Improvement of dynamic encoding algorithm for searches (DEAS) using hopping unidirectional search (HUDS)

Seong-Chul Choi*, Nam Gun Kim*, Jong Wook Kim** and Sang Woo Kim*

*Division of Electrical and Computer Engineering, POSTECH, Pohang, Korea
(Tel: +82-54-279-5018; Fax: +82-54-279-2903; Email:{alp, kimng, swkim}@postech.ac.kr)
**Technical Research Laboratories, POSCO, Pohang, Korea
(Tel : +82-54-220-6166; E-mail: kjune@posco.co.kr)

Abstract: Dynamic Encoding Algorithm for Searches (DEAS) which is known as a fast and reliable non-gradient optimization method, was proposed [1]. DEAS reaches local or global optimum with binary strings (or binary matrices for multi-dimensional problem) by iterating the two operations; bisectional search (BSS) and unidirectional search (UDS). BSS increases binary strings by one digit (i.e., 0 or 1), while UDS performs increment or decrement of binary strings in the BSS' result direction with no change of string length. Because the interval of UDS exponentially decreases with increment of bit string length (BSL), DEAS is difficult to escape from local optimum when DEAS falls into local optimum. Therefore, this paper proposes hopping UDS (HUDS) which performs UDS by hopping as many as BSL in the final point of UDS process. HUDS helps to escape from local optimum and enhances a probability searching global optimization. The excellent performance of HUDS will be validated through the well-known benchmark functions.

Keywords: DEAS, Global Optimization, BSS, UDS, α-HUDS, Benchmark

1. Introduction

Many optimization problems in engineering, science and economics require solutions that are globally optimal. Many new theoretical, algorithmic, and computational contributions of global optimization have been investigated to solve many problems in many other fields. Every constrained optimization consists of three components: a set of unknowns or variables to be determined, an objective function to be minimized or maximized, and a set of constraints to be satisfied. In other words, optimization is to find values of variables that optimize (minimize or maximize) the objective function while satisfying all the constraints. In general, the objective function may be not convex and have more than one local minimum in the feasible region *D*. So the solution can be not globally optimal but locally optimal. And since the general criterion for global optimization does not exist, global optimization problems are typically quite difficult to solve exactly.

Generally, global optimization can be categorized into two different approaches; indirect and direct methods [3], [4]. The indirect approaches use gradient information of the objective function for iterative approximation as in the steepest descent method, the Newton-Raphson method, and the conjugate gradient method. In these algorithms in the feasible region, the objective function should be sufficiently smooth and be twice differentiable, which is quite rigorous conditions in real world problems.

The direct approaches utilize alternative techniques, e.g. a random walk in the random search method and the theory of natural selection in genetic algorithms (GAs) [5]. Not using the gradient information, these algorithms are more straightforward to apply but less efficient than the indirect approaches in terms of their running time and solution qualities. However, the available high performance of microprocessors means that direct approaches are being applied to a variety of engineering field.

To make a more effective global optimization strategy, two basic approaches need to be combined even if the basic philosophies behind the two approaches are quite different. In the sequel, a nonlinear optimization algorithm named dynamic encoding algorithm for searches (DEAS) is developed in the framework of discrete structures. DEAS is originated from the need of the compromise between indirect and direct methods. In other words, DEAS is an iterative algorithm which gets approximate gradient information, i.e. a search direction, through not analytical method but numerical method and directs to next minimum from a current minimum.

DEAS performs *bisectional search* (BSS) operation to get a search direction and *unidirectional search* (UDS) operation to successively approach to optimum. Especially, UDS is the operation that performs increment and decrement of binary strings in the search direction, a BSS' result direction. But, because the interval of UDS exponentially decreases, falling into local optimum, DEAS doesn't easily escape from local optimum. And DEAS doesn't basically guarantees global optimization. Therefore, this paper suggests hopping UDS (HUDS) as a means to escape from local optimum and to raise a probability to find global optimization. HUDS operates UDS by hopping into points as apart as bit string length (BSL) from the final point of UDS process without change of search direction.

The paper is organized as follows: Section 2 describes the optimization strategies of DEAS. Section 3 explains the proposed method. Section 4 gives the simulation results and the performance comparison with existing DEAS and proposed method. Finally, Section 5 brings to a conclusion.

2. Dynamic Encoding Algorithm for Searches

In this section, the basic search strategies of DEAS are introduced. In addition, several global search strategies are proposed [2].

2.1. Basic Behavior of DEAS

The basic behavior of DEAS is described by two terminologies: bisectional search (BSS) and unidirectional search (UDS).

2.1.1 Bisectional Search

If a binary digit, 0 or 1, is appended to an existing binary string as a least significant bit (LSB), the decoded real number of a new

binary string decreases for 0, and increases for 1 compared with that of the original binary string.

From the viewpoint of local search techniques, the BSS can be comprehended as the selection of the best neighborhood matrices among the 2^n neighborhood ones. In Fig. 1, the neighborhood strings generated by the BSS are depicted with points. Among those neighborhood matrices, the best one whose cost is lowest is saved as an optimal matrix.

The pseudocode for an *n*-dimensional problem of BSS is stated as follows, where BSS is undertaken for an $n \times m$ binary matrix. Note that n equals the number of search parameters, and bit number of string, m, can vary from 1 to any finite number.

Bisectional Search ($\mathbf{B}_{n\times m}$)
Initialization:
Set a direction vector as an all-zero binary string;
$\mathbf{d} = \mathbf{0}_{n\times 1}$.
Set $J_{min} \leftarrow$ temporary variable, $M \gg 1$ and $i \leftarrow 1$.
while $i \leq 2^n$ **do**
Add $\mathbf{d}$ as a least significant column;
$\mathbf{B}_{n\times(m+1)} = [\mathbf{B}_{n\times m}\ \ \mathbf{d}]$.
Decode a temporary matrix into a real vector as
$\mathbf{B}_{n\times(m+1)} \xrightarrow{f_d} \mathbf{X}_{n\times 1}$.
Evaluate the cost;
$J = f(\mathbf{X})$.
if $J < J_{min}$ **then**
Save the current best optima;
$\mathbf{B}^*_{n\times(m+1)} \leftarrow \mathbf{B}_{n\times(m+1)}$,
$\mathbf{d}_{opt} \leftarrow \mathbf{d}$,
$\mathbf{X}_{min} \leftarrow \mathbf{X}$,
$J_{min} \leftarrow J$.
end if
$\mathbf{d} \leftarrow \mathbf{d} + 1$
$i \leftarrow i + 1$
end while

Let an *i*-th dimensional *m*-bit-long binary string,

$$\mathbf{B}_{i,m} = b_m \cdots b_j \cdots b_1,\ \ b_j \in \{0,1\},\ j = 1, \cdots, m,$$

be termed a *parent string*. If 0 is added to the LSB of $\mathbf{B}_{i,m}$, the new string is termed a *'left-hand child string'*, $\mathbf{B}^l_{i,m+1}$. On the other hand, if 1 is added, it is termed a *'right-hand child string'*, $\mathbf{B}^r_{i,m+1}$.

In the process of DEAS, the mechanism of decoding from the binary vector space $\{0,1\}^m$ to restricted continuous space $\mathbb{R}^n$ on

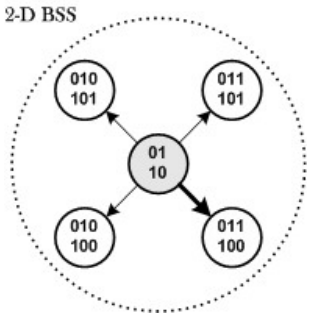

Fig. 1. 2-dimensional diagrams of neighborhood points generated by BSS

finite intervals $[u_i, v_i]$ for each variable $\mathbf{x^i} \in \mathbb{R}$ is required. According to the standard binary decoding function $f^i_d : \{0,1\}^m \rightarrow [u_i, v_i]$, where [6], the real value of the parent string is

$$x_i = f_d(\mathbf{B}_{i,m}) = u_i + \frac{r_i}{2^m - 1}\sum_{j=1}^{m} b_j 2^{j-1},\ \text{ where }\ r_i = v_i - u_i$$

and the child strings are respectively

$$\begin{aligned} x^l_{i+1} &= f_d(\mathbf{B}^l_{i,m+1}) = u_i + \frac{r_i}{2^{m+1}-1}\sum_{j=1}^{m} b_j 2^j \\ &= \frac{2^{m+1}-2}{2^{m+1}-1}\ x_i, \\ x^r_{i+1} &= f_d(\mathbf{B}^r_{i,m+1}) = u_i + \frac{r_i}{2^{m+1}-1}\sum_{j=1}^{m} b_j 2^j + \frac{r_i}{2^{m+1}-1} \\ &= \frac{1}{2^{m+1}-1} + \frac{2^{m+1}-2}{2^{m+1}-1}\ x_i. \end{aligned}$$

Difference between the parent string and the child strings is the search step size that is important factor to determine the convergence velocity of DEAS and is given by

$$| x^l_{i+1} - x_i | = \frac{1}{2^{m+1}-1}\ x_i, \qquad (1)$$

$$| x^r_{i+1} - x_i | = \frac{1}{2^{m+1}-1}\ (1 - x_i). \qquad (2)$$

From (1) and (2), the magnitude of the differences decreases exponentially as the length of a parent string increases.

2.1.2 Unidirectional Search

BSS alone, however, has a serious drawback of regional limitation. The reachable search area by pure BSS is $1/2$ of intervals $[u_i, v_i]$. However, the simple operations of increment addition (INC) and decrement subtraction (DEC) for a binary string can readily remove the barrier between any binary trees. In a unidirectional search (UDS) of DEAS, these operations are iterated while a better optimum is sought in a guided search direction obtained by the previous BSS.

In conjunction with the pseudo-code of BSS, the UDS routine is written in detail as follows.

Unidirectional Search ($\mathbf{B}^*_{n\times k}$, $\mathbf{d}_{opt}$, J_{min}):
Load $\mathbf{B}^*_{n\times k}$, $\mathbf{d}_{opt}$ and J_{min} of BSS
Initialization: $\mathbf{U}^*_{n\times k} \leftarrow \mathbf{B}^*_{n\times k}$.
while A better solution is attained **do**
Set an extension vector; $\mathbf{e} = [\underbrace{0\ 0 \cdots\ 0}_{l-1}\ 1]^T$.
for $i = 1 : 2^l - 1$
Redundancy check:
if e is computed by $\mathbf{e_{opt}}$ as a reevaluating direction **then** CONTINUE.
Load the current best matrix into a temporary matrix;
$\mathbf{T}_{n\times k} \leftarrow \mathbf{U}^*_{n\times k}$.
for $j = 1 : l$
$p = \mathbf{u}(j)$.
Select the p-th row of $\mathbf{T}$; $\mathbf{r^T_{k\times 1}} = \mathbf{T}(\mathbf{p}, \mathbf{1} : \mathbf{k})$.
if $\mathbf{e}(j) = 1$ **then**
if $\mathbf{d}_{opt}(p) = 0$ **then** $\mathbf{r} = \mathbf{r} - 1$.

else $\mathbf{r} = \mathbf{r} + 1$.
end if
$\mathbf{T}(\mathbf{p}, \mathbf{1} : \mathbf{k}) = \mathbf{r}^{\mathbf{T}}$.
end for
Decode the modified binary matrix into a real vector:
$\mathbf{T}_{n\times k} \overset{f_d}{\rightarrow} \mathbf{X}_{\mathrm{n}\times 1}$.
Evaluate the cost; $J = f(\mathbf{X})$.
if $J \leq J_{min}$ **then**
Save the current best optimum;
$\mathbf{U}^*_{n\times k} \leftarrow \mathbf{T}$,
$\mathbf{e}'_{\mathbf{opt}} \leftarrow \mathbf{e}$,
$\mathbf{X}_{min} \leftarrow \mathbf{X}$,
$J_{min} \leftarrow J$.
end if
$\mathbf{e} \leftarrow \mathbf{e} + 1$
end for
$\mathbf{e_{opt}} \leftarrow \mathbf{e}'_{\mathbf{opt}}$
end while

From the viewpoint of a local optimization strategy, successive reformation of binary strings in BSS and UDS corresponds to the determination of optimal search directions and step lengths, respectively, and a neighborhood of a current solution is mapped to all the vertexes of gradually shrinking hypercubes in multi-dimensional problems. Therefore, DEAS is deemed to retain the unique property of a dynamic-sized neighborhood selection, while most combinatorial optimization algorithms use a fixed-sized neighborhood selection. A combination of BSS and UDS for a given string length, referred to as a *session* hereafter, is repeated with gradually increasing binary strings.

2.2. Global search strategies

In this section, global search strategies using information of DEAS are proposed.

2.2.1 History Check

In searching with DEAS, the whole search space of interest is dynamically divided by a fixed number of grids. It implies that the search path of each binary string is automatically determined according to the cost function landscape unless it is time-variant. That is, if DEAS is applied to any two identical strings, the subsequent search process will yield exactly the same search results. Moreover, DEAS enables UDS to search horizontally. Therefore, search paths of different initial matrices can encounter at one of the subordinate vertexes. Since this revisit gives rise to unnecessary computation, it must be detected and prohibited at every instance of session starting. As a solution to this problem, an integer- or real-valued representative for a string concatenated from a matrix is saved and compared in the lookup table by the following assignment function:

$$f_a(b_{m-1}b_{m-2}\cdots b_1 b_0) = \sum_{j=0}^{m-1} b_j a^j \tag{3}$$

where $b_i \in \{0, 1\}$, and a is an integer or real base value. As an illustration, a binary matrix is assigned an integer value by a series of concatenations and an assignment with (3) of base 2 as

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \Rightarrow 6$$

Then, every newly assigned value is compared with those of previous optimal matrices whose corresponding row lengths are identical. Using this method, DEAS can effectively check the revisits and reduce useless computations.

Initialize the history table.
For $restart = 1 : maxRestart$
Randomly generate an initial binary matrix; $\mathbf{T}_{n\times initLen}$
HISTORY CHECK:
Check if the initial matrix has already been visited.
For $m = initLen : finLen$
$\left[\mathbf{B}^*_{n\times(m+1)},\ \mathbf{d}_{opt},\ J_{min}\right]$=**BSS (** $\mathbf{T}_{n\times m}$**)**
$\mathbf{B}^*_{n\times(m+1)}$=**UDS (** $\mathbf{B}^*_{\mathbf{n}\times(\mathbf{m+1})}$, $\mathbf{d}_{opt}$, J_{min}**)**
HISTORY CHECK:
Check if $\mathbf{B}^*_{\mathbf{n}\times(\mathbf{m+1})}$ has already been visited.
$\mathbf{T}_{n\times(m+1)} = \mathbf{B}^*_{\mathbf{n}\times(\mathbf{m+1})}$
end for
end for

The above mentioned global search strategy is shown as simplified pseudocodes. In an n-dimensional problem, the cost function is evaluated 2^n and $2^{l(k)} - 1 - r(k)$ times for BSS and UDS, respectively, where $l(k)$ and $r(k)$ represent the number of permitted variables and the number of redundant searches at the k-th transition, respectively.

2.2.2 Preliminary Search

An additional important feature for global optimization is an escaping scheme. As optimal binary matrices are lengthened, their step lengths decrease exponentially, which means that search points continuously converge to one of the local minima. Thus when successive cost values do not drop below a predefined acceptable value, it can be inferred that a current point falls inside the region of attraction (ROA) of an unacceptable local minimum. For this situation, DEAS commands to escape from a current point and restart, which is similar to the multistart method.

However, it is difficult to theoretically determine the optimal indexes of escaping, which is common to all global optimization methods. DEAS resorts to a kind of heuristics, i.e., surveying the landscape of the cost function by a finite number of trials as a preliminary search. This brief survey provides such significant information as *optInitRowLen*, *rowIndRestart*, and *costIndRestart*. The *optInitRowLen* means an optimal initial row length by which a global minimum can be discovered. Since step lengths are larger with a smaller row length, a small *optInitRowLen* means ROAs of global minima spread wide over the whole search space. The *rowIndRestart* and *costIndRestart* represent optimal indices of a row length and a cost value for restarts by which it is determined to restart or to continue. The *rowIndRestart* is selected as the minimal row length from which global minima and local minima can be discriminated with naked eyes, and *costIndRestart* is an approximately intermediate value between the best and the second-best local minima. These values are closely related to the smoothness, the ratios of ROAs, and the slopes of the cost function currently handled. Therefore, the preliminary search of DEAS additionally provides approximate information of the cost function.

3. α-HUDS

In previous section, DEAS proposed history check, preliminary search and the restart method as the strategies of global optimization. In summary, history check reduces cost computations, preliminary search provides the criterions which DEAS needs and the restart helps to escape from local optimum. However, with the increment of bit string length (BSL), the interval of UDS exponentially decreases, which interrupts to escape from local optimum. And if DEAS falls into local optimum, the success number of UDS sharply declines. On the other words, DEAS is difficult to escape from local optimum as session is repeated. Falling in local optimum, DEAS does not try to search the better optimum in the current position and just restarts. As a result, the chance to find global optimum which can be near to current local optimum is lost. Therefore, other methods for extricating local optimum need.

In [2], a method for escaping from local optimum is proposed, which performs UDS by hopping as many as the success number of previous UDS in the final point of a session. For instance, in the 2-dimensional problem if the result of BSS is $[x_1, x_2]$=[0100b, 0010b], the search directions are both increment directions, and the final point of the session is $[x_1, x_2]$=[1001b, 0110b], then the success number of UDS process is [1001b, 0110b]−[0100b, 0010b]=[0101b, 0100b]. Next, the cost of final point is compared with the costs of points [1001b+0101b, 0110b], [1001b, 0110b+0100b] and [1001b+0101b, 0110b+0100b] and the smallest cost among the costs is chosen. Finally, if the selected cost is the final cost of the session, perform the next session, unless operate continuously UDS in the selected new point and repeat the same procedure. However, approaching local optimum, this method is hard to overcome local optimum because the success rate of UDS decreases.

For this reason, this paper proposes a new method that performs UDS by hopping as many as BSL in the final point of the current session. Using above example, since BSL is 4=0100b, the final cost of the session is balanced with the costs of points [1001b+0100b, 0110b], [1001b, 0110b+0100b], [1001b+0100b, 0110b+0100b] and the proposed method reiterates the same process like the proposed method in [2]. Hereafter, we will name the proposed method Hopping UDS (HUDS). Changing a little HUDS, BSL times α, hopping ratio, $\alpha > 0$, instead of BSL can be used. It will be called α-HUDS. For example, if $\alpha = 2$, the hopping length is $2\times4 = 8 = 1000$b. The cost of final point is compared with the costs of points [1001b+1000b, 0110b], [1001b, 0110b+0100b], [1001b+1000b, 0110b+0100b]. But, because 1001b+1000b=10001b is out of the variable's range, the boundary value, i.e. 1111b in substitute for 10001b is used. Contrarily, if the value is negative, 0000b is used. Fig. 2 shows the process of 2-D 1-HUDS. In Fig. 2, the session of DEAS stops in the point *A*, while 1-HUDS escapes from local optimum and reaches the better optimum point *B*.

When BSL is m, the hopping length l_i of variable x_i is

$$l_i(m) = \frac{\alpha m}{2^m - 1} r_i. \tag{4}$$

where $initLen+1 \leq m \leq finLen$, and r_i is the range of variable x_i. And the variation d_i of the hopping length of the variable x_i is

$$\begin{aligned} d_i &= l_i(m+1) - l_i(m) \\ &= \frac{\alpha(m+1)}{2^{m+1}-1} r_i - \frac{\alpha m}{2^m - 1} r_i \\ &= \alpha \left[\frac{(m+1)(2^m-1) - m(2^{m+1}-1)}{(2^m-1)(2^{m+1}-1)} \right] r_i \\ &= -\alpha \left[\frac{(m-1)2^m + 1}{(2^m-1)(2^{m+1}-1)} \right] r_i < 0, \end{aligned}$$

and the hopping length decreases with increment of BSL. It means that α-HUDS is a means to find global optimum not far but near to local minimum. Intuitively, in the final point of a session, the operation of HUDS gives propulsion that can escape from local optimum and raises a probability searching global optimum near to local optimum.

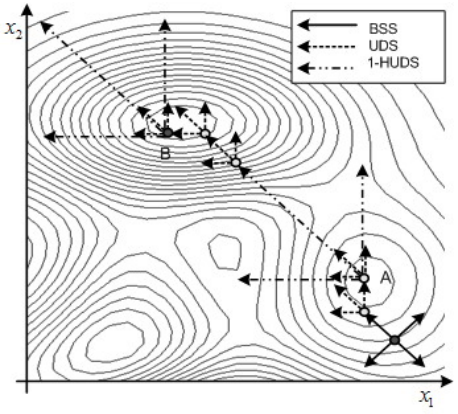

Fig. 2. An example of the process of 2D 1-HUDS (BSL = 4)

4. Simulation and Performance Evaluation

To evaluate the proposed method, fourteen benchmark functions [8], [9] were used in this paper. This number is enough to estimate the performance of α-HUDS. Functions $f_1 - f_5$ are unimodal. Function f_6 is the step function which has a minimum surface and is discontinuous. Functions $f_7 - f_{11}$ are multimodal functions which have many local minima. Functions $f_{12} - f_{14}$ are multimodal functions which have only a few local minima. A more detailed description of each function is given in the Appendix.

The simulation compared α-HUDS with existing DEAS and the proposed method in [2]. Only in the 2-dimensional problem, the evaluation of algorithms is tested with these functions in the same condition. To get enough information for evaluation, $initLen$ is set to 5. So the number of candidates of starting points is $2^{2\times initLen} = 1024$. The criterions for restart are obtained through the preliminary search.

Table. 1 shows the results of simulation and consists of 2 parts. The first column is benchmark functions and the remaining columns are composed of the probability that is reached to global optimum among all starting points and the number of cost computation during simulation. In case of functions $f_1 - f_4$, f_{13}, all algorithms always reach global optimum and the cost computations of α-HUDS increases a few in relation to DEAS. In case of the remaining functions, the performance of α-HUDS is as equal as or superior to those of DEAS and the proposed method in [2]. In case of function f_5,

Table 1. Performance comparison of existing DEAS, α-HUDS and the proposed method in [2]

		DEAS	α=0.5	α=1.0	α=2.0	α=3.0	α=5.0	α=10	[2]
f_1	Prob.(%)	100	100	100	100	100	100	100	100
	Cost CT.	5183	5210	5210	5210	5210	5210	5210	5183
f_2	Prob.(%)	100	100	100	100	100	100	100	100
	Cost CT.	5183	5210	5210	5210	5210	5210	5210	5183
f_3	Prob.(%)	100	100	100	100	100	100	100	100
	Cost CT.	6790	6884	6912	6894	6912	6914	6912	6950
f_4	Prob.(%)	100	100	100	100	100	100	100	100
	Cost CT.	5791	5853	5853	5853	5853	5853	5856	5893
f_5	Prob.(%)	0.00	0.00	**0.30**	0.00	0.00	0.00	0.00	0.00
	Cost CT.	6360	6551	6628	6573	6546	6583	6574	6404
f_6	Prob.(%)	0.88	28.22	**43.07**	**43.07**	14.06	4.69	2.34	0.88
	Cost CT.	26637	11919	10926	14312	17040	25062	29347	26637
f_7	Prob.(%)	3.52	3.52	3.52	**7.91**	3.52	4.79	**14.06**	3.52
	Cost CT.	6445	6629	6746	6618	6287	6480	6296	6445
f_8	Prob.(%)	1.56	1.56	**47.27**	14.06	**66.02**	3.52	9.77	1.56
	Cost CT.	8288	9800	9800	9596	8240	10564	10608	8288
f_9	Prob.(%)	50.39	50.39	50.39	50.39	50.39	50.39	50.39	50.39
	Cost CT.	7556	7565	8013	6965	6949	6953	6505	7556
f_{10}	Prob.(%)	5.86	5.86	5.86	**6.45**	5.86	5.86	5.86	6.25
	Cost CT.	8613	9463	9324	9508	8933	9330	9001	8641
f_{11}	Prob.(%)	4.00	**37.01**	**41.99**	14.65	**39.61**	8.79	4.69	4.20
	Cost CT.	20625	15787	13547	17409	18891	18688	17515	20535
f_{12}	Prob.(%)	33.40	37.70	**44.14**	**87.89**	**43.55**	37.70	**44.53**	51.76
	Cost CT.	5386	5510	5490	5514	5468	5556	5646	5538
f_{13}	Prob.(%)	100	100	100	100	100	100	100	100
	Cost CT.	6620	6779	6788	6801	6797	6775	6740	6849
f_{14}	Prob.(%)	44.82	44.82	**51.46**	**66.50**	**82.81**	**64.75**	**57.52**	53.52
	Cost CT.	9094	9321	8972	8857	9043	8934	8481	9867

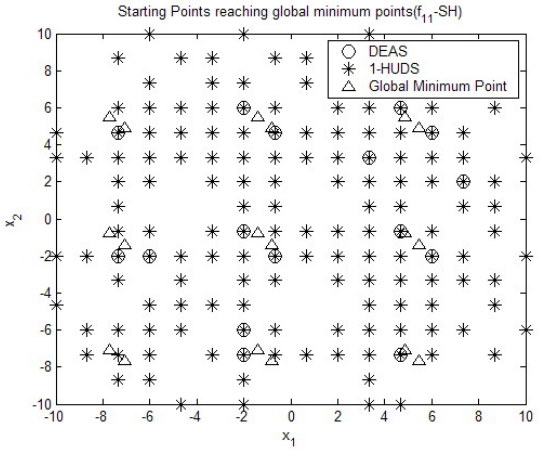

Fig. 3. Starting points reaching global optimum (*initLen*=4, circle : DEAS, asterisk : 1-HUDS, triangle : global minimum point)

especially, other algorithms except 1-HUDS did not find global optimum. In general, the cost computation of α-HUDS increases in comparison with existing DEAS, but the increment rate of the cost computation is small relatively in relation to that of the performance enhancement.

Fig. 3 is to plot the starting points reaching global minimum of a function f_{11}, *Shubert Function* and well shows the feature of α-HUDS. To well see points in a figure, *initLen* is set to 4. The candidates of starting points is $2^{2\times initLen} = 256$. Function f_{11} has 18 global minimums and 720 local minimum. In case of existing DEAS, only 19 starting points near to global minimum reaches global minimum points. On the other hand, in case of α-HUDS, 153 starting points far as well as near to global minimum points reach global minimum points. Results of simulation shows that α-HUDS is more excellent than existing DEAS.

5. Conclusion

DEAS approaches local or global optimum through BSS and UDS process. Both BSS and UDS are simple and powerful operations. Increasing BSL, however, the interval of UDS process exponentially decreases. As a result, falling into local optimum, DEAS is hard to escape from local optimum by using only BSS and UDS process.

For this reason, this paper proposed α-HUDS as a method for finding the better optimum around current local optimum. α-HUDS performs UDS by hopping into points as apart as $\alpha\times$BSL in the final point of a session. It gives momentum to extricate from local optimum and enhances a probability to find global optimum.

As a result of simulation, the performance of α-HUDS is more excellent than that of existing DEAS. Generally, α-HUDS requires the additional computation, but returns the relatively good results. Using DEAS together with α-HUDS, the better results can be obtained. However, it is still difficult to determine adequate α by every objective function. In the future, we will try to make the criterion to decide hopping ratio, α or not constant but adaptively changing α.

References

[1] Kim, J.-W. and Kim, S.W., "Numerical method for global optimisation: dynamic encoding algorithm for searches", *IEE Proc.- Control Theory & Appl.*, vol.151, No.5, Sept. 2004, , pp.661-668.

[2] Kim N.-G., Kim J.-W., and S.-W. Kim, "A Study for global optimization using dynamic encoding algorithm for searches", *International Conference on Control, Automation and Systems*, pp. 857-862, ICASS, Bangkok, Thailand, 2004.

[3] L.C.W. Dixon and G.P. Szegö, *The global optimization: An Introduction*, in: Dixon and Szegö, eds., *Towards global optimization 2*, North-Holland, Amsterdam, 1978, pp. 1-15.

[4] A. Törn and A. Žilinskas, *Global Optimization*, Springer- Verlag, Berlin, 1989.

[5] Goldberg, D.E., *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley, 1989

[6] T. Bäck, *Evolutionary Algorithm in Theory and Practice*, Oxford University Press, New York, 1996.

[7] B. B. Brey, *The Intel 32-Bit Microprocessors*, Prentice-Hall Inc. Englewood Cliffs, 1995.

[8] Xin Yao, Yong Liu and Guangming Lin, "Evolutionary Programming Made Faster", *IEEE Transactions on Evolutionary Computation*, vol. 3, No. 2, July 1999, , pp. 82-102.

[9] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

Appendix

BENCHMARK FUNCTIONS

1. Sphere Model

$$f_1(x) = \sum_{i=1}^{2} x_i^2, \qquad -10 \le x_i \le 10$$

$$\min(f_1) = f_1(0,0) = 0.$$

2. Schwefel's Problem 2.22

$$f_2(x) = \sum_{i=1}^{2} |x_i| + \prod_{i=1}^{2} |x_i|, \qquad -10 \le x_i \le 10$$

$$\min(f_2) = f_2(0,0) = 0.$$

3. Schwefel's Problem 1.2

$$f_3(x) = \sum_{i=1}^{2} \left(\sum_{j=1}^{i} x_j \right)^2, \qquad -10 \le x_i \le 10$$

$$\min(f_3) = f_3(0,0) = 0.$$

4. Schwefel's Problem 2.21

$$f_4(x) = \max_i \{|x_i|, i = 1, 2\}, \qquad -10 \le x_i \le 10$$

$$\min(f_4) = f_4(0,0) = 0.$$

5. Proposed Function 1

$$f_5(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2, \qquad -30 \le x_i \le 30$$

$$\min(f_5) = f_5(1,1) = 0.$$

6. Proposed Function 2

$$f_6(x) = \sum_{i=1}^{2} (\lfloor x_i + 0.5 \rfloor)^2, \qquad -10 \le x_i \le 10$$

$$\min(f_6) = 0 \text{ for } -0.5 \le x_i < 0.5$$

7. Generalized Schwefel's Problem 2.26

$$f_7 = -\sum_{i=1}^{2} \Big(x_i \sin(\sqrt{|x_i|}) \Big), \qquad -500 \le x_i \le 500$$

$$\min(f_7) = f_7(420.9687, 420.9687) = -837.9657.$$

8. Generalized Rastrigin's Function

$$f_8(x) = \sum_{i=1}^{2} [x_i^2 - \cos(2\pi x_i) + 10], \qquad -5.12 \le x_i \le 5.12$$

$$\min(f_8) = f_8(0,0) = 0.$$

9. Ackley's Function

$$f_9(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{2} \sum_{i=1}^{2} x_i^2} \right)$$

$$- \exp\left(\frac{1}{2} \sum_{i=1}^{2} \cos 2\pi x_i \right) + 20 + e,$$

$$-32 \le x_i \le 32, \qquad \min(f_9) = f_9(0,0) = 0.$$

10. Generalized Griewank Function

$$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1,$$

$$-10 \le x_i \le 10, \qquad \min(f_{10}) = f_{10}(0,0) = 0.$$

11. Shubert Function

$$f_{11}(x) = \Big\{ \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \Big\} \Big\{ \sum_{i=1}^{5} i \cos[(i+1)x_2 + i] \Big\},$$

$$-10 \le x_i \le 10, \quad \min(f_{11}) = -186.731$$

with 18 global minima and 720 local minima.

12. Six-Hump Camel-Back Function

$$f_{12}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$$

$$-5 \le x_i \le 5, \quad \min(f_{12}) = f_{12}(x_{min}) = -1.0316$$

$$\text{for} \quad x_{min} = (0.08983, -0.7126), (-0.08983, 0.7126).$$

13. Branin Function

$$f_{13}(x) = \Big(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \Big)^2$$

$$+10\Big(1 - \frac{1}{8\pi}\Big)\cos x_1 + 10,$$

$$-5 \le x_1 \le 10, \quad 0 \le x_2 \le 15,$$

$$\min f_{13} = f_{13}(x_{min}) = 0.3979$$

$$\text{for} \quad x_{min} = (-3.142, 12.275), (3.142, 2.275), (9.425, 2.425).$$

14. Goldstein-Price Function

$$f_{14} = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$$

$$\times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$$

$$-2 \le x_i \le 2, \quad \min(f_{14}) = f_{14}(0,-1) = 3.$$