Journal of Uncertain Systems
Vol.4, No.1, pp.22-33, 2010
Online at: www.jus.org.uk

Lambda Algorithm

Yanhong Cui[1], Renkuan Guo[1,*], Danni Guo[2]
[1] *Department of Statistical Sciences, University of Cape Town, Private Bag, Rondebosch 7701, Cape Town, South Africa*
[2] *Kirstenbosch Research Center, South African National Biodiversity Institute Private Bag X7, Claremont 7735, Cape Town, South Africa*

Received 25 July 2008; Revised 16 February 2009

Abstract

In this paper, we propose a new global optimization algorithm inspired by stochastic model and graph theory, which is named as lambda algorithm (LA). The new algorithm utilizes strings of element from member set {0, 1, 2, 3, 4} to represent the values of candidate solutions (typically represented as vectors in n-dimensional Euclidean space). LA draws useful information from both repeated and unrepeated elements from candidate solutions (strings), to simulate best global schema towards final optimization. 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 lambda algorithm engages the simplest mathematics but reaches the highest searching efficiency.

Keywords: string, λ operator, weight matrix, genetic algorithm, five-element string algorithm

1 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.

We have created a new global optimization search algorithm with following features:

First, the algorithm use lambda operator to build a five-state regular Markov chain model and initialize its individual solution as strings of 0s, 1s, 2s, 3s, 4s. Let P be a regular transition probability matrix with state space $S=\{0,1,2,3,4\}$. Then the limiting probability distribution $\Pi=(\pi 0,\pi 1,\pi 2,\pi 3,\pi 4)$ with $\pi i=0.2$ and satisfying $\Pi=\Pi P$. The property of limiting probability will hold the algorithm search result being independent of initial individual solution setting, keeping algorithm to seek the intrinsic harmonious state of a system under investigation.

Secondly, the algorithm considers five-state system as a filter, filtering advanced schema (ex.10**243**2) (Genetic algorithm's idea) from each individual solution. The key issue is selecting 3 different candidate solutions, their fitness values are similar. The algorithm made each of them compare to each other, then drawing out information from repeated and unrepeated digits. When most of candidate solutions of population have the same digit at same position, the probability of the digit portion is larger than a confidence value γ, $0\le\gamma\le 1$, thus the digit at this position goes to a steady state.

Thirdly, the algorithm operation is a pseudo-linear transformation such that searching the optimum of a nonlinear multivariate objective function is essentially linear. According to the limiting probability distribution property and advanced schema filtering, a weighting system is created, which leads some large weighted digits of population soon approach a probability stationary. Then extracting the steady digits by shrink the search area of candidate solutions, this procedure will repeat again and again since the algorithm is running, the algorithm will run towards to final optimum direction until the search area be shrink smaller enough. Thus, Cauchy sequences will be generated in this algorithm to satisfy the optimum approach condition.

2 An Inspiring Example

[*] Corresponding author. Email: rguo@worldonline.co.za (R. Guo).

The objective function for illustrating purpose is the Rosenbrock function

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

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

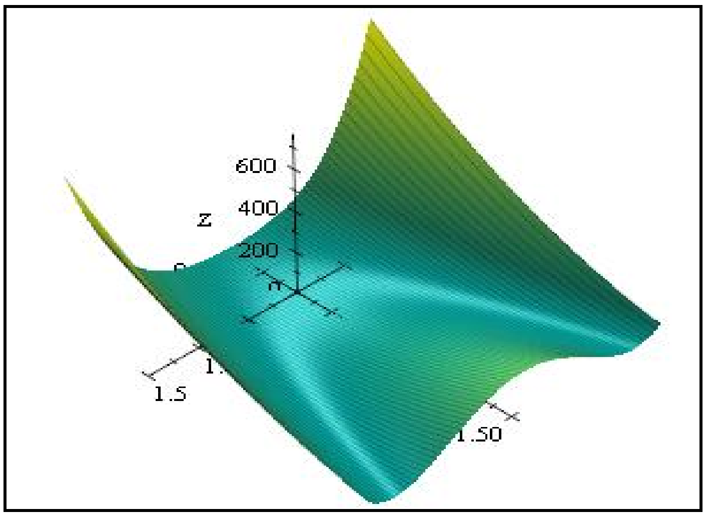


Figure1: Plot of Rosenbrok function

We used GA [7] 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 $\underline{x}=(x_1,x_2)^T \in D \subset \mathrm{R}^2$. 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 [5, 6, 9].

GA does not work on system state $\underline{x} \in D \subset \mathrm{R}^n$ of the objective function $f(\underline{x})$ directly, rather it uses string like 00110011001001110101111100 for representing the state $\underline{x}^T=(x_1,x_2,\ldots,x_n)$ 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 $\underline{x}^T=(x_1,x_2,\ldots,x_n)$ efficiently for covering the whole domain because the element change in a string is 1.

GA string member set {0,1} is too simple, which restrict itself not being able to operate complicated "mutation", only after "selection" and "crossover" two procedures' support, which then simulates advanced schema (ex.10*01**1). But for "crossover", for now we only follow "random crossover" or "60% crossover". It is a fuzzy intuitive way to operate, which restricts us to improve GA itself in further research. So, by expanding the string member set, we can make a more complicated "mutation" operation, and drop the fuzzy "crossover" idea, and it may lead to a better way to generate a more efficient heuristic algorithm.

3 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 $n_1,n_2,\ldots,n_t$. Number P is called the length of a string. For operational convenience, a string may be repressed by a row vector, $\underline{n}^T=(n_1,n_2,\ldots,n_p)$.

Definition 3.2: The collection of the elements for constructing a string, denoted by $\{0,1,\ldots,s-1\}$, 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 lambda 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 $\underline{x}=(x_1,x_2,\ldots,x_n)^T \in D \subset \mathrm{R}^2$ 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 $(x_1,x_2,\ldots,x_n)^T$ is

$$e_1e_2\cdots e_u e_{u+1}\cdots e_{2u}\cdots e_{(n-1)u+1}\cdots e_{nu}.$$

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

$$\underbrace{e_1e_2\cdots e_u}_{x_1}\ \underbrace{e_{u+1}\cdots e_{2u}}_{x_2}\cdots\underbrace{e_{(n-1)u+1}\cdots e_{nu}}_{x_n}.$$

Lemma 3.5: Let the system state be $\underline{x} \in D \subset \mathrm{R}^n$, 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 $\underline{u}_{\min}\le\underline{x}\in D$, $\underline{u}_{\max}\ge\underline{x}\in D$. The weight matrix $O=(o_{ij})_{n\times nu}$ with the i^{th} row vector $\underline{o}_i^T$ having a form

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

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

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

$$e=\begin{cases} e+1 & if \quad e\in\{0,1,2,\cdots.s-2\} \\ 0 & if \qquad e=s-1. \end{cases}$$

Definition 3.6's element e value change rule is called λ lambda operation, and $\lambda[\ ]$ is called lambda operator, we will discuss and define that part carefully at later section. In the remaining sections of this paper, we will use 5-element string for illustration and the establishment of the lambda algorithm.

4 Five-Element String Representation

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

Example 4.1: Let D≡$[u_{\min},u_{\max}]\times[u_{\min},u_{\max}]$ 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 1 1 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

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

Let

$$O_{2\times 2u} = \begin{bmatrix} \frac{5^5}{5^6} & \cdots & \frac{5^0}{5^6} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{5^5}{5^s} & \cdots & \frac{5^0}{5^s} \end{bmatrix},\ \underline{e}_{2u\times 1} = \begin{bmatrix} e_1 \\ \vdots \\ e_6 \\ e_7 \\ \vdots \\ e_{12} \end{bmatrix},\ \underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},\ \underline{u}_{\min} = \begin{bmatrix} u_{\min} \\ u_{\min} \end{bmatrix},\ u_r = u_{\max} - u_{\min}.$$

Then a matrix equation for string to state vector transformation is

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

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 $e_1e_2\ldots e_6e_7e_8\ldots e_{12}$, the 1-6 members are the first half of the string, representing x_1, the 7-12 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}$=$2\times10^{10}$. 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 1 summaries the changes and impacts.

Table 1: The impacts of weights in global searching and local tune-up

String	x_1	x_1	$\lvert \blacksquare x \rvert$
1 2 4 3 0 1 2 4 3 2 1 1	-3662720000	175680000	
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.

5 Lambda Operation

Before we introduce lambda operator, let us first review some useful knowledge of stochastic process and graph theory.

Suppose the Markov chain M is a digraph (directed graph) G=G_M having the set of nodes N={1,2,…,n} and the set of edges E. Each node corresponds to a state of M, and G contains edge (i,j)$\in$$E$ if and only if p_{ij}>0. Thus the digraph, or state transition diagram, G captures the structure of possible one-step state transitions. For any state i we let p_i denote the probability that, starting in state i, the process will ever reenter state i. State i is said to be recurrent if p_i=1 and transient if p_i<1.In graph-theoretic terms, $p_{ij}^k > 0$ means there is a directed path Q of length $l(Q)=k$(number of edges) from node i to node j in G. If this holds for some $k\geq0$, then node j is accessible from node i, written $i\to j$. If both $i\to j$ and $j\to i$ hold, then we say that states i and j communicate, written $i\leftrightarrow j$. A path joining a node to itself is called a circuit. If this circuit contains no repeated nodes, then it is a cycle.

Theorem 5.1: If G is strongly connected then there is a unique stationary distribution π for M. Moreover, this distribution satisfies $\pi_j>0$ for all $j\in N$.
Definition 5.2: (λ operator) let $e\in\{0,1,2,3,4\}$, then

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

$\lambda^{(l)}[\ ]$ is lth order λ operator , which repeats λ operation m times.
Definition 5.3: (Modulo operator): Let d be a positive integer, q be the quotient and r remainder r satisfying $d = nq + r$, Then we write the modulo operation as d mod(q) = r.
Definition 5.4: Let $\underline{e}=(e_1,e_2,\ldots,e_g)$ be a five-element string, then the λ operation on a string is a component-wise operation, i.e., $1\ [\underline{e}]= (1\ [e_1]\,1\ [e_2]\,L\ ,1\ [e_2])$. Furthermore, let $A=\left(e_{ij}\right)_{h\times g}$ be a five-element matrix, i.e., $e_{ij}\in\{0,1,2,3,4\}$, then $\lambda\,[A]=(\lambda[e_{ij}\,])_{h\times g.}$
Proposition 5.5: $\lambda^{(1)}[e]=\lambda^{(l\ \ \mathrm{mod}(\ 4))}[e]$, where $\lambda^{(0)}[e]\equiv e$.
Proof: Note that $e\in\{0,1,2,3,4\}$, the number e only has five choices. For example, $e=0$, and $\lambda^{(5)}[e]=\lambda^{(4)}[1]=\lambda^{(3)}[2]=\lambda^{(2)}[3]=\lambda^{(1)}[4]=0$.

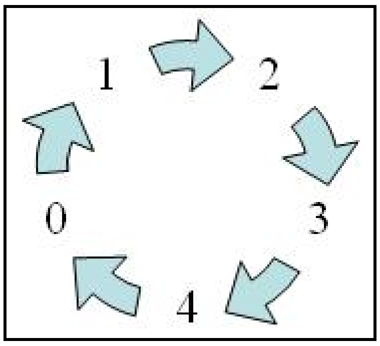


Figure 2: $\lambda[\ ]$operation cycle

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 lth-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, $\lambda^{(l)}[e]=\lambda^{(l\ \ \mathrm{mod}(4))}[e]$since 0 is the first member of the element set.

Proposition 5.6: 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 5.7: (λ spreading operation) Let $\underline{x}^{(k)}=\underline{u}_{\min}+u_r O\lambda^{(k)}[\underline{e}],k=0,1,2,3,4$ 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$\{\underline{e},\lambda^{(1)}[\underline{e}],\lambda^{(2)}[\underline{e}],\lambda^{(3)}[\underline{e}],\lambda^{(4)}[\underline{e}]\}$, the procedure to generate the string cycle, we called λ spreading operation.
Definition 5.8: (λ comparing operation) If we compare 2 strings $\underline{e}_1$ and $\underline{e}_2$, string $\underline{e}_1$ (candidate solution)'s fitness value better than string $\underline{e}_2$'s, then we manage some change to $\underline{e}_2$. l is length of strings $\underline{e}_1$, $\underline{e}_2$. e_{1i}, e_{2i} is one of the element of $\underline{e}_1$, $\underline{e}_2$ respectively.

```
For i=1:1:$l$
If $e_{1i}=e_{2i}$
$e_{2i}=\lambda(e_{2i})$
Else if $e_{1i}\neq e_{2i}$
$e_{2i}=e_{2i}$
End
End
```

From Definition 5.8 we know, compare to $\underline{e_1}$, $\underline{e_2}$ has $p(e_{1i} = e_{2i}) = 0.2$, and $p(e_{1i} \neq e_{2i}) = 0.8$, it means e_{2i} has probability 0.2 equal to $\lambda(e_{2i})$ to change its value, has probability 0.8 to keep its value.

We could consider *λ* comparing operation as a Markov chain model *M*, state space *S*={0,1,2,3,4}, the transition probability is *P*.

$$P = \begin{vmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 \\ 0.2 & 0 & 0 & 0 & 0.8 \end{vmatrix}.$$

Illustration the process, we could draw a digraph of *M*, called graph *G* [1], [2].

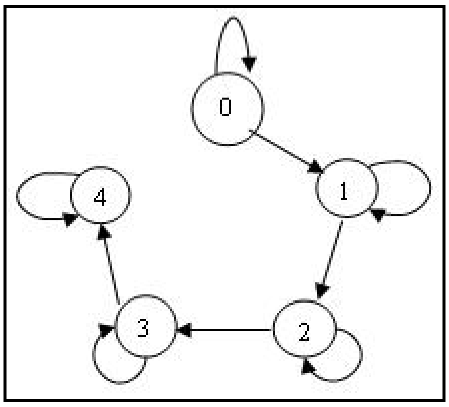


Figure 3: *λ* comparing operation model digraph

The Figure 3 graph *G* is a strongly connected graph, and then there is a unique stationary distribution *π* for *M*. (Definition given by theorem 5.1)

Let π_0, π_1, π_2, π_3, π_4 are stationary probability of state 0, 1, 2, 3, and 4. Then we have

$$\begin{cases} \pi_0 = 0.8\pi_0 + 0.2\pi_4 \\ \pi_1 = 0.8\pi_1 + 0.2\pi_0 \\ \pi_2 = 0.8\pi_2 + 0.2\pi_1 \\ \pi_3 = 0.8\pi_3 + 0.2\pi_2 \\ \pi_4 = 0.8\pi_4 + 0.2\pi_3 \\ \pi_0 + \pi_1 + \pi_2 + \pi_3 + \pi_4 = 1 \end{cases}.$$

From equation (2) we have: $\pi_0 = \pi_1 = \pi_2 = \pi_3 = \pi_4 = 0.2$.

And also: state 0 $\leftrightarrow$ state 1 $\leftrightarrow$ state 2 $\leftrightarrow$ state 3 $\leftrightarrow$ state 4.

From the above showing to us, the property of limiting probability will hold the algorithm search result independent of initial individual solutions setting, keep algorithm to seek the intrinsic harmonious state of a system under investigation.

Proposition 5.9: Let $f_{\min} = \min\{f(\underline{x}), f(\underline{x}^{(1)}), f(\underline{x}^{(2)}), f(\underline{x}^{(3)}), f(\underline{x}^{(4)})\}$, $f_{\max} = \max\{f(\underline{x}), f(\underline{x}^{(1)}), f(\underline{x}^{(2)}), f(\underline{x}^{(3)}), f(\underline{x}^{(4)})\}$. 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) $f_{\min} \leq f(x) \leq f_{\max}$, i.e., the cycle starting value $f(\underline{x})$ falls between cycle minimum and maximum.

Remark 5.10: 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 *λ*[ ] operations guarantee the chance for global optimum searching.

6 LA Optimization Working Principal

From our intuitive observation of whole population of candidate solutions (5-elements represent), the only information we could directly observe is: about 20% digits repeated at same position of each string (candidate solution), and 80% digits are not. Different with many optimization algorithms, the LA draws useful information from both repeated and unrepeated elements from candidate solutions (strings), to simulate best global schema towards final optimization.

Now we start to introduce the optimization principal of LA. Suppose we have *N* randomly simulated candidate solutions, called population of strings.

1. Rank the whole *N* strings by fitness checking, sort them by ascending order according to objective function values with respect to *N* strings.

2. Packed-Rolling operation. 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;
end
end
end
```

Verbally, Packed-Rolling operation can be explained as follows: Defined 3 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. If the second string don't have repeated element, 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. 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-2 package is defined and checked.

3. Executive λ spreading operation: spreading the string vector $\underline{Q}$ to $\{\underline{Q},\lambda^{(1)}[\underline{Q}],\lambda^{(2)}[\underline{Q}],\lambda^{(3)}[\underline{Q}],\lambda^{(4)}[\underline{Q}]\}$, then all the strings is re-evaluated via $f(\underline{u}_{\min}+u_r O\underline{e})$. From $\{\underline{Q},\lambda^{(1)}[\underline{Q}],\lambda^{(2)}[\underline{Q}],\lambda^{(3)}[\underline{Q}],\lambda^{(4)}[\underline{Q}]\}$, then select best fitness N strings as new string vector $\underline{Q}'$ .From Packed-Rolling operation procedure we could see, 3 fitness values most similar candidate solutions be seemed as one "package". Whole population of candidate solutions is classified to ROW-2 packages. The repeated elements according to different classified "package" were transferred to $\lambda^{(4)}[\underline{Q}]$, unrepeated elements stay at $\underline{Q}$. all repeated and unrepeated elements joined with new digits combined to new strings via evaluation. Select 1/5 strings from whole $\{\underline{Q},\lambda^{(1)}[\underline{Q}],\lambda^{(2)}[\underline{Q}],\lambda^{(3)}[\underline{Q}],\lambda^{(4)}[\underline{Q}]\}$, ask comparing two new strings which contains unrepeated and repeated elements. Loop above procedure several times, more and more advanced schema is simulated by testing and selection, successfully draws out the useful information during LA running.

The algorithm operation is a pseudo-linear transformation such that searching the optimum of a nonlinear multivariate objective function is essentially linear. According to the limiting probability distribution property and advanced schema filtering, a weighting system is created via $f(\underline{u}_{\min}+u_r O\underline{e})$, which leads some large weighted digits of population soon approach a probability stationary.

Example: For $\underline{e}=(e_1,e_2,\cdots,e_g)$, by weight comparing, we has $\underline{e}=(e_1>e_2>e_3\cdots>e_g)$, then e_1,e_2 might be first and second elements approach probability stationary.

In LA, larger weight element always approach probability stationary more soon than smaller weight element, it means before e_1 goes to stationary, $e_2,e_3,\cdots,e_g$ have no way to approach.

Definition 6.1: (Cauchy sequence): A sequence x_1, x_2, x_3 of real numbers is called Cauchy, if for every positive real number ε, there is a positive integer *N* such that for all natural numbers m, n > N, $|x_m - x_n| < \varepsilon$ (where the vertical bars denote the absolute value). (See Figure 4)

When e_1 goes to stationary, the algorithm will ask to extract e_1 from $\underline{e} = (e_1, e_2, \cdots, e_g)$. e_2 will transform to new e_1', e_3 to e_2', …, e_g to e_{g-1}'. A new e_g will add by simulation randomly. A new $u_{\max}'$, $u_{\min}'$ will replace $u_{\max}$, $u_{\min}$. Also $|u_{\max}' - u_{\min}'| < |u_{\max} - u_{\min}|$.Repeat this procedure again and again when algorithm is running, until $|u_{\max}^{(n)} - u_{\min}^{(n)}| < \varepsilon$, ε is a smaller enough positive number, we say the algorithm approach to final optimum.

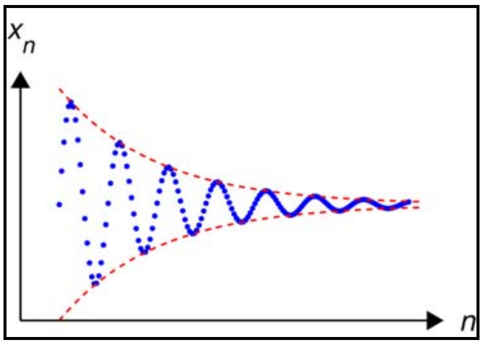
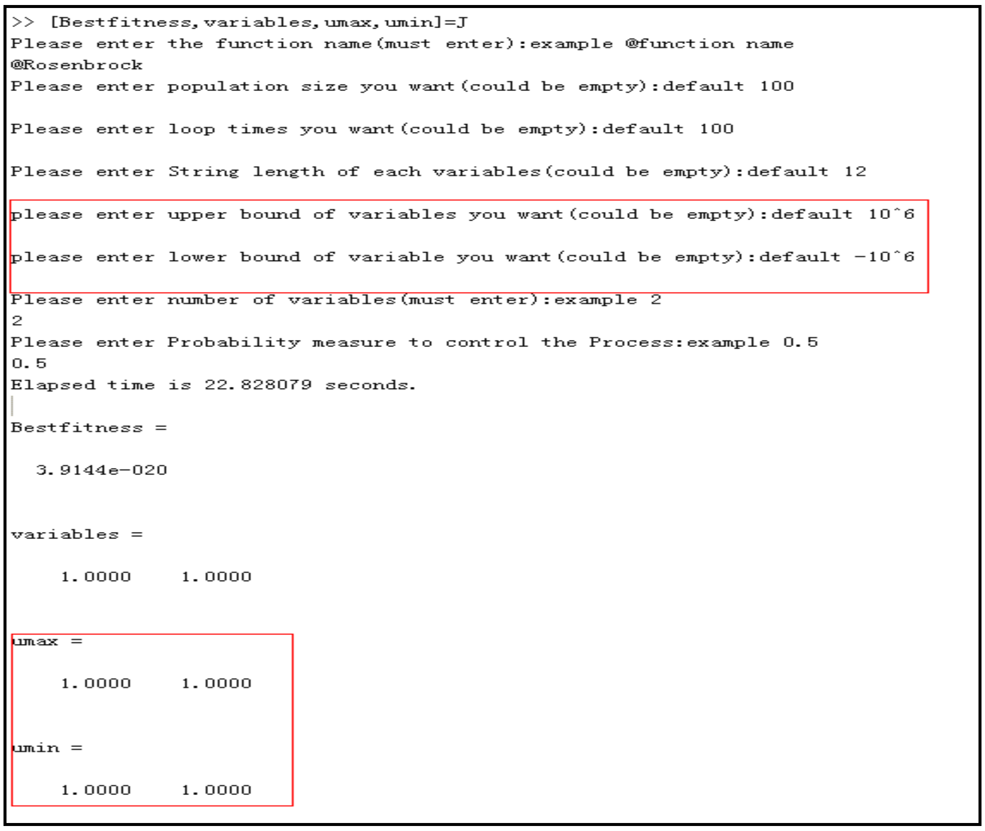


Figure 4: Cauchy sequence illustration


```
>> [Bestfitness,variables,umax,umin]=J
Please enter the function name(must enter):example @function name
@Rosenbrock
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

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
Please enter Probability measure to control the Process:example 0.5
0.5
Elapsed time is 22.828079 seconds.

Bestfitness =

  3.9144e-020


variables =

    1.0000    1.0000


umax =

    1.0000    1.0000


umin =

    1.0000    1.0000
```
Figure 5: Two dimensional Rosenbrock LA optimization result

From Figure 5 we can see, at the beginning, the search area given by program is:

$u_{min} \le x_1, x_2 \le u_{max}$, $u_{min} = -1.0 \cdot 10^6$, $u_{max} = -1.0 \cdot 10^6$, $|u_{max} - u_{min}| = 2.0 \cdot 10^6$, after $u¢_{\min}$ =0.999999930368000, $u¢_{\max}$ =1.000000035225600, $|u^{ii}_{\max}$ - $u_{\min}|$ =1.048576·10^{-7}, at computer screen, it shows: $u¢_{\min}$ =1.000, $u¢_{\max}$ =1.000.

Cauchy sequence idea is proved. Confidence probability given by 0.5, if e_i's portion probability larger than 0.5, we consider e_i goes to stationary and exact e_i out.

7 A Lambda Algorithm 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.

Confidence probability: Give a probability P, usually let $0.5 \le P \le 1$, if 1st element's portion probability larger than P, then we believe the element will goes to stationary. 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 LA 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: 2N string cycles creation. By applying λ spreading operation to $Q_{\min}$ and $Q_{\max}$ respectively, ten string vectors (including $Q_{\min}$ and $Q_{\max}$), denote them by Q_i , i=1,…,5,6,…10. Note that $Q_1 = Q_{\min}$ and $Q_6 = Q_{\max}$. Mathematically,

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

Mathematically, **step 1** is creating 200 ($2N$ in general) string cycles according to **Proposition 5.6**, which paves the way toward the global optimum searching.

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

1. Combine Q_i, i=1,2,…,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 5.9.

Checking stationary of $Q'_{\min}$. If each variable's string representation, its 1st element's portion probability larger than P, then we believe the element will goes to stationary. Exact the element, and a new $u'_{\max}$, $u'_{\min}$ will replace $u_{\max}$, $u_{\min}$. Search area $|u'_{\max} - u'_{\min}| < |u_{\max} - u_{\min}|$. * $Q'_{\max}$ will be selected from $Q_{\min}$'s λ spreading operation 500 strings. The operation is ranked 500 strings by descending order, $Q'_{\max}$ will be top 100.

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 λ[ ] 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.

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 5.9 guarantees the success of the scheme as what we pointed in Remark 5.10.

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 LA optimization searching scheme is shown in Figure 6.

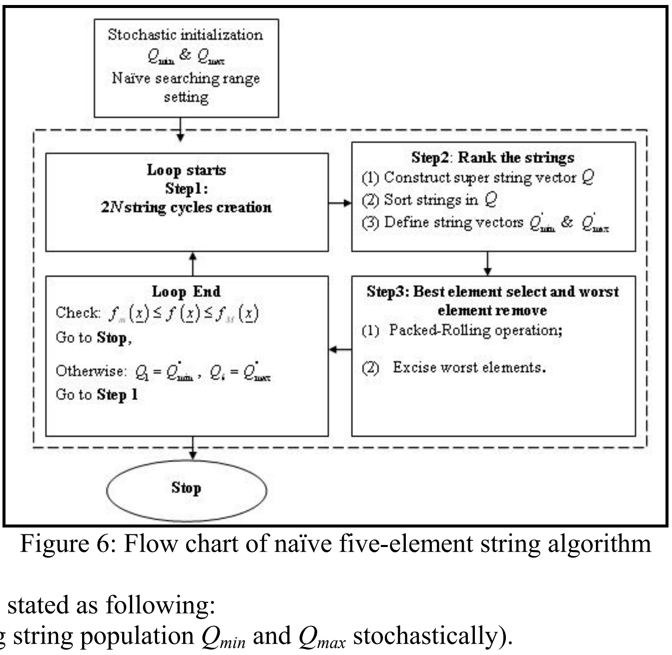

Stochastic initialization
Q_{min} & Q_{max}
Naïve searching range setting
Loop starts
Step1:
2N string cycles creation
Step2: Rank the strings
(1) Construct super string vector Q
(2) Sort strings in Q
(3) Define string vectors Q'_{min} & Q'_{max}
Step3: Best element select and worst element remove
(1) Packed-Rolling operation;
(2) Excise worst elements.
Loop End
Check: $f_m(\underline{x}) \le f(\underline{x}) \le f_M(\underline{x})$
Go to Stop,
Otherwise: $Q_1 = Q'_{min}$, $Q_i = Q'_{max}$
Go to Step 1
Stop

Figure 6: Flow chart of naïve five-element string algorithm

The LA algorithm can be stated as following:

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

Start loop

2N string cycles creation;

Rank the strings and checking stationary;

Best element select and worst element remove;

Check the loop stop criteria: (yes, GoTO 1, yes, Loop Stops);
End loop

8 Illustrative Examples

We use Lambda 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

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

The Lambda algorithm searching by 100 loops gives $f_{\min}$ =3.9144e−020, and the global minimum (x_1^m,x_2^m) =(0.999999999878608,0. 999999999741594). The searching area is D≡[−10^6,10^6]× [−10^6,10^6]. Spend 22.828079 seconds.

B. Rastrigin function

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

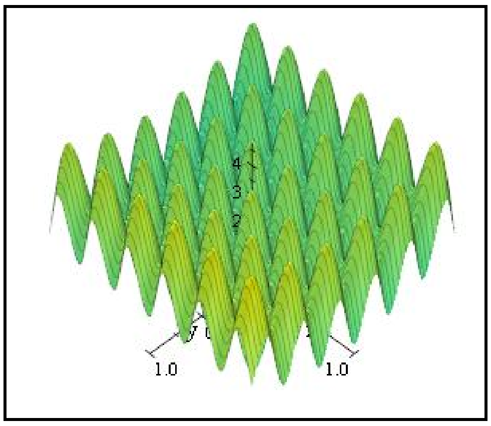


Figure 7: 3D-plot of Rastrigin function

The global minimum is 0 at (x_1^m,x_2^m) =(0,0) and it is well-known that in area [−1,1]× [−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 D≡[−10^6,10^6]× [−10^6,10^6] by 26 loops gives the global minimum 0 at (x_1^m,x_2^m) =(0,0).

Table 2: Comparisons between GA and five-element naïve string algorithm

Function	Algorithm	Searching cube	Loops	Global min
Rosenbrock	GA	$[-10^6,10^6]^2$	57	0.11935
	LA	$[-10^6,10^6]^2$	100	3.9144e-020
Rosenbrock	GA	$[-5.12,5.12]^{30}$	84	105.7045
	LA	$[-5.12,5.12]^{30}$	400	1.0165
Rastrigin	GA	$[-10^6,10^6]^2$	51	1.2178
	LA	$[-10^6,10^6]^2$	26	0.000
Griewank	GA	$[-10^6,10^6]^2$	52	0.12506
	LA	$[-10^6,10^6]^2$	77	0.000
Griewank	GA	$[-600,600]^{1000}$	52	0.55736
	LA	$[-600,600]^{1000}$	37	2.3360e-011

C. Griewank function

This function is 1000-dimensional. In the cube $[-600,600]^{1000}$, there are thousands of local minima around and the global minimum 0 at the origin.

$$f_n(\bar{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, i = 1,2,\cdots,1000\,.$$

Using lambda algorithm to search in the cube, by 37 loops, the algorithm locates x_i=0.2458, i=1,2,…,1000 which gives global minimum 2.3360e-011. Spend 337.4632 seconds.

9 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.

In the new algorithm development, (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 say 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 it already has successful applications [3], [4], [8] should be examined further.

Acknowledgements

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

References

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

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

[3] Cui, Y.H., and R. Guo, Naïve String Algorithm, *Proceedings of 2008 International Workshop on Education Technology and Training and 2008 International Workshop on Geoscience and Remote Sensing*, pp.517-520, 2008.

[4] Cui, Y.H., R. Guo, *et al.*, Harmony element algorithm – A naive initial searching range, *Proceedings of the International Conference on 'Advances in Mechanical Engineering'*, pp.479-484, 2008.

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

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

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

[8] Savsani, V.J., R.V. Rao, *et al.*, Modified Harmony elements algorithm for the optimization of mechanical engineering design, *Proceedings of the International Conference on 'Advances in Mechanical Engineering'*, pp.545-550, 2008.

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

Theorem 8 *Let $(\Omega,\ d)$ be a metric space and let $\boldsymbol{B}$ be a partition of Ω. For every $B \in \boldsymbol{B}$ with positive and finite Hausdorff outer measure in its dimension denote by $\mu = P(A|B)$ the restriction to the Borel σ-field of the upper conditional probability defined as in Theorem 2. Let $L^*(B)$ be the class of all Borel measurable random variables on B. Then the convergence in μ-distribution of a sequence of random variables of $L^*(B)$ to a random variable X is equivalent to the pointwise convergence of expectation functionals on all bounded and continuous function f that is $lim_{n\to\infty} \int f d\mu_n = \int f d\mu$.*

Proof: If X_n and X are Borel-measurable random variables and H is a Borelian set then the sets $X_n^{-1}(H)$ and $X^{-1}(H)$ are also Borelian sets; moreover since every Hausdorff s-dimensional outer measure is countably additive on the Borel σ-field then the (upper) conditional probabilities μ_n and μ induced respectively by X_n and X on $(\Re,\boldsymbol{F})$ are probability measures. Then convergence in μ-distribution is equivalent to the pointwise convergence of expectation functionals on all bounded and continuous function f.

6 Conclusions

This paper investigates the relations among different types of convergence for random variables when they are based on an upper probability approach where conditional upper expectations with respect to Hausdorff outer measures are used whenever we have to condition on a set with probability zero.

Upper (lower) conditional previsions defined with respect to Hausdorff outer measures are proven to be the upper (lower) envelopes of all linear extensions to the class of all random variables of the restriction to the Borel-measurable random variables of the given upper conditional previsions.

It is proven that the relations among different types of convergences of random variables defined with respect to upper conditional probability defined by Hausdorff outer measures are the same that hold if convergences are defined with respect to a probability measure. When the conditioning event has finite Hausdorff outer measure in its dimension these results are obtained because Hausdorff outer measures are Borel regular outer measures and so continuous from below and continuous from above on the Borel σ-field. In general if upper conditional probability is defined as natural extension of a coherent merely finitely additive probability defined on a σ-field we have that μ-stochastically convergence does not imply convergence in μ-distribution since in this case the upper conditional probability is not continuous from above.

References

[1] Billingsley, P., *Probability and Measure*, Wiley, 1986.

[2] Couso, I., S. Montes, and P. Gil, Stochastic convergence, uniform integrability and convergence in mean on fuzzy measure spases, *Fuzzy Sets and Systems*, vol.129, pp.95–104, 2002.

[3] de Cooman, G., M.C.M. Troffaes, and E. Miranda, n-Monotone lower previsions and lower integrals, *Proceedings of the Fourth International Symposium on Imprecise Probability: Theories and Applications*, pp.145–154, 2005.

[4] Doria, S., Probabilistic independence with respect to upper and lower conditional probabilities assigned by Hausdorff outer and inner measures, *International Journal of Approximate Reasoning*, vol.46, pp.617–635, 2007.

[5] Denneberg, D., *Non-additive Measure and Integral*, Kluwer Academic Publishers, 1994.

[6] Falconer, K.J., *The Geometry of Fractals Sets*, Cambridge University Press, 1986.

[7] Rogers, C.A., *Hausdorff Measures*, Cambridge University Press, 1998.

[8] Seidenfeld, T., M. Schervish, and J.B. Kadane, Improper regular conditional distributions, *The Annals of Probability*, vol.29, no.4, pp.1612–1624, 2001.

[9] Walley, P., *Statistical Reasoning with Imprecise Probabilities*, Chapman and Hall, London, 1991.