Random Search via Probability Algorithm

for Single Objective Optimization Problems

Tran Van Hao, Nguyen Huu Thong
Mathematics-Informatics department, University of Pedagogy
280, An Duong Vuong, Ho Chi Minh city, Viet Nam
E-mail: thong_nh2002@yahoo.com

Abstract—*This paper proposes a new numerical optimization technique, the Random Search via Probability Algorithm (RSPA), for single optimization problem. There are three problems: 1-To evaluate objective functions, the role of left digit is more important than the role of right digit of a decided variable, 2-The relation of decided variables in the formula of constrains and object function, 3-We can't calculate exactly the number of iterance of a random algorithm. Based on these remarks, we calculate mutative changing probabilities of digits of decided variables for searching optimal solutions, we select k (1≤k≤n) variables to change the value of variables for every iterance, and we use unfixed number of iterance, which has capability to find an optimal solution first with necessary number of iterance. We tested this approach by implementing the Random Search via Probability Algorithm on some test single objective optimization problems, and we found results very stable.*

1. THE MODEL OF SINGLE OBJECTIVE OPTIMIZATION PROBLEM

We consider a model of single objective optimization problem as follows:

Minimize $f(x)$

subject to

$$g_j(x) \le 0 \qquad (j = 1,\ldots,r)$$

where

$$a_i \le x_i \le b_i,\ \ a_i, b_i \in R,\ \ i = 1,\ldots,n.$$

We suppose that every dicided variable x_i ($1 \le i \le n$) have m digits that are listed from left to right $x_{i1}, x_{i2},\ldots, x_{im}$ (x_{ij} is an integer and $0 \le x_{ij} \le 9$). The role of left digit x_{ij} is more important than the role of right digit $x_{i,j+1}$ ($1 \le j \le m-1$) for evaluating objective function.

2. PROBABILITIES OF CHANGE

2.1. Probabilities permit to change values of variables

Base on feasible region of problem is narrow or large; we have two cases as follows:

2.1.1. Case 1: the right digit depend on every the left digit.

Let q_j be changing probability of digit j ($1 \le j \le m$), let A_j be event digit x_j find its correct value of an optimal solution.

- Probability of event A_1 to occur

$$p(A_1) = q_1 * \frac{1}{10}$$

This probability is max if $q_1 = 1$

- Probability of event A_2 to occur

The second digit can only find its correct value after the first digit has found its correct value. We have two small cases as follows:

Small case 1: The first digit found its correct value, we have to fix the first digit and change the value of second digit to can find the correct value of second digit of an optimal solution.

$$p(A_2 / A_1) = (1 - q_1) * q_2 * \frac{1}{10}$$

Small case 2: The first digit did not found its correct value

$$p(A_2 / \overline{A_1}) = 0$$

Applying conditional probability, we have:

$$p(A_2) = p(A_1)p(A_2 / A_1) + p(\overline{A_1})p(A_2 / \overline{A_1}) = p(A_1)p(A_2 / A_1)$$
$$= q_1 * \frac{1}{10}(1 - q_1)q_2 * \frac{1}{10} = \frac{1}{10^2} * q_1 * (1 - q_1)q_2$$

Because q_1 and q_2 are independent, this probability is max if $q_1 = \frac{1}{2}$ and $q_2 = 1$.

- Generally, probability of event A_j ($2 \le j \le m$)

The j th digit can only find its correct value after all left digits (1, 2, …, j-1) has found its correct value. We have two small cases as follows:

Small case 1: All left digits found its correct value, we have to fix all left digit and change the value of j th digit to can find the correct value of j th digit of an optimal solution.

$$p(A_j / A_1 A_2 \dots A_{j-1}) = (1 - q_1) * (1 - q_2) * \dots * (1 - q_{j-1}) * q_j * \frac{1}{10}$$

Small case 2: There is a left digit that has not found its correct value.

$$p(A_j / A_1 \dots \overline{A_k} \dots A_{j-1}) = 0 \qquad (1 \le k \le j - 1)$$

$$p(A_j / A_1 \dots \overline{A_k} \dots \overline{A_l} \dots A_{j-1}) = 0 \qquad (1 \le k < l \le j - 1)$$

……

$$p(A_j / \overline{A_1}\,\overline{A_2} \dots \overline{A_{j-1}}) = 0$$

Applying conditional probability, we have:

$$p(A_j) = p(A_1 A_2 \dots A_{j-1})p(A_j / A_1 A_2 \dots A_{j-1}) + 0 = p(A_{j-1})p(A_j / A_1 A_2 \dots A_{j-1})$$
$$= \left[\frac{1}{10^{j-1}} q_1 (1 - q_1)^{j-2} q_2 (1 - q_2)^{j-3} \dots q_{j-1}\right] \times \left[(1 - q_1)(1 - q_2)\dots(1 - q_{j-1}) q_j \frac{1}{10}\right]$$
$$= \frac{1}{10^j} q_1 (1 - q_1)^{j-1} q_2 (1 - q_2)^{j-2} \dots q_{j-1}(1 - q_{j-1}) q_j$$

Because probabilities $q_1, q_2, \ldots, q_j$ are independent, we have probability $P(A_j)$ max if

$$q_1 = \frac{1}{j}, q_2 = \frac{1}{j-1}, \ldots, , q_{j-1} = \frac{1}{2}, q_j = \frac{1}{1} = 1$$

- Average probabilities of change

$$p_j = \frac{1}{j}(1 + \frac{1}{2} + \ldots + \frac{1}{j}) \qquad (1 \le j \le m)$$

Example: m=7, p=(0.37, 0.41, 0.46, 0.52, 0.61, 0.75, 1)

2.1.1. Case 2: Some problems have constrains too tie, its feasible regions are very small or narrow. To find a feasible solution, many left successive digits have to find its correct values simultaneously. We consider case m=7:

Probabilities for finding 7 digits	1	1	1	1	1	1	1
Probabilities for finding next digits.	½	1	1	1	1	1	1
	½	½	1	1	1	1	1
	½	½	½	1	1	1	1
	½	½	½	½	1	1	1
	½	½	½	½	½	1	1
	½	½	½	½	½	½	1
Average probabilities	0.57	0.64	0.71	0.79	0.86	0.93	1

With m=7, we have two changing probabilities as follows:

Probability I: (0.37, 0.41, 0.46, 0.52, 0.61, 0.75, 1)

Probability II: (0.57, 0.64, 0.66, 0.70, 0.75, 0.83, 1)

Remarks:

- The changing probabilities of digits of a variable are increase from left to right. It means that left digits are stable than right digits.
- According to statistics of many experiments, the best thing is to use probability I in the ratio 60%-70% and probability II in the ratio 40%-30%.

3.2. Probabilities for selecting value of a digit.

When probability p_j happen, let r_1, r_2 and r_3 be probabilities of events as follows:

r_1: choose a random integer number between 0 and 9 for jth digit

r_2: probability of increase j digit.

r_3: probability of decrease j digit.

Now we consider two digits a_j and a_{j+1}. When two probabilities 1-p_j and p_{j+1} happen, if aj has correct value of an optimal solution, we have the probability so that a_j and a_{j+1} can find its correct values of the same optimal solution as follows

$$r_1 \frac{1}{10} + r_2 \frac{1}{100} + r_3 \frac{1}{100}$$

Because of $r_1+r_2+r_3=1$, this probability is max if $r_1=1$, $r_2=r_3=0$.

If aj had not correct value of an optimal solution, we have the probability of digits a_j and a_{j+1} can find correct values of an optimal solution as follows

$$r_1 0 + r_2 \frac{1}{100} + r_3 \frac{1}{100}$$

Because of $r_1+r_2+r_3=1$, this probability is max if $r_1=0$, $r_2=r_3=0.5$

The average probabilities r_1, r_2 and r_3 of both two cases as follows: $r_1=0.5$, $r_2=r_3=0.25$

3.3. Selecting k variables (1≤k≤n) to change its values

In every iteration, we select k variables (1≤k≤n) to change its values.

Probability of a variable to be selected: $\frac{C_{n-1}^{k-1}}{C_n^k} = \frac{k}{n}$

Probability of a digit can find its correct value:

$$\frac{k}{n} p_j \frac{1}{10} = \frac{kp_j}{10n} \qquad (1 \le j \le m)$$

Let A be an event such that k variables are selected and can find its correct values of an iteration:

$$T = T(A) = \left(\frac{kp_{j_i}}{10n} \right)^k$$

with $1 \le j_i \le m,\ 1 \le i \le k$.

Let X is a random variable that represents number of appearances of event A in d iterations of algorithm. X conforms to the law of binomial distribution *B(d,X)*. We have:

$$p(X = x) = C_d^x T^x (1-T)^{d-x}$$

Probability of event A happened at least once is:

$$p(X > 0) = 1 - p(0) = 1 - (1-T)^d$$

We calculate number of iterations d such that this probability greater or equal α (0<α<1):

$$\begin{aligned} p(X > 0) \ge \alpha &\Rightarrow 1 - (1-T)^d \ge \alpha \\ &\Rightarrow (1-T)^d \le 1 - \alpha \\ &\Rightarrow d \ln(1-T) \le \ln(1-\alpha) \\ &\Rightarrow d \ge \frac{\ln(1-\alpha)}{\ln(1-T)} \\ &\Rightarrow d \ge \frac{\ln(1-\alpha)}{\ln\left(1 - \left(\frac{kp_{j_i}}{10n} \right)^k \right)} \end{aligned}$$

Select a minimum number of iterations and when $n \to +\infty$, we have:

$$\frac{\ln(1-\alpha)}{\ln\left(1-\left(\frac{kp_{j_i}}{10n}\right)^k\right)} \approx \frac{\ln(1-\alpha)}{-\left(\frac{kp_{j_i}}{10n}\right)^k} = -\ln(1-\alpha)\left(\frac{10n}{kp_{j_i}}\right)^k \le -\ln(1-\alpha)\left(\frac{10n}{kp_1}\right)^k = -\ln(1-\alpha)\left(\frac{10}{p_1}\right)^k \left(\frac{1}{k}\right)^k n^k$$

because of $p_1<p_2<\ldots<p_m$.

We have a necessary number of iterations for transforming a solution from a current state to a better state:

$$-\ln(1-\alpha)\left(\frac{10}{p_1}\right)^k \left(\frac{1}{k}\right)^k n^k$$

In every iteration, it takes k*O(1) of time. We have necessary time for transforming a solution from a current state to a better state:

$$-\ln(1-\alpha)\left(\frac{10}{p_1}\right)^k \left(\frac{1}{k}\right)^k n^k kO(1)$$

If k is a fix number, independent from n, the complexity for transforming a solution from a current state to a better state is $O(n^k)$. If k=n, it is $O(na^n)$ where $a=10/p_1$.

With k=1, in every iteration we select only one variable for transforming a solution from a current sate to a better state. These problems of this type as follows:

$$\textit{Minimize} \qquad f(x) = \sum_{i=1}^{n} f_i(x_i)$$

$$a_i \le x_i \le b_i, \;\; a_i, b_i \in R, \;\; i = 1,\ldots,n.$$

In the formula of objective function, every variable x_i is calculated independently of other variables.

4. THE RANDOM SEARCH VIA PROBABILITY ALGORITHM (RSPA)

We suppose that a solution of problem has n decided variables, every variable has m=7 digits. Let BD=50000. RSPA is described generally as follows:

Select BD=50000 (a number for changing a solution in an iteration)

RSPA is described with general steps as follows:

b1. Choose a random feasible solution x, calculate its value of objective function: Fx=f(x).

b2. Loop:=0;

b3. We assign the value of x to y (y:=x).

b4. Chose k variables ($1 \le k \le n$) of solution y, sign y_i ($1 \le i \le k$).

b5. Let P= (p_1, p_2,…, p_m).

if (probability of a random event is 30%) then <select probabilities I for P>

else < select probabilities II for P >

b6. if (probability of a random event is 50%) then

<select a random number fix from 3 to 5, and set pi=0 (i=0,.., fix)>

b7. Let y_{ij} be jth digit ($1 \le j \le m$) of variable y_i ($1 \le i \le k$). The technique for changing value via probability of j th digit is described as follows

for i:=1 to k do

Begin

for j:=1 to m do

if (probability of a random event is p_j) then

if (probability of a random event is r_1) then

< select a random integer from 0 to 9 for y_{ij} >

else if (probability of a random event is r_2) then

< y_{ij} + random(a) (where a=2 or 3 if i=0,2,3) >

else < y_{ij} − random(a) (where a=4 or 5 if i=4,5,6) >

else < don't change the value of digit y_{ij} >;

if ($y_i < a_i$) then $y_i = a_i$; if ($y_i > b_i$) then $y_i = b_i$;

End;

b8. If y is an infeasible solution then return b3.

b9. Let Fy=f(y).

b10. If Fy <Fx then x:=y; Fx:=Fy; loop:=0;

b11. if loop<BD then loop:=loop+1, return b3.

b12. End of RSPA.

RSPA has characteristics as follows:

- RSPA search correct values of digits of variables from left digits to right digits of every variable according to the guide of probabilities.
- Variable Loop will be set 0 if RSPA search a better solution. It means that RSVA can find an optimal solution with a necessary number of iterations.
- In step 6, the left digits can be found very quickly therefore we set the right probabilities pi=0 with probability 50% to increase the speed for searching the right digits.
- In step 7, we set a=2 or 3 if i=0,2,3 and a=4 or 5 if i=4,5,6 because the right digits have change very high.

5. EXAMPLES

Using PC, Celeron CPU 2.20GHz, Borland C 3.1. Select value to parameter BD=100000. Statistical table below of 30 performances of RSPA.

5.1. Multimodal test functions [2][3]

5.1.1. Five multimodal test functions

These five functions below have size n= 30, -100≤xi≤100 (i=1,…, n).

Functions	Minimum
Ackley's function [1]	$f(x) = 20 + e - 20 * \exp\left(-0.2 * \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - epx\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2 * \pi * x_i) \right)$
Schwefel's function [2]	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
Rastrigin's function [1]	$f(x) = 10 * n + \sum_{i=1}^{n} (x_i^2 - 10 * \cos(2 * \pi * x_i))$
Sphere function [1]	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2$
Ellipsoid function[2]	$f(x) = \sum_{i=1}^{n} i * x_i^2$

Optimal solutions of these five problems above are: x_i=0 (i=1,…,n), f(x)=0.

- Statistical table of five multimodal test functions (Time was counted according to second).

		Function									
		Ackley		Schwefel		Rastrigin		Sphere		Ellipsoid	
	f(x)	Iterations	Time	Iterations	Time	Iterations	Time	Iterations	Time	Iterations	Time
Best	0	53335	25	58992	24	69259	34	93946	36	137784	40
Worst	0	107857	38	98174	32	126048	49	212543	52	376679	75
Average	0	84140	32	74109	26	89747	39	138568	41	231405	52
Median	0	83456	33	73275	25	84823	38	137049	39	202025	49

5.1.2. Rosenbrock's function [2]

Minimun $f(x) = \sum_{i=1}^{n-1} (100 * (x_i^2 - x_{i+1}^2) + (x_i - 1)^2)$

$-100 \le x_i \le 100 \ \ (i = 1, \ldots, n)$

Optimal solution: x_i=1 (i=1,…,n), f(x)=0.

- Statistical table of Rosenbrock's function

N	f(x)	Iterations
10	0.0009	278571
20	0.0009	1657817
30	0.0019	7050527

Remarks about experiment problems: Variables of problems 1-5 are not interdependent, the increase or decrease of one variable x_i ($1 \le x_i \le n$) influences only on a term including this variable x_i. Therefore we can randomly select one variable (k=1) to change its values. However we can select k=1+10% of n to increase convergent speed of algorithm. Experiment problem 6 has variables

interdependent, the increase or decrease of one variable x_i ($1 \le x_i \le n$) influences previous term and back term, therefore we have to select many variables ($k>1$) to change its values. See the statistical table of problem 6 for n=10, n=20 and n=30, because problem 6 has variables interdependent therefore number of iterations increases very quickly in the ratio of n. It means that the complexity of RSPA for these problems is not based on formulas of problems, such as linear or nonlinear, but is based on the relations of variables in objective function or constrains.

5.2. Four engineering design problems

5.2.1. Design of a Welded Beam

5.2.1.1. The version of Coello [4]

Minimize $f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

subject to

$$g_1(X) = \tau(X) - \tau_{\max} \le 0$$
$$g_2(X) = \sigma(X) - \sigma_{\max} \le 0$$
$$g_3(X) = x_1 - x_4 \le 0$$
$$g_4(X) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0$$
$$g_5(X) = 0.125 - x_1 \le 0$$
$$g_6(X) = \delta(X) - \delta_{\max} \le 0$$
$$g_7(X) = P - P_c(X) \le 0$$

where

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \qquad \tau'' = \frac{MR}{J}, \qquad M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \qquad J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(X) = \frac{6PL}{x_4x_3^2}, \qquad \delta(X) = \frac{4PL^3}{Ex_3^3x_4}$$

$$P_c(X) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000\ lb, \qquad L = 14\ in, \qquad E = 30x10^6\ psi; \qquad G = 12x10^6\ psi$$

$$\tau_{\max} = 13600\ psi, \qquad \sigma_{\max} = 30000\ psi, \qquad \delta_{\max} = 0.25\ in$$

$$0.1 \le x_1 \le 2.0, \qquad 0.1 \le x_2 \le 10.0, \qquad 0.1 \le x_3 \le 10.0, \qquad 0.1 \le x_4 \le 2.0$$

RSPA's the best solution:
x=(0.205730, 3.470484, 9.036616, 0.205730)
$g_1(x)$=-0.000131989392684773
$g_2(x)$=-0.000005218651494943

$g_3(x)$=0.000000000000000000
$g_4(x)$=-3.432982819036312970
$g_5(x)$=-0.080729999999999996
$g_6(x)$=-0.235540309929741787
$g_7(x)$=-0.028063313488019048
f(x)=1.7248536095

- Statistical table of design of welded beam problem

Min	1.72485360948791
Max	1.72596817625674
Average	1.72519259083808
Median	1.72485875677079
Standard deviation	0.00047669537779

5.2.1.2. The version of T. Ray and K. M. Liew [5]

Minimize

$$f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

subject to

$$g_1(X) = \tau(X) - \tau_{\max} \le 0$$
$$g_2(X) = \sigma(X) - \sigma_{\max} \le 0$$
$$g_3(X) = x_1 - x_4 \le 0$$
$$g_4(X) = \delta(X) - \delta_{\max} \le 0$$
$$g_5(X) = P - P_c(X) \le 0$$

where

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \qquad \tau'' = \frac{MR}{J}, \qquad M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \qquad J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(X) = \frac{6PL}{x_4x_3^2}, \qquad \delta(X) = \frac{4PL^3}{Ex_3^3x_4},$$

$$P_c(X) = \frac{4.013\sqrt{\frac{EGx_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6000\ lb, \qquad L = 14\ in, \qquad E = 30x10^6\ psi, \qquad G = 12x10^6\ psi,$$

$$\tau_{\max} = 13600\ psi, \qquad \sigma_{\max} = 30000\ psi, \qquad \delta_{\max} = 0.25\ in,$$

$$0.125 \le x_1 \le 10.0, \qquad 0.1 \le x_2 \le 10.0, \qquad 0.1 \le x_3 \le 10.0, \qquad 0.1 \le x_4 \le 10.0.$$

RSPA's the best solutions:
x=(0.244369, 6.217520, 8.291471, 0.244369)
g1=-0.0012452783412300
g2=-0.0001450054878660
g3=0.0000000000000000
g4=-0.2342408342216855
g5=-0.0015862250211285
f(x)=2.3809568104489079

- Statistical table of design of welded beam problem

Min	2.38095681044890
Max	2.38592551620492
Average	2.38171454199839
Median	2.38096620360886
Standard deviation	0.00160938941422

5.2.2. Design of a Pressure Vessel [4]

Minimize

$$f(x) = 0.6224 * x_1 * x_3 * x_4 + 1.7781 * x_2 * x_3^2 + 3.1661 * x_1^2 * x_4 + 19.84 * x_1^2 * x_3$$

subject to

$$g_1(X) = -x_1 + 0.0193 * x_3 \le 0$$

$$g_2(X) = -x_2 + 0.00954 * x_3 \le 0$$

$$g_3(X) = -\Pi * x_3^2 * x_4 - \frac{4}{3} * \Pi * x_3^3 + 1296000 \le 0$$

$$g_4(X) = x_4 - 240.0 \le 0$$

where

$$1 \le x_1, x_2 \le 99$$

$$10 \le x_3, x_4 \le 200$$

We fixed x_1=0.8125, x_2=0.437, and found x_3, x_4. RSPA's the best solutions as follows:

- Select PI=3.1416

x=(0.8125000000, 0.4375000000, 42.0984455957, 176.6360515332)
g_1(x)=-0.00000000000299
g_2(x)=-0.03588082901702
g_3(x)=-0.00000039026872
g_4(x)=-63.36394846680003
f(x)=6059.70160945089356

- Select PI=3.1415926536

x=(0.8125000000, 0.4375000000, 42.0984455958, 176.6365958424)
g_1(x)=-0.00000000000106
g_2(x)=-0.03588082901607
g_3(x)=-0.000000202104500860
g_4(x)=-63.36340415760000
f(x)=6059.71433503829212

5.2.3. Minimization of the Weight of a Tension/Compression String [4][5]

Minimize

$$f(X) = (x_3 + 2)*x_2*x_1^2$$

subject to

$$g_1(X) = 1 - \frac{x_2^3*x_3}{71785*x_1^4} \le 0$$

$$g_2(X) = \frac{4*x_2^2 - x_1*x_2}{12566*(x_2*x_1^3 - x_1^4)} + \frac{1}{5108*x_1^2} - 1.0 \le 0$$

$$g_3(X) = 1 - \frac{140.45*x_1}{x_2^2*x_3} \le 0$$

$$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

$$0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15$$

RSPA's the best solution:
x=(0.051693, 0.356812, 11.283461)
g1(x)=-0.000000078635
g2(x)=-0.000001143622
g3(x)=-4.053965174479
g4(x)=-0.727663333333
f(x)=0.012665261791

- Statistical table of Minimization of the Weight of a Tension/Compression String problem

Min	0.012665261791
Max	0.012667032520
Average	0.012666001588
Median	0.012665459293
Standard deviation	0.000000769691

5.2.4. Minimization of the Weight of a Speed Reducer [4][5]

Minimize

$$f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

subject to

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$$

$$g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0$$

$$g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0;$$

$$g_5(x) = \frac{\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right]^{1/2}}{110.0x_6^3} - 1 \le 0$$

$$g_6(x) = \frac{\left[\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6\right]^{1/2}}{85.0x_7^3} - 1 \le 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 \le 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

where

$$2.6 \le x_1 \le 3.6, \quad 0.7 \le x_2 \le 0.8, \quad 17 \le x_3 \le 28, \quad 7.3 \le x_4 \le 8.3,$$
$$7.3 \le x_5 \le 8.3, \quad 2.9 \le x_6 \le 3.9, \quad 5.0 \le x_7 \le 5.5.$$

RSPA's the best solution:
x=(3.500000, 0.700000, 17.000000, 7.300000, 7.715321, 3.350215, 5.286655)
g_1(x)=-0.073915280397873318
g_2(x)=-0.197998527141949127
g_3(x)=-0.499172447764996807

$g_4(x)$=-0.904643902796802735
$g_5(x)$=-0.000000298998887224
$g_6(x)$=-0.000000303397127444
$g_7(x)$=-0.702500000000000013
$g_8(x)$=-0.000000000000000063
$g_9(x)$=-0.583333333333333259
$g_{10}(x)$=-0.051325684931506944
$g_{11}(x)$=-0.000000064806117507
f(x)=2994.4715149989115200

- Statistical table of minimization of the weight of a speed reducer problem.

Min	2994.471066162960
Max	2994.471066162960
Average	2994.471066162960
Median	2994.471066162960
Standard deviation	0.000057544497

6. CONCLUSIONS

In this paper, we proposed a new approach for single objective optimization problem, Random Search via Probability Algorithm (RSPA). RSPA use probabilities to guide search for an optimal solution. RSVA is based on essential remarks,

- The role of left digit is more important than the role of right digit for evaluating objective function. We calculated probabilities for searching correct values of digits from left digits to right digits of every variable.
- The complexity of RSPA of a problem is not based on type of expressions in objective function or constraints (linear or nonlinear), but on the relation of decided variables in the formula of object function or constraints; therefore if there is k dependent variables ($1 \le k \le n$), we select k variables to change the value of variables for every iteration.
- We can't calculate exactly a number of iterations for searching an optimal solution first because RSVA is a random algorithm; therefore we use unfixed number of iterations which has capability to find an optimal solution first with necessary number of iterations.

We tested this approach by implementing the RSPA on some test single objective optimization problems, and we found results very stable.

REFERENCES

[1] Coello, C. A. C., Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems, Preprint submitted to Elsevier Preprint, 30 may 2001.

[2] Deb, K., Joshi, D., and Anand, A., Real Coded Evolutionary Algorithms with Parent Centric Recombination, KanGAL Report Number 2001003, (2001).

[3] Dolan, A., A general GA toolkit implemented in Java, for experimenting with genetic algorithms and handling optimization problems, http://www.aridolan.com/ofiles/ga/gaa/gaa.html.

[4] Mezura-Montes E., Coello C.A.C., On the Usefulness of the Evolution Strategies' Self-Adaptation Mechanism to Handle Constraints in Global Optimization, Technical Report EVOCINV-01-2003, CINVESTAV-IPN, México, January 2003.

[5] Ray, T. and Liew, K. M., Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behaviour, IEEE Trans. On Evolutionary Computing, Vol 7(4), pp. 386-396, (2003).

Appendix 1

The program illustrates RSPA for Rastrigin function (using Borland C 3.1)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
int const n=30;
double  inf[n],sup[n];int seg[n];
void init()
{int i;
 for (i=0;i<n;i++) {inf[i]=-100.00;  sup[i]=100.00;}
 for (i=0;i<n;i++)  seg[i]=1+(int)(sup[i]-inf[i]);}
void object(double x[],double &f)
{int i; double u=0;
 for (i=0;i<n;i++) u+=x[i]*x[i]-10.0*cos(2.0*3.1416*x[i]);
 f=10.0*n+u;}
void random_select(double x[])
{int i;
 for (i=0;i<n;i++)
  {x[i]=inf[i]+random(seg[i])+random(10000)*0.0001+random(100)*0.000001;
   if (x[i]<inf[i]) x[i]=inf[i]; if (x[i]>sup[i]) x[i]=sup[i];}}
void Change(double const x[],double y[])
{int t,i,a,d,d1,d2,r,r1,a0,a1,a2,a3,a4,a5,p[6],fix; double z,fy;
 d1=50; d2=75; r1=4+random(10);
 if (random(100)<30) {p[0]=57; p[1]=64;p[2]=71; p[3]=79; p[4]=86; p[5]=93;}
 else {p[0]=37; p[1]=41; p[2]=46; p[3]=52; p[4]=61;  p[5]=75;}
 if (random(100)<50) {fix=random(3)+3; for (i=0;i<fix;i++) p[i]=0;}
 for (i=0;i<n;i++)
  {if (random(100)<r1)
    {z=x[i];
     a0=(int) z; z=z-a0; z=10*z;a1=(int) z; z=z-a1; z=10*z;
     a2=(int) z; z=z-a2; z=10*z;a3=(int) z; z=z-a3; z=10*z;
     a4=(int) z; z=z-a4; z=10*z;a5=(int) z;
a=3; if (random(100)<p[0])
      {r=random(100);
       if (r<d1) y[i]=inf[i]+random(seg[i]);
       else if (r<d2) y[i]=a0-random(a);
          else y[i]=a0+random(a);}
     else y[i]=a0;
```

```
     if (random(100)<p[1])
      {r=random(100);
       if (r<d1)  y[i]=y[i]+0.1*random(10);
       else if (r<d2) y[i]=y[i]+0.1*(a1-random(a));
          else y[i]=y[i]+0.1*(a1+random(a));}
     else y[i]=y[i]+0.1*a1;
     if (random(100)<p[2])
      {r=random(100);
       if (r<d1) y[i]=y[i]+0.01*random(10);
       else if (r<d2) y[i]=y[i]+0.01*(a2-random(a));
          else y[i]=y[i]+0.01*(a2+random(a));}
     else y[i]=y[i]+0.01*a2;
a=5; if (random(100)<p[3])
      {r=random(100);
       if (r<d1)  y[i]=y[i]+0.001*random(10);
       else if (r<d2) y[i]=y[i]+0.001*(a3-random(a));
          else y[i]=y[i]+0.001*(a3+random(a)); }
     else y[i]=y[i]+0.001*a3;
     if (random(100)<p[4])
      {r=random(100);
       if (r<d1) y[i]=y[i]+0.0001*random(10);
       else if (r<d2) y[i]=y[i]+0.0001*(a4-random(a));
          else y[i]=y[i]+0.0001*(a4+random(a));}
     else y[i]=y[i]+0.0001*a4;
     if (random(100)<p[5])
      {r=random(100);
       if (r<d1) y[i]=y[i]+0.00001*random(10);
       else if (r<d2) y[i]=y[i]+0.00001*(a5-random(a));
          else y[i]=y[i]+0.00001*(a5+random(a)); }
     else y[i]=y[i]+0.00001*a5;
     y[i]=y[i]+0.000001*random(10);
   }
  else y[i]=x[i];
  if (y[i]<inf[i]) y[i]=inf[i];
  if (y[i]>sup[i]) y[i]=sup[i];
 }
}
void Print(double x[],double fx)
{
 int i;
 for (i=0;i<n;i++) {printf("\nx[%d]=%8.6lf",i,x[i]); clreol();}
 printf("\nf=%18.16lf",fx); clreol();
}
void copy(double const y[],double const fy,double x[],double &fx)
{
 int i;
 for(i=0;i<n;i++) x[i]=y[i]; fx=fy;
}
void Search()
{
 double loop=0,x[n],fx,y[n],fy;int i,t;
 randomize(); random_select(x); object(x,fx);
 gotoxy(1,1); Print(x,fx);
 do
 {
  Change(x,y); object(y,fy);loop++;
  gotoxy(1,34); printf("\nLoop=%.0lf",loop);clreol();
  if (fy<fx) {copy(y,fy,x,fx); gotoxy(1,1); Print(x,fx); loop=0;}
 } while (loop<50000);
```

```
}
void main()
{clrscr(); init(); Search(); getch();}
```
Remark: Select a=2 for sphere function because the value 0.99 can be found very quickly and the probability of event 0.99+0.01=1.00 can be happen highly.

Appendix 2

The program illustrates RSPA for Minimization of the Weight of a Speed Reducer problem (using Borland C 3.1)

```
//Speed Reducer Design
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
const int n=7,m=11;
double  inf[n],sup[n];
int seg[n];
void init()
{
 int i;
 inf[0]=2.6;  sup[0]=3.6;
 inf[1]=0.7;  sup[1]=0.8;
 inf[2]=17.0; sup[2]=28.0;
 inf[3]=7.3;  sup[3]=8.3;
 inf[4]=7.3;  sup[4]=8.3;
 inf[5]=2.9;  sup[5]=3.9;
 inf[6]=5.0;  sup[6]=5.5;
 for (i=0;i<n;i++)
   seg[i]=1+(int)(sup[i]-inf[i]);
}
void constraint(const double x[],double g[])
{
 g[0]=(27.0/(x[0]*x[1]*x[1]*x[2]))-1.0;
 g[1]=(397.5/(x[0]*x[1]*x[1]*x[2]*x[2]))-1.0;
 g[2]=(1.93*x[3]*x[3]*x[3]/(x[1]*x[2]*pow(x[5],4)))-1.0;
 g[3]=(1.93*pow(x[4],3)/(x[1]*x[2]*pow(x[6],4)))-1.0;
 g[4]=(sqrt(pow(745.0*x[3]/(x[1]*x[2]),2)+16.9*1000000.0)/(110.0*pow(x[5],3)))-1.0;
 g[5]=(sqrt(pow(745.0*x[4]/(x[1]*x[2]),2)+157.5*1000000.0)/(85.0*pow(x[6],3)))-1.0;
 g[6]=(x[1]*x[2]/40.0)-1.0;
 g[7]=(5.0*x[1]/x[0])-1.0;
 g[8]=(x[0]/(12.0*x[1]))-1.0;
 g[9]=((1.5*x[5]+1.9)/x[3])-1.0;
 g[10]=((1.1*x[6]+1.9)/x[4])-1.0;
}
void object(double const x[],double &f)
{
 f=0.7854*x[0]*x[1]*x[1]*(3.3333*x[2]*x[2]+14.9334*x[2]-43.0934)-
1.508*x[0]*(x[5]*x[5]+x[6]*x[6])+7.4777*(x[5]*x[5]*x[5]+x[6]*x[6]*x[6])+0.7854*(x[3
]*x[5]*x[5]+x[4]*x[6]*x[6]);
}
void random_select(double x[], double g[])
{
 int i,t;
 do
  {
   for (i=0;i<n;i++)
    {
```

```
     x[i]=inf[i]+random(seg[i])+random(10000)*0.0001+random(10000)*0.00000001;
     if (x[i]<inf[i]) x[i]=inf[i];
     if (x[i]>sup[i]) x[i]=sup[i];
    }
   constraint(x,g);
   t=0;
   for (i=0;i<m;i++)
     if (g[i]>0)t=1;
  } while (t);
}
void change(double const x[],double y[],double g[])
{
 double  z,t;
 int   p[6],fix,a,d1,d2,r,r1,i,a0,a1,a2,a3,a4,a5;
 if (random(100)<random(30)+20)
  {p[0]=57; p[1]=64; p[2]=71; p[3]=79; p[4]=86; p[5]=93;}
 else
  {p[0]=37; p[1]=41; p[2]=46; p[3]=52; p[4]=61; p[5]=75;}
 if (random(100)<50 )
   {fix=random(3);
    for (i=0;i<fix;i++)  p[i]=0;}
 d1=50;  d2=75;
 r1=random(50)+30;
 do
   {
    for (i=0;i<n;i++)
    {
     if (random(100)<r1)
      {
      z=x[i];
      a0=(int) z; z=z-a0; z=10*z;
      a1=(int) z; z=z-a1; z=10*z;
      a2=(int) z; z=z-a2; z=10*z;
      a3=(int) z; z=z-a3; z=10*z;
      a4=(int) z; z=z-a4; z=10*z;
      a5=(int) z;
a=3;
      if (random(100)<p[0])
        {
          r=random(100);
          if (r<d1)
              y[i]=inf[i]+random(seg[i]);
          else
            if (r<d2)
             y[i]=a0-random(a);
            else
            y[i]=a0+random(a);
        }
      else
          y[i]=a0;

      if (random(100)<p[1])

        {
          r=random(100);
          if (r<d1)
             y[i]=y[i]+0.1*random(10);
          else
            if (r<d2)
```

```
            y[i]=y[i]+0.1*(a1-random(a));
           else
           y[i]=y[i]+0.1*(a1+random(a));
     }
   else
       y[i]=y[i]+0.1*a1;

   if (random(100)<p[2])
     {

       r=random(100);
       if (r<d1)
          y[i]=y[i]+0.01*random(10);
       else
         if (r<d2)
          y[i]=y[i]+0.01*(a2-random(a));
         else
         y[i]=y[i]+0.01*(a2+random(a));
     }
   else
       y[i]=y[i]+0.01*a2;

a=4;
   if (random(100)<p[3])
     {
       r=random(100);
       if (r<d1)
          y[i]=y[i]+0.001*random(10);
       else
         if (r<d2)
          y[i]=y[i]+0.001*(a3-random(a));
         else
         y[i]=y[i]+0.001*(a3+random(a));
     }
   else
       y[i]=y[i]+0.001*a3;

   if (random(100)<p[4])
     {
       r=random(100);
       if (r<d1)
         y[i]=y[i]+0.0001*random(10);
       else
         if (r<d2)
          y[i]=y[i]+0.0001*(a4-random(a));
         else
         y[i]=y[i]+0.0001*(a4+random(a));
     }
   else
       y[i]=y[i]+0.0001*a4;

   if (random(100)<p[5])
     {
       r=random(100);
       if (r<d1)
         y[i]=y[i]+0.00001*random(10);
       else
         if (r<d2)
          y[i]=y[i]+0.00001*(a5-random(a));
```

```
            else
            y[i]=y[i]+0.00001*(a5+random(a));
         }
       else
           y[i]=y[i]+0.00001*a5;

       y[i]=y[i]+0.000001*random(10);
      }
     else
       y[i]=x[i];
     if (y[i]<inf[i]) y[i]=inf[i];
     if (y[i]>sup[i]) y[i]=sup[i];
    }
    constraint(y,g);
    t=0;
    for (i=0;i<m;i++)
        if (g[i]>0) t=1;
  } while (t);
}
void print(double x[],double gx[], double fx)
{
 int i;
 for (i=0;i<n;i++) printf("x[%d]=%9.6lf\n",i,x[i]);
 for (i=0;i<m;i++)  printf("\ng[%d]=%20.18lf",i,gx[i]);
 printf("\nf=%20.18lf",fx);
}
void search()
{
 double  x[n],y[n],gx[m],gy[m],fx,fy,loop;
 int i;
 randomize;
 random_select(x,gx);
 object(x,fx);
 loop=0;
 do
 {
  change(x,y,gy);
  object(y,fy);
  if (fy<fx)
    {
      fx=fy;
      for (i=0;i<n;i++)   x[i]=y[i];
      for (i=0;i<m;i++)  gx[i]=gy[i];
      gotoxy(1,1);
      print(x,gx,fx);
      loop=0;
    }
  loop++;
  gotoxy(1,22);
  printf("loop=%.0f",loop);
 } while (loop<100000);
}
void main()
{
 clrscr();
 init();
 search();
 getch();
}
```