A New Stochastic Algorithm for Engineering Optimization Problems

Thong Nguyen Huu, Hao Tran Van
Mathematics-Informatics department, University of Pedagogy
280, An Duong Vuong, Ho Chi Minh city, Viet Nam

ABSTRACT

This paper proposes a new stochastic algorithm, Search via Probability (SP) algorithm, for single-objective optimization problems. The SP algorithm uses probabilities to control the process of searching for optimal solutions. We calculate probabilities of the appearance of a better solution than the current one on each iteration, and on the performance of SP algorithm we create good conditions for its appearance. We test this approach by implementing the SP algorithm on some test engineering optimization problems, and we find very stable results.

Keywords: Numerical Optimization, Stochastic, Random, Probability, Algorithm.

1. INTRODUCTION

There are many algorithms, traditional computation or evolutionary computation, for single-objective optimization problems. Almost all focus on the determination of positions neighbouring an optimal solution and handle constraints based on violated constraints. We can suppose that every decided variable of an optimization problem has digits that are listed from left to right, we have our remarks as follows: To evaluate objective function, the role of left digits is more important than the role of right digits of a decided variable; We calculate the changing probabilities of the appearance of a better solution than the current one on each iteration, and on the performance of SP algorithm, we create good conditions for its appearance. Based on the relation of decided variables in the formulas of constrains and objective function we select k variables ($1 \le k \le n$) to change their values instead of selecting all n variables on each iteration. Because we can not calculate exactly the number of iterations of a stochastic algorithm for searching an optimal solution the first time on each performance, we use unfixed number of iterations, which has more chance to find an optimal solution the first time with necessary number of iterations. Based on these remarks we introduce a new stochastic algorithm, Search via Probability (SP) algorithm, the SP algorithm uses probabilities to control the process of searching for optimal solutions.

2. THE MODEL OF SINGLE-OBJECTIVE OPTIMIZATION PROBLEM

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

$$\begin{aligned} &\textit{Minimize} && f(x) \\ &\textit{subject to} && g_j(x) \le 0 \quad (j = 1, \ldots, r) \\ &\textit{where} && a_i \le x_i \le b_i,\ a_i, b_i \in R,\ i = 1, \ldots, n. \end{aligned}$$

We can suppose that every decided variable x_i ($1 \le i \le n$) of a solution of an optimization problem has m digits that are listed from left to right $x_{i1}, x_{i2}, \ldots, x_{im}$ (x_{ij} is an integer and $0 \le x_{ij} \le 9$, $1 \le j \le m$). We have our remarks as follows: To evaluate objective function, the role of left digits is more important than the role of right digits of a decided variable; we calculate changing probabilities of digits which can find better values than the current ones on each iteration.

3. PROBABILITIES OF CHANGES AND SELECTING VALUES OF A DIGIT

3.1. Probabilities of changes

Consider the j-th digit x_{ij} of variable x_i, let A_j be an event that the value of digit x_{ij} can be changed ($1 \le i \le n,\ 1 \le j \le m$). Event A_j is more important than event A_{j+1}, it means that the occurrence of event A_j has a decisive influence on the occurrence of event A_{j+1}, and after event A_j occurs a certain number of times, it will create good conditions for occurrences of event A_{j+1}. Let q_j be probability of A_j, r_j be number of occurrences of event:

$$\overline{A_1}\,\overline{A_2}\ldots\overline{A_{j-1}}A_j \quad (1 \le j \le m)$$

We consider event:

$$(A_1)^{r_1}\left(\overline{A_1}A_2\right)^{r_2}\left(\overline{A_1}\,\overline{A_2}A_3\right)^{r_3}\ldots\left(\overline{A_1}\,\overline{A_2}\ldots\overline{A_{m-1}}A_m\right)^{r_m}$$

Because $A_1, A_2, \ldots, A_m$ are independent of one another, the probability of this event is:

$$(q_1)^{r_1}(1-q_1)^{r_2+r_3+\ldots+r_m}(q_2)^{r_2}(1-q_2)^{r_3+r_4+\ldots+r_m}\ldots(q_{m-1})^{r_{m-1}}(1-q_{m-1})^{r_m}(q_m)^{r_m}$$

and this probability is maximum if

$$q_j = \frac{r_j}{r_j + r_{j+1} + \ldots + r_m} \quad (1 \le j \le m)$$

Because left events are more important than right events, it means that left events are more stable than right events, we have to have:

$$r_1 \le r_2 \le \ldots \le r_m$$

Therefore:

$$q_1 \le q_2 \le \ldots \le q_m$$

and

$$\frac{r_j}{r_j + (m-1)r_m} \le q_j \le \frac{1}{m+1-j} \quad (1 \le j \le m)$$

The changing probabilities of digits of a variable increase from left to right. This means that left digits are more stable than right digits, and right digits change more than left digits. In other words, 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.

3.2. Probabilities for selecting values of a digit.

Consider j-th digit with changing probability q_j ($1 \le j \le m$), let R_1 be the probability of choosing a random integer number between 0 and 9 for j-th digit, let R_2 be probabilities of j-th digit incremented by one or a certain value, let R_3 be probabilities of j-th digit decremented by one or a certain value. Now we consider two digits a_{j-1} and a_j, with two probabilities $(1-q_{j-1})$ and q_j ($2 \le j \le m$). We have two cases:

- Case 1: If the value of a_{j-1} is not worse than the previous one, we have the probability so that a_j can find a better value than the current one 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 maximum if $R_1=1$, $R_2=R_3=0$.

- Case 2: If the value of a_{j-1} is worse than the previous one, we have the probability so that a_{j-1} and a_j can find better values than the current ones 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 maximum if $R_1=0$, $R_2=R_3=0.5$.

The average probabilities of R_1, R_2 and R_3 of both two cases: $R_1=0.5$, $R_2=R_3=0.25$

4. SELECTING K VARIABLES (1≤K≤N) TO CHANGE THEIR VALUES

On each iteration, if we select n variables to change their values, the ability of finding a better solution than the current one may be very small. Therefore we select k variables (1≤k≤n) to change their values, and after a number of iterations the algorithm has more chance to find a better solution than the current one. Let A_j be an event that variable x_i can find a better value than the current one (1≤i≤n). On each iteration, we select randomly k events (1≤k≤n) and consider product of these k selected events. Probability of selecting an event is k/n. Let A be the product event of these k selected events, the probability of A is:

$$\left(\frac{k}{n}q_{i_1}\right)\left(\frac{k}{n}q_{i_2}\right)\ldots\left(\frac{k}{n}q_{i_k}\right)=\left(\frac{k}{n}\right)^k q_{i_1}q_{i_2}\ldots q_{i_k}$$
$$where\ 1\le i_1<i_2<\ldots i_k\le n.$$

We consider the worst case, let

$$q=\min\left\{q_{i_1}q_{i_2}\ldots q_{i_k}\,|\,1\le i_1<i_2<\ldots i_k\le n\right\}$$

and

$$p=(k/n)^k q^k.$$

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

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

Probability for event A to occur at least once:

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

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

$$P(X>0)\ge\alpha\Rightarrow 1-(1-p)^d\ge\alpha$$
$$\Rightarrow(1-p)^d\le 1-\alpha\Rightarrow d\ln(1-p)\le\ln(1-\alpha)$$
$$\Rightarrow d\ge\frac{\ln(1-\alpha)}{\ln(1-p)}\Rightarrow d\ge\frac{\ln(1-\alpha)}{\ln\left(1-\left(\frac{k}{n}\right)^k q^k\right)}$$

Select a minimum number of iterations and when n→+∞, we have:

$$\frac{\ln(1-\alpha)}{\ln\left(1-\left(\frac{k}{n}\right)^k q^k\right)}\approx-\frac{\ln(1-\alpha)}{\left(\frac{k}{n}\right)^k q^k}=-\frac{\ln(1-\alpha)}{k^k q^k}n^k=Cn^k$$

If k is a fix number and independent from n, the complexity for finding a better solution than the current one on each iteration is $O(n^k)$. According to statistics of many experiments, the best thing is to use k in the ratio 20%-80% of n.
If k=n,

$$\frac{\ln(1-\alpha)}{\ln\left(1-\left(\frac{k}{n}\right)^k q^k\right)}=\frac{\ln(1-\alpha)}{\ln\left(1-\left(\frac{n}{n}\right)^n q^n\right)}=\frac{\ln(1-\alpha)}{\ln\left(1-q^n\right)}$$
$$\approx-\frac{\ln(1-\alpha)}{q^n}=-\ln(1-\alpha)\left(\frac{1}{q}\right)^n=Ca^n\ \left(a=\frac{1}{q}>1\right)$$

the complexity for finding a better solution than the current one on each iteration is $O(a^n)$.

Ex: With k=1, on each iteration we only need to select one variable, it is sufficient for finding a better solution than the current one. The problems of this type have the following similar form:

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

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

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

5. THE RANDOM SEARCH VIA PROBABILITY ALGORITHM

The main idea of SP algorithm is that variables of problem are separated into discrete digits, and then they are changed with the guide of probabilities and combined to a new solution. We suppose that a solution of problem has n variables, every variable has m=7 digits that are listed from left to right $x_{i1}, x_{i2},\ldots, x_{im}$ (x_{ij} is an integer and $0 \le x_{ij} \le 9$, j=1, …,m). M is a number of inside iterations of an outside iteration. SP algorithm is described with general steps as follows:

S1. Select a random feasible solution x. Let Fx=f(x). Loop=0.
S2. We apply the procedure to generate a new solution y.
S3. If y is an infeasible solution then return S2.
S4. Let Fy=f(y). If Fy <Fx then x=y; Fx=Fy; loop=0;
S5. If loop<M then loop=loop+1, return S2.
S6. End of SP algorithm.

The procedure of generating a new solution as follows:

S2.1. We create changing probabilities q_j ($1 \le j \le m$) and $q_1 \le q_2 \le \ldots \le q_m$.

S2.2. The technique for changing value via probability to create a new solution $y=(y_1, y_2, \ldots, y_n)$ is described as follows:

```
k=20+random(80);
For i=1 to n do
 Begin_1
  If (random(100)<k) then  {select k variables }
   Begin_2  {variable y_i is selected for changing its value}
    y_i=0;
    For j=1 to m-1 do
     Begin_3
      If j=1, 2, 3 then a=2 or 3 else a=4 or 5; (*)
      If (probability of a random event is q_j) then
         If (probability of a random event is R1) then y_ij= y_ij+10^(1-j)*random(10);
         else   if (probability of a random event is R_2) then  y_ij= y_ij+10^(1-j)*( x_ij –random(a));
                else y_ij= y_ij +10^(1-j)*( x_ij +random(a));
      else y_ij= y_ij +10^(1-j)* x_ij;
     End_3;
   y_im= y_im +10^(-m) *random(10);
  End_2;
 Else y_i=x_i;  {variable y_i is not selected for changing its value}
 if (y_i<ai) then y_i=a_i;   if (y_i >b_i) then y_i=b_i;
End_1;
```

The SP algorithm has the following characteristics:

- The SP algorithm finds the value of each digit from left digit to right digit of every variable with the guide of probabilities, and the newly-found value may be better than the current one.
- Variable Loop will be set to 0 if SP algorithm finds a better solution than the current one; this means that SP algorithm can find an optimal solution the first time after a necessary number of iterations.

- (*): Where a=2,3 if j=1,2,3, and a=4,5 if j=4,5,6,7 because the right digits vary more than the left digits.

6. EXAMPLES

Using PC, Celeron CPU 2.20GHz, Borland C++ 3.1. Select value to parameter M=100000. We performed 30 independent runs for each example. The results for all test problems are reported in Tables.

6.1. Design of a Welded Beam

6.1.1. 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

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \sigma(X) = \frac{6PL}{x_4x_3^2}, \delta(X) = \frac{4PL^3}{Ex_3^3x_4}, P_c(X) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$P = 6000lb, L = 14in, E = 30x10^6 psi; G = 12x10^6 psi, \tau_{\max} = 13600\, psi, \sigma_{\max} = 30000\, psi, \delta_{\max} = 0.25in,$

$0.1 \le x_1 \le 2, 0.1 \le x_2 \le 10, 0.1 \le x_3 \le 10, 0.1 \le x_4 \le 2.$

For solving this problem, Coello used a Self-Adaptive Penalty Approach for Engineering Optimization Problems, Mezura-Montes et al. used a Simple Evolutionary Algorithm.

Table 1: Comparison of results for Welded Beam Design (The version of Coello [4]).

	SP alg.	**Coello [4]**	**Mezura-Montes et al.[6]**
x_1	0.205730	0.2088	0.203642
x_2	3.470484	3.4205	3.593228
x_3	9.036616	8.9975	8.980190
x_4	0.205730	0.2100	0.208393
$g_1(x)$	-0.000131	-0.337812	-165.394302
$g_2(x)$	-0.000005	-353.902604	-10.008150
$g_3(x)$	0.000000	-0.00120	-0.004751
$g_4(x)$	-3.432982	-3.411865	-3.411678
$g_5(x)$	-0.080729	-0.08380	-0.078642
$g_6(x)$	-0.235540	-0.235649	-0.235454
$g_7(x)$	-0.028063	-363.232384	-210.369049
f(x)	1.724853	1.74830941	1.748594

- Statistical table of 30 independent runs of SP algorithm for Welded Beam Design (The version of Coello).

Min	1.72485360948791
Max	1.72596817625674
Average	1.72519259083808
Median	1.72485875677079
St. Dev.	0.00047669537779

6.1.2. The version of T. Ray et al. [15]

Minimize

$$f(X) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

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

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2},\ \tau' = \frac{P}{\sqrt{2} x_1 x_2}, \tau'' = \frac{MR}{J},\ M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, J = 2\left\{\frac{x_1 x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\ \sigma(X) = \frac{6PL}{x_4 x_3^2},$$

$$\delta(X) = \frac{4PL^3}{E x_3^3 x_4},\ P_c(X) = \frac{4.013\sqrt{\frac{EGx_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6000\, lb, L = 14\, in, E = 30x10^6\ \ psi, G = 12x10^6\ psi,\ \tau_{\max} = 13600\, psi,$$

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

$$0.125 \le x_1 \le 10, 0.1 \le x_2 \le 10, 0.1 \le x_3 \le 10, 0.1 \le x_4 \le 10.$$

For solving this problem, Sidall used an analytical decision-making in engineering design, Ray et al. used an optimization algorithm based on the simulation of social behaviour, Ragsdell et al. used an optimal design of a class of welded structures using geometric programming, Akhtar et al. used a socio-behavioural simulation model for engineering design optimization, Deb used an optimal design of a welded beam via genetic algorithms.

Table 2: Comparison of results for Welded Beam Design (The version of Ray et al. [15])

	SP algorithm	**Siddall [8]**	**Ray et al. [15]**	**Ragsdell et al. [10]**
x_1	0.244369	0.2444	0.2444382760	0.2455
x_2	6.217520	6.2819	6.2379672340	6.1960
x_3	8.291471	8.2915	8.2885761430	8.2730
x_4	0.244369	0.2444	0.2445661820	0.2455
$g_1(x)$	-0.0012452783412300		-32.4102447520399437	
$g_2(x)$	-0.0001450054878660		-3.2454275687741756	
$g_3(x)$	0.0000000000000000		-0.0001279060000000	
$g_4(x)$	-0.2342408342216855		-0.2342370355568332	
$g_5(x)$	-0.0015862250211285		-13.0793049586818597	
f(x)	2.3809568104489079	2.3815	2.3854347515606569	2.3859

	Akhtar et al. [14]	**Deb [12]**
x_1	0.2407	0.2489
x_2	6.4851	6.1730
x_3	8.2399	8.1789
x_4	0.2497	0.2533
$g_1(x)$		
$g_2(x)$		
$g_3(x)$		
$g_4(x)$		

$g_5(x)$		
f(x)	2.4426	2.4431

- Statistical table of 30 independent runs of SP algorithm for Welded Beam Design (The version of Ray et al. [15])

Min	2.38095681044890
Max	2.38592551620492
Average	2.38171454199839
Median	2.38096620360886
St. Dev.	0.00160938941422

6.2. Design of a Pressure Vessel

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

$$1 \le x_1, x_2 \le 99,\ 10 \le x_3, x_4 \le 200$$

x_1 and x_2 are integer multiples of 0.0625 inch.

For solving this problem, Mezura-Montes et al. used a simple evolutionary algorithm, Akhtar et al. used a socio-behavioral approach, Coello used a self-adaptive penalty approach for engineering optimization problems, Deb used a robust optimal design technique for mechanical component design (GeneAS), Kannan et al. used an augmented Langrangian multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, Sandgren used a nonlinear integer and discrete programming in mechanical design.

Table 3: Comparison of results for Pressure Vessel Design.

	Montes et al. [6] (PI=3.1416)	**Akhtar et al. [14]**	**Coello [4]**	**Deb [11]**	**Kannan et al. [3]**	**Sandgren [7]**
x_1	0.812500	0.8125	0.8125	0.9375	1.125	1.125
x_2	0.437500	0.4375	0.4375	0.5000	0.625	0.625
x_3	42.098370	41.9768	40.3239	48.3290	58.291	47.700
x_4	176.637146	182.2845	200.0000	112.6709	43.690	117.701
$g_1(x)$	-0.000001	-0.0023	-0.034324	-0.004750	0.000016	-0.204390
$g_2(x)$	-0.035882	-0.0370	-0.052847	-0.038941	-0.068904	-0.169942
$g_3(x)$	-0.878122	-23420.5966	-27.105845	-3652.8768	-21.220104	54.226012
$g_4(x)$	-63.362858	-57.7155	-40.00000	-127.3210	-196.3100	-122.299000
f(x)	6059.714355	6171.0	6288.7445	6410.3811	7198.0428	8129.1036

SP algorithm's the best solutions for Pressure Vessel Design 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

$g_3(x)$=-0.00000020210450, $g_4(x)$=-63.36340415760000
f(x)=6059.71433503829212

6.3. Minimization of the Weight of a Tension/Compression String

Minimize

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

subject to

$$g_1(x) = 1 - \frac{x_2^3*x_3}{71785*x_1^4} \le 0,\ g_2(x) = \frac{4*x_2^2 - x_1*x_2}{12566*(x_2*x_1^3 - x_1^4)} + \frac{1}{5108*x_1^2} - 1.0 \le 0,$$

$$g_3(x) = 1 - \frac{140.45*x_1}{x_2^2*x_3} \le 0,\ g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

where

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

For solving this problem, Ray et al. used an optimization algorithm based on the simulation of social behaviour, Mezura-Montes et al. used a simple evolutionary algorithm, Coello used a self-adaptive penalty approach for engineering optimization problems, Arora used a numerical optimization technique called Constraint Correction at constant Cost (CCC), Belegundu used eight numerical optimization techniques (CONMIN, OPTDYN, LINMR, GRP-UI, SUMT, M-3, M4, and M-5).

Table 4: Comparison of results for Tension/Compression String Design.

	SP alg.	**Ray et al. [15]**	**Mezura-Montes et al. [6]**
x_1	0.051693	0.052160217	……..
x_2	0.356812	0.368158695	
x_3	11.283461	10.648442259	
$g_1(x)$	-0.000000078635	-0.00000000745274	
$g_2(x)$	-0.000001143622	-0.00000000368096	
$g_3(x)$	-4.053965174479	-4.07580499710799	
$g_4(x)$	-0.727663333333	-0.71978739200000	
f(x)	0.012665261791	0.01266924933881	0.012688

	Coello [4]	**Arora [9]**	**Belegundu [1]**
x_1	0.051480	0.053396	0.050000
x_2	0.351661	0.399180	0.315900
x_3	11.632201	9.185400	14.25000
$g_1(x)$	-0.002080	0.000019	-0.000014
$g_2(x)$	-0.000110	-0.000018	-0.003782
$g_3(x)$	-4.026318	-4.123832	-3.938302
$g_4(x)$	-4.026318	-0.698283	-0.756067
f(x)	0.0127047834	0.0127302737	0.0128334375

- Statistical table of 30 independent runs of SP algorithm for Tension/Compression String Design.

Min	0.012665261791
Max	0.012667032520
Average	0.012666001588
Median	0.012665459293
St. deviation	0.000000769691

6.4. Minimization of the Weight of a Speed Reducer

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

For solving this problem, Ray et al. used an optimization algorithm based on the simulation of social behaviour, Mezura-Montes et al. used a simple evolutionary algorithm, Akhtat et al. used a socio-behavioural simulation model for engineering design optimization.

Table 5: Comparison of results for Speed Reducer Design.

	SP alg.	**Ray et al. [15]**	**Mezura-Montes et al. [6]**	**Akhtar et al. [14]**
x_1	3.500000	3.50000681	3.506163	3.503122
x_2	0.700000	0.70000001	0.700831	0.700006
x_3	17.000000	17.00000000	17	17
x_4	7.300000	7.32760205	7.460181	7.549126
x_5	7.715321	7.71532175	7.962143	7.859330
x_6	3.350215	3.35026702	3.362900	3.365576
x_7	5.286655	5.28665450	5.308949	5.289773
$g_1(x)$	-0.073915280397873318	-0.07391711	-0.077734	-0.075548
$g_2(x)$	-0.197998527141949127	-0.19800011	-0.201305	-0.199413
$g_3(x)$	-0.499172447764996807	-0.49350137	-0.474119	-0.456175
$g_4(x)$	-0.904643902796802735	-0.90464384	-0.897068	-0.899442
$g_5(x)$	-0.000000298998887224	-0.00000064	-0.011021	-0.013213
$g_6(x)$	-0.000000303397127444	-0.00000002	-0.012500	-0.001740
$g_7(x)$	-0.702500000000000013	-0.70250000	-0.702147	-0.702497
$g_8(x)$	-0.000000000000000063	-0.00000193	-0.000573	-0.001738
$g_9(x)$	-0.583333333333333259	-0.58333253	-0.583095	-0.582608
$g_{10}(x)$	-0.051325684931506944	-0.05488856	-0.069144	-0.079580
$g_{11}(x)$	-0.000000064806117507	-0.00000023	-0.027920	-0.017887
f(x)	2994.4715149989115200	2994.73051820	3025.005127	3008.08
		2994.744241(*)	3025.00569(*)	

(*) These statistics were checked by using our test program.

- Statistical table of 30 independent runs of SP algorithm for Speed Reducer Design.

Min	2994.471066162960
Max	2994.471066162960
Average	2994.471066162960
Median	2994.471066162960
St. deviation	0.000000000000

7. CONCLUSIONS

In this paper, we proposed a new approach for single-objective optimization problems, Search via Probability algorithm. The SP algorithm used probabilities to control the process of searching for an optimal solution. We calculated the probabilities of the appearance of a better solution than the current one, and on each iteration of the performance of SP algorithm, we created good conditions for its appearance. The idea of SP algorithm was based on essential remarks as follows:

- The role of left digits was more important than the role of right digit for evaluating objective function. We calculated probabilities for searching better values than the current ones of digits from left digits to right digits of every variable. Decided variables of problem were separated into discrete digits, and then they were changed with the guide of probabilities and combined to a new solution.
- The complexity of SP algorithm of a problem was not based on the type of expressions in the objective function or constraints (linear or nonlinear), but on the relation of decided variables in the formulas of object function or constraints; therefore if there were k independent variables ($1 \le k \le n$), it would be sufficient to find a better solution than the current one, we only needed to select k variables to change their values on each iteration.
- We could not calculate exactly a number of iterations for searching an optimal solution the first time because SP algorithm was a stochastic algorithm; therefore we used unfixed number of iterations which has more chance to find an optimal solution the first time with necessary number of iterations.

We tested this approach by implementing the SP algorithm on some test single-objective optimization problems, and we found very stable results. However if the feasible region of problem was very small or narrow, SP algorithm could not generate a random feasible solution or it found slowly the values of right digits of variables. We were researching these drawbacks with many various approaches of SP algorithm. We were also applying SP algorithm for solving multiobjective optimization and discrete optimization problems.

REFERENCES

[1] A. D. Belegundu, "A Study of Mathematical Programming Methods for Structural Optimization," *Dept. of civil and environmental engineering*, University of Iowa, Iowa, 1982.

[2] A. Homaifar, S. H. Y. Lai and X. Qui, "Constrained Optimization via Genetic Algorithms," *Simulation*, 62(4):242-254, 1994.

[3] B. K. Kannan and S. N. Kramer, "An augmented Langrangian multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design," *ASME Journal of Mechanical Design*, 116, 318–320, 1994.

[4] C. A. C. Coello, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems," *Computers in Industry, Elsevier Science*, Vol. 41, No. 2, pp. 113-127, January 2000.

[5] D. M. Himmelblau, "Applied Nonlinear Programming,", *McGraw-Hill, New York*, [4.1], 1972.

[6] E. Mezura-Montes, C. A. C. Coello, and R. Landa-Becerra, "Engineering Optimization Using a Simple Evolutionary Algorithm," *In Proceedings of the Fiftheenth International Conference on Tools with Artificial Intelligence (ICTAI'2003)*, pages 149-156, Los Alamitos, CA, November 2003. Sacramento, California, IEEE Computer Society, 2003.

[7] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design," *In Proceedings of the ASME Design Technology Conference*, pages 95-105, Kissimine, Florida, 1988.

[8] J. N. Siddall, "Analytical Decision-making in Engineering Design," *Prentice-Hall, Englewood Cliffs*, 1972.

[9] J. S. Arora, "Introduction to Optimum Design," *McGra-Hill*, New York, 1989.

[10] K. M. Ragsdell and D. T. Phillips, "Optimal design of a class of welded structures using geometric programming," *ASME Journal of Engineering for Industry Series B*, 98(3), 1021–1025, 1976.

[11] K.Deb, "GeneAS: A robust optimal design technique for mechanical component design, Evolutionary Algorithms in Engineering Applications," *Springer-Verlag*, pp. 497–514, 1997.

[12] K.Deb, "Optimal design of a welded beam via genetic algorithms," *AIAA Journal*, 29(11), 2013–2015, 1991.

[13] M. Gen and R. Cheng, "Genetic Algorithms & Engineering Design," *John Wiley & Sons*, Inc, NewYork, 1997.

[14] S. Akhtar, K. Tai and T. Ray, "A Socio-Behavioural Simulation Model for Engineering Design Optimization," *Engineering Optimization*, 34(4):341–354, 2002.

[15] T. Ray and K. M. Liew, "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.

[16] T. V. Hạo, N. H. Thong, "Search via Probability Algorithm for Engineering Optimization Problems," *In Proceedings of XIIth Applied Stochastic Models and Data Analysis (ASMDA2007) International Conference*, 2007.

[17] W. Hock and K. Schittkowski, "Test Examples for Nonlinear Programming Codes," *Lecture Notes in Economics and Mathematical Systems*, Vol. 187. Springer-Verlag, Berlin Heidelberg New York [4.3], 1981.

Appendix

The program illustrates SP algorithm for Minimization of the Weight of a Speed Reducer problem (using Borland C++ 3.1 for DOS).

```
#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 objective(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  a,b[6],r[7],s[6],fix,d1,d2,r1,r2,i,j;  r[0]=random(99)+1;
 for(i=1;i<=6;i++) r[i]=r[i-1]+random(100-r[i-1]);
 for(i=0;i<=5;i++)    {s[i]=0; for(j=i;j<=6;j++) s[i]=s[i]+r[j];}
 d1=50;  d2=75;  r1=random(30)+50;
 do
  {for (i=0;i<n;i++)
   {if (random(100)<r1)
     {z=x[i];
      b[0]=(int)z;z=z-b[0];z=10*z;b[1]=(int) z;z=z-b[1];z=10*z;b[2]=(int) z;z=z-b[2];
      z=10*z;b[3]=(int)z;z=z-b[3];z=10*z;b[4]=(int) z;z=z-b[4];z=10*z;b[5]=(int)z;
      y[i]=0;
      for (j=0;j<6;j++)
        {if (j<3) a=3; else a=5;
         if (random(s[j])<r[j])
          { r2=random(100);
            if (r2<d1)  y[i]=y[i]+pow(10,-j)*random(10);
            else if (r2<d2)
                    y[i]=y[i]+pow(10,-j)*(b[j]-random(a));
                 else y[i]=y[i]+pow(10,-j)*(b[j]+random(a)); }
          else  y[i]=y[i]+pow(10,-j)*b[j];}
      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); clreol();}
void search()
{double  x[n],y[n],gx[m],gy[m],fx,fy,loop; int i;
 randomize(); random_select(x,gx); objective(x,fx); loop=0;
 do
 {change(x,y,gy);  objective(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();}
```