A NEW SEARCH VIA PROBABILITY ALGORITHM FOR SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

NGUYEN HUU THONG*

ABSTRACT

This paper proposes a new stochastic algorithm, Search Via Probability (SVP) algorithm, for single-objective optimization problems. The SVP algorithm uses probabilities to control a process of searching for optimal solutions. We calculate probabilities of an appearance of a better solution than the current one on each iteration, and on the performance of SVP algorithm we create good conditions for its appearance.

Keywords: numerical optimization, stochastic, random, probability, algorithm.

TÓM TẮT

Một giải thuật tìm kiếm theo xác suất mới cho bài toán tối ưu một mục tiêu

Bài viết này đề xuất một giải thuật ngẫu nhiên mới, giải thuật Tìm kiếm theo Xác suất (TKTXS), cho bài toán tối ưu một mục tiêu. Giải thuật TKTXS sử dụng xác suất để điều khiển quá trình tìm kiếm lời giải tối ưu. Chúng tôi tính toán xác suất của sự xuất hiện một lời giải tốt hơn lời giải hiện hành trên mỗi lần lặp, và trong việc thực thi giải thuật TKTXS chúng tôi tạo điều kiện tốt cho sự xuất hiện của lời giải này.

Từ khóa: tối ưu số, ngẫu nhiên, xác suất, giải thuật.

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. However the information of violated constraints is not sufficient for determining a position of an optimal solution.

We can suppose that every decided variable of an optimization problem has m digits that are listed from left to right; we have our remarks as follows:

- To evaluate an objective function, the role of left digits is more important than the role of right digits of a decided variable. We calculate probabilities of changing the values of digits of variables for an appearance of a better solution than the current one on each iteration, and on the performance of SVP algorithm, we create good conditions for its appearance.
- Based on relations of decided variables in the formulas of constrains and an objective function we select k variables ($1 \le k \le n$) to change their values instead of selecting all n variables on each iteration.

* MSc., HCMC University of Education

- Because we can not calculate exactly a number of iterations of a stochastic algorithm for searching an optimal solution the first time on each performance of the algorithm, we use an unfixed number of iterations, which has more chance to find an optimal solution the first time with a necessary number of iterations.

Based on these remarks we introduce a new stochastic algorithm, Search Via Probability (SVP) algorithm, the SVP 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:

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

We can suppose that every decided variable of a solution of an optimization problem has m digits that are listed from left to right. The role of left digits is more important than the role of right digits of a decided variable for evaluating an objective function. Hence we should find the values of digits from left digits to right digits one by one. To do it we want to use probabilities, that is to calculate changing probabilities of digits which can find better values than the current one on each iteration. The proposed algorithm below is a repeated algorithm. On each repeat of algorithm, we select k variables ($1 \le k \le n$) and change their values with the guide of probabilities to find a better solution than the current one. Hence the next work is that we should calculate probabilities of changing values of variables, probabilities of selecting values of changed digits of a variable, and the complex of selecting k variables ($1 \le k \le n$) to change their values.

3. Probabilities of changes and selecting values of a digit

We 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 $0 \le x_{ij} \le 9$, $1 \le j \le m$). To evaluate an objective function, the role of left digits is more important than the role of right digits of a decided variable; hence we calculate changing probabilities of digits which can find better values than the current ones on each iteration.

3.1. Probabilities of changes

Consider the j-th digit x_{ij} of a 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 the probability of A_j, r_j be a number of occurrences of event:

$$\left(\overline{A_1}\,\overline{A_2}\,\mathrm{K}\,\overline{A_{j-1}}A_j\right)^{r_j} \quad (1 \le j \le m)$$

We find values of digits of a variable from left digits to right digits one by one. If the value of left digit x_{ik} (k=1, 2, …, j-1) is not worse than the previous one, we have to fix the values of left digits and change the value of j-th digit to find a new value, and we hope that it may be a better value than the current one of an optimal solution.

The event for finding a new value of 1-th digit: $(A_1)^{r_1}$

The event for finding a new value of 2-th digit: $\left(\overline{A_1}A_2\right)^{r_2}$

Generally, the event for finding a new value of j-th digit:

$\left(\overline{A_1}\,\overline{A_2}\,\mathrm{K}\,\overline{A_{j-1}}A_j\right)^{r_j}$ $(1 \le j \le m)$

After a certain number of iterations of the algorithm, we want to create good conditions for appearances of these events such that these events occur one after the other. Hence we have the product of these events:

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

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

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

and this probability is maximum if

$$q_j = \frac{r_j}{r_j + r_{j+1} + \mathrm{K} + r_m} \quad (1 \le j \le m) \tag{I}$$

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

$r_1 \le r_2 \le \mathrm{K} \le r_m$

Therefore:

$q_1 \le q_2 \le \mathrm{K} \le q_m$

and

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

If m=7, the maximum values of $q=(q_1, q_2, q_3, q_4, q_5, q_6, q_7)$ are:

$$q = \left(\frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\right) = (0.14, 0.17, 0.20, 0.25, 0.33, 0.50, 1) \tag{II}$$

q_1=0.14; q_2=0.17; q_3=0.20; q_4=0.25; q_5=0.33; q_6=0.50; q_6=1.

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 an objective function.

We do not know what digit of variables the algorithm is finding at the current iteration, hence we consider all cases of digits with m=1,2,3,4,5,6,7.

- m=1: $q_1 = 1$.
- m=2: $q_1 = \frac{1}{2} = 0.5,\ q_2 = 1$.
- m=3: $q_1 = \frac{1}{3} = 0.3,\ q_2 = \frac{1}{2} = 0.5,\ q_3 = 1$.
- m=4: $q_1 = \frac{1}{4} = 0.25,\ q_2 = 0.3,\ q_3 = 0.5,\ q_4 = 1$.
- m=5: $q_1 = \frac{1}{5} = 0.2,\ q_2 = 0.25,\ q_3 = 0.3,\ q_4 = 0.5,\ q_5 = 1$.
- m=6: $q_1 = \frac{1}{6} = 0.17,\ q_2 = 0.20,\ q_3 = 0.25,\ q_4 = 0.33,\ q_5 = 0.50,\ q_6 = 1$.
- m=7: $q_1 = \frac{1}{7} = 0.14,\ q_2 = 0.17,\ q_3 = 0.20,\ q_4 = 0.25,\ q_5 = 0.33,\ q_6 = 0.50,\ q_7 = 1$.

We select the values of q_1 where m=1,…,7:

$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}$.

We have the sum of all denominators: 1+2+3+4+5+6+7=28.

We have changing probabilities of each case of m:

Table 1. *The statistics of probabilities changing the value of the digits*

m	Probabilities	Changing probabilities
1	1/28	(1,1,1,1,1,1,1)
2	2/28	(0.5,1,1,1,1,1,1)
3	3/28	(0.33,0.5,1,1,1,1,1)
4	4/28	(0.25,0.33,0.5,1,1,1,1)
5	5/28	(0.20,0.25,0.33,0.5,1,1,1)
6	6/28	(0.17,0.20,0.25,0.33,0.5,1,1)
7	7/28	(0.14,0.17,0.20,0.25,0.33,0.5,1)

The procedure of generating probabilities of changes as follows:

```
R=random(28);
If (R<1) then q=(1,1,1,1,1,1,1)
  Else If (R<3) then q=(0.5,1,1,1,1,1,1)
       Else If (R<6) then q=(0.33,0.5,1,1,1,1,1)
```

Else If (R<10) then q=(0.25,0.33,0.5,1,1,1,1)

Else If (R<15) then q=(0.20,0.25,0.33,0.5,1,1,1)

Else If (R<21) then q=(0.17,0.20,0.25,0.33,0.5,1,1)

Else q=(0.14,0.17,0.20,0.25,0.33,0.5,1)

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 probability of j-th digit incremented by one or a certain value, let R_3 be the probability 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 $R_1=0$, $R_2=R_3=0.5$.

We have 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 ($1 \le k \le n$) 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 randomly ($1 \le k \le n$) 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 B_i be an event that variable x_i can be changed to find a better value than the current one ($1 \le i \le n$). Let B be an event that k variables are selected and then their values are changed to find a better solution than the current one. We have:

$$\Pr(B) = \left(\frac{k}{n}\Pr(B_{i_1})\right)\left(\frac{k}{n}\Pr(B_{i_2})\right)\mathrm{K}\left(\frac{k}{n}\Pr(B_{i_k})\right) = \left(\frac{k}{n}\right)^k \Pr(B_{i_1})\Pr(B_{i_2})\mathrm{K}\ \Pr(B_{i_k})$$

where $1 \le i_1 < i_2 < \mathrm{K} < i_k \le n$

We consider the context of one iteration; probabilities $\Pr(B_i)\ (1 \le i \le n)$ are still fixed if the values of variables $x_i\ (1 \le i \le n)$ are not changed. We consider the worst case, let

$$s = \min_{1 \le i \le n} \Pr(B_i)$$

and $$p = \left(\frac{k}{n}\right)^k \times s^k$$

We have:

$$\Pr(B) = \left(\frac{k}{n}\right)^k \Pr(B_{i_1}) \Pr(B_{i_2}) \text{K } \Pr(B_{i_k}) \ge p$$

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

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

Probability for event B to occur at least once on one of iteration:

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

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

$$\Pr(X > 0) \ge \alpha \Rightarrow 1 - (1-p)^d \ge \alpha \Rightarrow (1-p)^d \le 1 - \alpha \Rightarrow d \ln(1-p) \le \ln(1-\alpha)$$

$$\Rightarrow d \ge \frac{\ln(1-\alpha)}{\ln(1-p)} \Rightarrow d \ge \frac{\ln(1-\alpha)}{\ln\left(1 - \left(\frac{k}{n}\right)^k s^k\right)}$$

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

$$\frac{\ln(1-\alpha)}{\ln\left(1 - \left(\frac{k}{n}\right)^k s^k\right)} \approx \frac{\ln(1-\alpha)}{-\left(\frac{k}{n}\right)^k s^k} = -\frac{\ln(1-\alpha)}{k^k s^k} n^k = C_1 n^k \quad \left(C_1 = -\frac{\ln(1-\alpha)}{k^k s^k}\right)$$

On each iteration, the algorithm performs k works and each work loses time O(1). We have the time of transforming a state:

$$C_1 n^k kO(1) = \left[C_1 kO(1)\right] n^k = C_2 n^k \quad \left(C_2 = C_1 kO(1)\right)$$

If k is a fixed 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, we have the minimum number of iteration:

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

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 select only one variable, it is sufficient for finding a better solution than the current one. The problems of this type have the following similar form:

$$Minimize\ f(x)=\sum_{i=1}^{n} f_i(x_i)$$
$$a_i \le x_i \le b_i,\ a_i, b_i \in R,\ i=1,\text{K},n.$$

In the formula of objective function, all variables are independent. Every variable x_i is calculated independently of other variables.

5. The random search via probability algorithm

The main idea of SVP 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},…, 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. SVP 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 SVP 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 else a=4 ; (*)

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 SVP algorithm has the following characteristics:

- The central algorithmic ideas: The SVP 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 (according to probabilities).
- Variable Loop will be set to 0 if SVP algorithm finds a better solution than the current one; this means that SVP algorithm can find an optimal solution the first time after a necessary number of iterations.
- (*): The right digits vary more than the left digits.

6. Examples

It would be interesting to see how the algorithm SVP performs on more widely used benchmark problems in continuous optimization with or without constraints. Eight test multimodal functions without constraints including Sphere function, Ellipsoidal function, Griewank's function, Ackley's function, two functions of Schwefel, Rastrigin's function, and Rosenbrock's function [1][2][3][4].

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. Sphere function

$$f(x) = \sum_{i=1}^{n} x_i^2 \quad \text{(IV)}$$

Search space: $-100 \le x_i \le 100$, i=1,…,n.

Global optimum solution: $x^*=(x_i=0)$ (i=1,…,n). Global optimum value: $f(x^*)=0$.

There is only an optimum point of this function and it is a global optimum point, i.e. the function is uni-modal. Al variables are not interdependent.

The experimental value of objective function for SVP algorithm: f(x)=0.

Table 2. *Statistics of SVP algorithm for finding the minimum of sphere function*

	k=1		k=3		k=1-5	
	Iterations	Time	Iterations	Time	Iterations	Time
Max	391507	63	889480	155	536015	82
Min	104579	24	282869	50	180913	35
Average	274259	48	513236	85	323574	53
Medium	273174	52	492710	80	299056	51
Standard Deviation	93020	13	178582	30	120814	16

6.2. Ellipsoidal function

$$f(x) = \sum_{i=1}^{n} i * x_i^2 \quad \text{(V)}$$

Search space: $-100 \le x_i \le 100$, i=1,…,n.

Global optimum solution: $x^*=(x_i=0)$ (i=1,…,n). Global optimum value: $f(x^*)=0$.

There is only an optimum point of this function and it is a global optimum point, i.e. the function is uni-modal. Al variables are not interdependent.

The experimental value of objective function for SVP algorithm: f(x)=0.

Table 3. *Statistics of SVP algorithm for finding the minimum of ellipsoidal function*

	k=1		k=5		k=1-3	
	Iterations	Time	Iterations	Time	Iterations	Time
Max	40058	23	201727	56	41661	20
Min	26122	16	105729	32	22133	14
Average	30609	19	139822	41	30545	17
Medium	30010	20	128206	39	29158	17
Standard Deviation	3983.80	1.87	35055.78	8.63	6561.44	2.23

6.3. Griewank's function

$$\textit{Minimize} \qquad f(x) = \frac{1}{4000} * \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \left(\frac{x_i}{\sqrt{i}} \right) + 1 \tag{VI}$$

Search space: $-600 \le x_i \le 600$, i=1,…,n.

Global optimum solution: x*=(x_i=0) (i=1,…,n). Global optimum value: f(x*)=0.

This function is strongly multi-modal, the number of local optimal points increases with the dimensionality.

Select Loop=50000, K=n/10+random(10)

Table 4. *Statistics of SVP algorithm for finding the minimum of Griewank's function*

	n=30		n=50		n=100	
	fx)	Time	f(x)	Time	f(x)	Time
Min	0.0000000000	16	0.0000000000	26	0.0000000008	68
Max	0.0811674346	18	0.0393113727	31	0.0344011556	109
Average	0.0238396198	17	0.0127923868	28	0.0085991893	75
Medium	0.0159957028	17	0.0098585146	28	0.0000000030	71
Standard Deviation	0.0269205669	1	0.0137978969	1	0.0140545242	12

6.4. Ackley's function

$$f(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - epx\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right) \tag{VII}$$

Search space: $-100 \le x_i \le 100$, i=1,…,n.

Global optimum solution: x*=(x_i=0) (i=1,…,n). Global optimum value: f(x*)=0.

This function has many local optima, i.e. it is multi-modal. The difficult part about finding optimal solutions to this function is that an optimization algorithm easily can be trapped in a local optimum on its way towards the global optimum.

The experimental value of objective function for SVP algorithm: f(x)=0, and n=100.

Table 5. *Statistics of SVP algorithm for finding the minimum of Ackley's function*

	k=1% of n		k=5% of n		k=1%-5% of n	
	Iterations	Time	Iterations	Time	Iterations	Time
Min	32897	15	192222	38	35925	14
Max	44943	17	277869	53	45988	15

Average	38550	16	218740	42	42218	15
Medium	38180	17	202435	39	43479	15
Standard Deviation	5157	1	40205	7	4523	1

6.5. Schwefel's function 1

$$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 \qquad \text{(VIII)}$$

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

The experimental value of objective function for SVP algorithm: f(x)=0.1.

Table 6. *Statistics of SVP algorithm for finding the minimum of Schwefel's function 1*

	K=1		k=20%-60% of n		K=10%-80% of n	
	Iterations	Time	Iterations	Time	Iterations	Time
Max	323912	434	70984	37	**60880**	**32**
Min	289192	293	52863	24	**46284**	**26**
Average	311449	340	60318	29	**53330**	**30**
Medium	316345	317	58712	28	**53078**	**30**
Standard Deviation	16221.28	63.94	7728.72	5.56	**6988.01**	**2.65**

6.6. Schwefel's function 2

$$\begin{aligned} &\textit{Minimun } f(x) = 418.9829.n + \sum_{i=1}^{n}(-x_i.\sin(\sqrt{|x_i|})) \\ &n = 30, -500 \le x_i \le 500 \ (i = 1,\text{K},30) \end{aligned} \qquad \text{(IX)}$$

Search space: $-500 \le x_i \le 100$ (i=1,…,n).

Global optimum solution: $x^*=(x_i=0)$ (i=1,…,n). Global optimum value: f(x*)=0.

Loop=50000, k=10-30%.

Table 7. *Statistics of SVP alg. for finding the minimum of Schwefel's function 2*

	N=30		N=50		N=100	
	f(x)	Time	F(x)	Time	f(x)	Time
Min	0.000382	14	0.000657	22	0.151995	39
Max	0.000382	16	0.000720	24	0.585872	44
Average	0.000382	15	0.000681	23	0.357464	42

Medium	0.000382	15	0.000682	24	0.391186	43
Standard Deviation	0.000000	1	0.000019	1	0.154470	2

Loop=100000, n=100

Table 8. *Statistics of SVP alg. for finding the minimum of Schwefel's function 2*

	k=10-50%		k=40-80%		k=50-100%	
	f(x)	Time	F(x)	Time	f(x)	Time
Min	0.079213	55	886.560075	50	3843.442593	56
Max	0.263182	58	1428.157463	52	4773.125910	57
Average	0.143390	57	1151.183931	51	4196.844639	57
Medium	0.137396	57	1153.912549	51	4181.108227	57
Standard Deviation	0.047134	1	211.887194	1	266.041875	1

6.7. Rastrigin's function

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \qquad (X)$$

Search space: $-100 \le x_i \le 100$ (i=1,…,n).

Global optimum solution: $x^* = (x_i = 0)$ (i=1,…,n). Global optimum value: $f(x^*) = 0$.

This function has many local optima, i.e. it is multi-modal. The difficult part about finding optimal solutions to this function is that an optimization algorithm easily can be trapped in a local optimum on its way towards the global optimum.

The experimental value of objective function for SVP algorithm: f(x)=0.

Table 9. *Statistics of SVP algorithm for finding the minimum of Rastrigin's function*

	k=1		k=5		k=1-5	
	Iterations	Time	Iterations	Time	Iterations	Time
Max	63934	21	317432	57	54585	17
Min	28393	13	156305	30	35509	14
Average	37156	16	212384	40	44314	15
Medium	33060	15	205548	40	44166	15
Standard Deviation	10994.74	2.37	55697.37	9.17	6129	1

6.8. Rosenbrock's function

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

Search space: $-100\le x_i\le 100$ (i=1,…,n).

Global optimum solution: $x^*=(x_i=1)$ (i=1,…,n). Global optimum value: f(x*)=0.

The global optimum point is the only optimum point. However, the variables are strongly interdependent, which makes this function very difficult to optimize using particle swarms and evolutionary algorithms.

Loop=100000, n=100

Table 10. *Statistics of SVP alg. for finding the minimum of Rosenbrock's function*

	K=5-20%		K=20-80%		K=50-100%	
	F(x)	Time	F(x)	Time	f(x)	Time
Min	0.095616	88	57.499233	93	335.043873	100
Max	70.879085	76	205.010381	76	447.997597	99
Average	9.360828	79	124.836888	79	380.865796	100
Medium	2.979793	78	121.599714	77	377.163320	100
Standard Deviation	21.654109	4	53.888290	5	34.741264	0

Remarks on 8 test multimodal functions: 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 should select k=10%-20% of n to increase the convergent speed of algorithm. The test problem 6 has n variables which are interdependent, the increase or decrease of one variable x_i ($1\le x_i\le n$) influences previous term and next term, therefore we have to select many variables (k>1) to change their values. See the statistical table 8 of problem 6 for n=10, n=20 and n=30, because the problem 6 has interdependent variables, therefore the number of iterations increases very quickly in the ratio of n. **It means that the complexity of SVP algorithm for these problems is not based on formulas of problems, such as linear or nonlinear, but it is based on the relations of variables in the objective function and constraints**.

7. Conclusions

In this paper, we proposed a new approach for single-objective optimization problems, Search via Probability algorithm, this algorithm was extended from [4]. The SVP 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 SVP algorithm, we created

good conditions for its appearance. The idea of SVP algorithm was based on essential remarks as follows:

- The role of left digits was more important than the role of right digit for evaluating an 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 SVP 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 SVP 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 SVP algorithm on some test single-objective optimization problems, and we found very stable results. We are applying SVP algorithm for solving multiobjective optimization and discrete optimization problems.

REFERENCES

1. 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.
2. *EMOO Home Page*, http://www.lania.mx/~ccoello/
3. *GEATbx: Example Functions (single and multi-objective functions)*, http://www.geatbx.com/docu/fcnindex-01.html
4. Nguyễn Hữu Thông and Trần Văn Hạo (2007), "Search via Probability Algorithm for Engineering Optimization Problems", In Proceedings of XIIth Applied Stochastic Models and Data Analysis (ASMDA2007) International Conference. In book: Recent Advances in Stochastic Modeling and Data Analysis, editor: Christos H. Skiadas, publisher: World Scientific Publishing Co Pte Ltd, 454 – 463.
5. Yang X.-S.(2010), A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (Eds. J. R. Gonzalez et al.), SCI 284, 65-74.

(*Received: 28/5/2012; Revised: 03/12/2012; Accepted: 18/02/2013)*