1-\vartheta_i)x_i^2\cos(\alpha x_i)}{\left(\gamma x_i+\sin(\alpha x_i)\right)^2} \quad (47)$$
$$\frac{\partial^2 l(\alpha,\gamma|\mathrm{K})}{\partial\gamma^2}=-\sum_{i=1}^{N}\frac{(1-\vartheta_i)x_i^2}{\left(\gamma x_i+\sin(\alpha x_i)\right)^2}$$

Remark 6.6. Theorem 6.4 and 6.5 facilitate classical maximum likelihood estimation with derivatives up to the second order for the two types of wave-like lifetime distributions. Reliability engineers can use these two theorems for modeling and analysis in traditional Newton-Raphson procedure or use semi-derivative or non-derivative Likelihood-GA procedure if they do not mind the computation time consumptions. To reach a better efficiency, we intend to switch our attention to replacing the GA part by lambda algorithm.

6.2. An likelihood-lambda algorithm example

The ML-lambda procedure for searching solutions to the joint non-linear equation system:

$$\begin{cases}\partial l(\alpha,\gamma|\mathrm{K})/\partial\alpha=0\\ \partial l(\alpha,\gamma|\mathrm{K})/\partial\gamma=0\end{cases} \quad (48)$$

because the integral term appears in the wave-like log-likelihood function. The searching results for the two models are listed in *Table 7*.

Table 7. The MLE of parameters for wave-likelihood lifetime distributions

Type	I	II
$\hat{\alpha}$	6.5202 (0.00169)	0.0412 (0.01310)
		0.0961 (0.0006961)
$\hat{\gamma}$	0.0001 (0.00001)	0.0001 (0.00001)
		0.0206 (0.0000085)
$l(\hat{\alpha},\hat{\gamma}\|K)$	-3293.1074	-1719.2372
		-36496.9421
Accuracy	5.5807e-008	2.5757e-008
		2.1534e-006
Computation time	17.9384 sec.	17.0387 sec.
		74.9782 sec.

In the *Table 7*, for the parameter estimate columns, the top figures are the estimators whereas the figures in brackets are estimated standard deviations.

It is observed that in the case of Type I, the first pair gives the local maximum ($l(\hat{\alpha},\hat{\gamma}|K)$ = -3293.1074), the second is a global ($l(\hat{\alpha},\hat{\gamma}|K)$ = -1719.2372), whereas one suspects that the Type II model is a better description of the failure/repair process in operation here ($l(\hat{\alpha},\hat{\gamma}|K)$ = -36496.9421). We found two optimal solutions for Type II model (blue is the first, black is the second). The following three figures plot the estimated hazard functions and e-score plots.

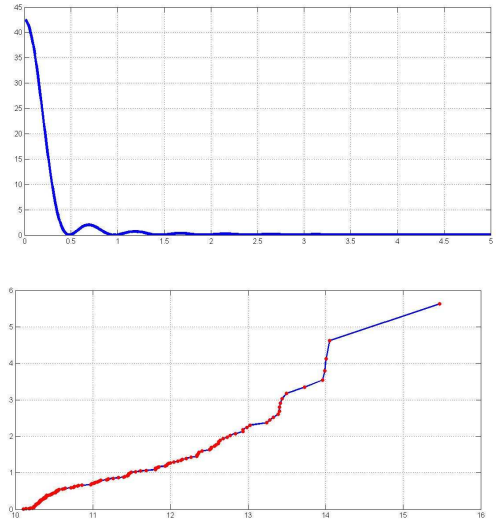


Figure 6. The estimated hazard function of Type I wave-like lifetime distribution ($\hat{\alpha}$ =6.5202, $\hat{\gamma}$ =0.0001) and approximated e-score plot

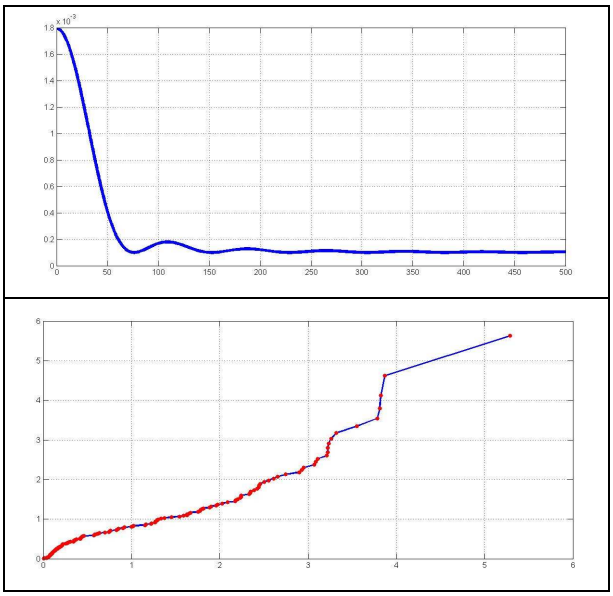
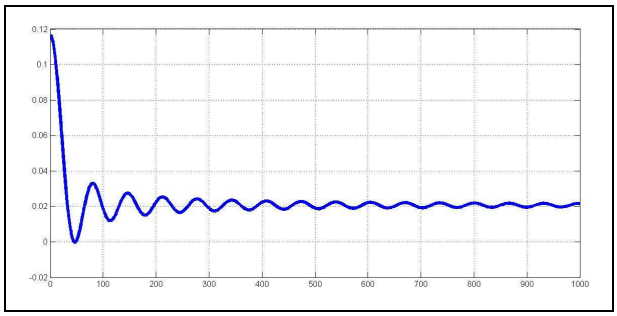


Figure 7. The estimated hazard function of Type I wave-like lifetime distribution ($\hat{\alpha}$ =0.0412, $\hat{\gamma}$ = 0.0001) and approximated e-score plot



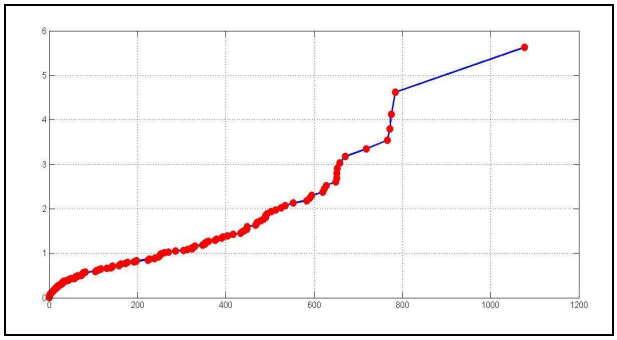


Figure 8. The estimated hazard function of Type II wave-like lifetime distribution ($\hat{\alpha}$ =0.0961, $\hat{\gamma}$ =0.0206) and approximated e-score plot

Remark 6.7. The e-score plot (Lawless [11]) is based on a fact that

$$\hat{e}_i = \int_0^{x_i} h(s;\hat{\alpha},\hat{\gamma})\,ds \overset{d}{\sim} \exp(-x_i) \tag{49}$$

and

$$\mathrm{E}\left[\hat{e}_{(i)}\right] = \sum_{l=1}^{i} \frac{1}{n-l+1} \tag{50}$$

where $\hat{e}_{(i)}$ is the i^{th} order statistic in calculated e-scores $\{\hat{e}_1,\hat{e}_2,\cdots,\hat{e}_N\}$. E-score plot plots ($\hat{e}_{(i)}$, $\mathrm{E}\left[\hat{e}_{(i)}\right]$), $i=1,2,\cdots,N$. If the plot demonstrates a straight-line then the good-fitness of the maximum likelihood is good enough. From the three e-score plots, we see similar patterns, but Type I model global result in *Figure 7* ($\hat{\alpha}$ =0.0412, $\hat{\gamma}$ = 0.0001) convinces us more.

7. Conclusion

In this paper, we introduce the new lambda algorithm first, and then investigate the underlying operating mechanism of the lambda algorithm. Furthermore, we explore the merging the lambda algorithm with maximum likelihood procedure. We have a detailed illustrative application. In the future, we will strive to explore more safety and reliability applications.

Acknowledgements

This research is supported by the South African National Research Foundation (IFR2010042200062) and (IFR2009090800013). The authors are grateful to Professor Charles Ernie Love (Simon Fraser University, Canada) for his data collection, and preliminary analysis efforts from the National Cement Company of the United Arab Emirates presented in the likelihood-lambda illustrative example.

References

[1] Ben-Gal, I. (2007). Bayesian Networks, in Ruggeri F., Faltin F. & Kenett R. *Encyclopedia of Statistics in Quality & Reliability*, Wiley & Sons.

[2] Blischke, W.R. & Murthy, D.N.P. (2000). *Reliability – Modeling, Prediction, and Optimization*. John Wiley & Sons, Inc. New York.

[3] Buntine, W. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research* 2.

[4] Cui, Y.H., Guo, R. & Guo, D. (2009). A Naïve five-element string algorithm. *Journal of Software*, 4,9, 925-934.

[5] Cui, Y., Guo, R., Dunne, T. & Guo, D. (2010). Lambda algorithm. *Journal of Uncertain Systems*, 4, 1, 22-23.

[6] Cui, Y., Guo, R., Savsani, V., Rao, R. & Vakharia, D. (2008). Harmony element algorithm- A naïve initial searching Range. *Proc. of the International conference on Advances in mechanical engineering*, 479-484.

[7] Guo, R., Guo, D. & Cui, Y.H. (2010). Wave-like bathtub hazard function. *Proc. of the Ninth International Conference on Information and Management Sciences*, 434-440.

[8] Guo, R., Love, C.E. & Cui, Y.H. (2010). Maximum Likelihood Estimation of 2-Parameter Wave-Like Lifetime Distributions. Advanced Reliability Modeling IV – Beyond the Traditional Reliability and Maintainability Approaches. *Proc. of the 4th Asia-Pacific International Symposium*, 225-232.

[9] Kjrulff, U. (1992). A computational scheme for reasoning in dynamic probabilistic networks, *Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence*, 121-129.

[10] Lawless, J.F. (1982). *Statistical Models and Methods for Lifetime Data*. Wiley, NY.

[11] Ushakov, I.A. (1994). *Handbook of Reliability Engineering*. John Wiley & Sons, Inc., New York.