2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

Modified Inductive Search for Solving Global Optimization Problems

Hideo Kanemitsu†

†Hokkaido University of Education;
1–2 Hachiman-cho, Hakodate, Hokkaido 040-8567, Japan.
Email: †kanemitsu.hideo@h.hokkyodai.ac.jp

Abstract—

Inductive Search for solving global optimization problems has attracted much attention because it showed the best performance at the 1st ICEO. However, since details of this method are not clear, the method has not received much attention. We investigated details of the method and implemented the method. We propose a modified inductive search by using a deterministic one-dimensional global search. Finally, we evaluate the performance of the implemented method and that of proposed method.

1. INTRODUCTION

Global optimization problem: "(global) minimize $f(\boldsymbol{x}) \equiv f(x_1, \ldots, x_n) : \mathbb{R}^n \to \mathbb{R}$ under the constraint : $\boldsymbol{x} \in S \subset \mathbb{R}^n$" is widely formulated as mathematical models and is applied in many fields. Many methods for solving continuous global optimization problems have been proposed [1], and these methods are classified into *deterministic* framework, *stochastic* framework and *heuristic* framework.

The *deterministic framework* [2] repeatedly divides a given region into subregions and selects a subregion in which a global optimum is included. The *stochastic framework* [3] involve random sampling or a combination of random sampling and local search [4]. On the other hand, the *heuristic framework* (e.g., SA [5], GA [6], PSO [7]) have been intensively since the latter half of the 1980s and has been applied many fields. However, most of these methods have no guarantees to find a global optimal solution.

Searching spaces of all of those frameworks exponentially increase with increase in the number of dimensions in the problem (P). This phenomenon, known as the "*curse of dimensionality*", led to the abandonment of those search methods in favor of ones using some *a priori* knowledge or *priori* structure of the function f.

Inductive search was proposed[8] at "the 1st international contest on evolutionary optimisation" and the search has achieved an best result in this contest[9]. However, since details of the method were not clear, the method has not attracted much attention recently.

The purpose of this paper is to introduce the original *inductive search* and to reconstruct the *modified inductive search* using our univariate global search[13].

The remainder of the paper is organized as follows. A formulation and assumptions of the problem are given in **Sect. 2**. In **Sect. 3**, the algorithm of the inductive search is introduced. An algorithm of a deterministic inductive search using a one-dimensional global search is shown in **Sect. 4**. Finally, concluding remarks are given in **Sect. 5**.

2. PRELIMINARIES

A global optimization problem that minimizes (min.) an objective function $f : \mathbb{R}^n \to \mathbb{R}$ subject to (s.t.) constrains $D^n \subset \mathbb{R}^n$ is formulated as follows

$$(\mathrm{P}^n) \begin{cases} \text{min.} : & f(\boldsymbol{x}) \equiv f(x_1, x_2, \ldots, x_n), \\ \text{s.t.} \ : & (x_1, x_2, \ldots, x_n) \in D^n \equiv \prod_{i=1}^{n} [L_i, U_i] \subset \mathbb{R}^n. \end{cases}$$

In this problem we assume that objective function f is a Morse function, that is, f is 2nd continuous differentiable and Hessian matrix $\nabla^2 f(\boldsymbol{x}^*)$ at critical point $\boldsymbol{x}^*$ (s.t. $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$) is non-degenerate (i.e., $|\nabla^2 f(\boldsymbol{x}^*)| \neq 0$). Then, the following two properties hold: a) all local minima of the problem (P^n) are isolated and b) f has a convex region around a critical point $\boldsymbol{x}^*$.

3. INDUCTIVE SEARCH

3.1. Algorithm and its Implementation

The most novel idea of the method is to solve a subproblem (P^k) ($k = 1, 2, \ldots, n$) *inductively* by increasing the number of variables. The pseudo-C++ implementation[1] of the overall structure of the idea is given as follows.

```
1  #include "t1.c"          // t1: Sphere nv. problem
2  main() {
3    int iter_count=0, int n;
4    for (int i=0; i<n; i++) oracle(i+1);
5  }
```

Fig.1: Main code for calling inductive search `oracle(..)`.

In the above code, the argument `i+1` of `oracle(i+1)` denotes the number of dimensions on procedure `oracle`, and `i` increases by 1 until $n-1$. For a problem, this is easy to achieve even by till treating them as black boxes, because the test function is defined in terms of two parameters, the number of dimensions and a vector of the input variables:

$$f(x_1), \ f(x_1, x_2), \ldots$$

[1] Later codes are simplified codes of the original code[1, 2)] with maintenance of logical structure.

At a later stage, i.e., when solving $f(x_1,\ldots,x_i)$, the oracle can "update" a previous answer by changing the values of x_1 to x_{i-1}. This is necessary because when the oracle solves $f(x_1,\ldots,x_{i-1})$, it has no knowledge of how this function will be updated to $f(x_1,\ldots,x_i)$.

The 1st line of Fig.1 is an include file for *Sphere problem*, and the C-code of the file "`t1.c`" is shown in Fig.2.

```
1  #define STP_C 0
2  #define LEARN 0
3  #define L -5.0
4  #define U 5.0
5  float f(float *x,int nv) {
6    int i; float S;
7    for (S=0.0, i=0; i<nv; i++) S+=x[i]*x[i];
8    return (S);
9  }
```

Fig.2.: Content of C-code:`t1.c` for Sphere problem

Bilchev[8] proposed a simple (deterministic) version of `oracle` that obtained very good results in the 1st ICEO test problems. The pseudo-C++ implementation of the basic algorithm is shown in Fig.3.

```
1  void oracle(int nv) {
2    int count=0;
3    float xmin,FMIN=1e30,xmin,XMIN;
4    float ax=L,cx=U,bx=(ax+cx)/2;
5    sortseq<float,Interval> Pop; seq_item S;
6    Interval I; I.L=ax; I.U=cx;
7    Pop.insert(cx-ax,I);
8  /* Global learning */
9    while (1) {
10     S=Pop.max();  Pop.del_item(S); I=Pop.inf(S);
11     ax=I.L; cx=I.U; bx=(ax+cx)/2;
12     fmin=bren(ax,bx,cx,f1v,TOL,&xmin);
13     if (fmin < FMIN) { FMIN=fmin; XMIN=xmin; }
14       /* --- omission --- */
15       count++;
16       if (count > STP_C) break;
17   }
18 /* Local learning */
19   x[nv-1]=xmin;
20   if ( (LEARN) && (nv > 1) ) local_learn(fl,x,nv);
21 }
```

Fig.3: An implementation of the oracle.

In this code, the 5th, 7th and 10th lines denote services of LEDA[12]. If `STP_C` = 0, then the 11th line is equivalent to "`ax=L; cx=U; bx=(L+U)/2;`" from the 4th and 6th lines. In this case, these LEDA steps are unnecessary.

`oracle(..)` consists of two main steps:

S1) *Global learning*, which is a search for a better solution at the current dimension than the previous best solution(s);

S2) *Local learning*[10], which an `nv`-(sub) dimensional local search:`losrch(..)`.

In this implementation, the global learning is a series of calls Brent's local optimizer routine:`bren(..)`, and its external specification of the routine is as follows:

```
float bren(float ax, float bx, float cx,
           float (*f1v)(float), float TOL, *xmin);
```

where `(*f1v)(float)` is a routine for computing the function value of one variable. `ax`, `bx` and `cx` are 3-neighbor points. The routine `bren(..)` has finished, local minimum: `xmin` and its function value:`fmin` are returned at 12th line.

C-code file:"`f1v.c`" of C-function is shown Fig.4.

```
1  float f1v(float y) {
2    x[nvDim-1] = y;
3    float res = f(x,nvDim);
4    return res;
5  }
```

Fig.4: An auxiliary routine of a one-dimensional function

3.2. Problems of an Inductive Search

Since the inductive search has not been followed, a pseudo-code is difficult to implement for the following reasons.

1) These codes use many global constants (e.g., `STP_C`, `LEARN`, `L` and `U`) and variables (e.g., `nv` and `x[]` in **Fig 4.**).
2) Searching regions `L` and `U` are all fixed for any test problem of files "`tj.c`" ($j = 1,2\ldots,5$).
3) `oracle(..)` needs steps supported by LEDA[12].
4) If `STP_C` = 0, then the above LEDA-steps are unnecessary and *global learning* is not performed.

4. MODIFIED INDUCTIVE SEARCH

4.1. One-Dimensional Global Search Algorithm

The algorithm finds the minimum x_i^{**} of i-th variables and its function value $\underline{f}^{**}$for an objective function $f_i(x)$ at the i-th variable on a closed interval $D_i \equiv [L_i, U_i]$ for a given step size h and an already found global minimum $\boldsymbol{x}_{i-1}^{**} = (x_1^{**},\ldots,x_{i-1}^{**})$. The steps of the algorithm are as follows.

$(f_i^{**}, x_i^{**}) \leftarrow Go\_1DimSrch(f_i, D_i, h, \boldsymbol{x}_{i-1}^{**})$;

GL1. [Initialize]
$X_b \leftarrow \emptyset$; $F_b \leftarrow \emptyset$; $N \leftarrow \lceil (U_i - L_i)/h \rceil$;
$f^{(0)} \leftarrow f_i(L)$; $f^{(1)} \leftarrow f_i(L+h; \boldsymbol{x}_{i-1}^{**})$; $f_i^{**} \leftarrow \infty$.

GL2. [Find three neighboring points with bracketing of a local minimum]
for $j = 2$ **to** N **do**
  $x^{(j)} \leftarrow L + j \cdot h$; $f^{(j)} = f_i(x^{(j)}; \boldsymbol{x}_{i-1}^{**})$;
  if $f^{(j-2)} > f^{(j-1)}$ **and** $f^{(j-1)} < f^{(j)}$ **then**
    $X_b \leftarrow X_b \cup \{(x^{(j-2)}, x^{(j-1)}, x^{(j)})\}$;
    $F_b \leftarrow F_b \cup \{(f^{(j-2)}, f^{(j-1)}, f^{(j)})\}$;
    if $f^{(j-1)} < \tilde{f}_i^{**}$ **then** $\tilde{f}_i^{**} \leftarrow f^{(j-1)}$; $\tilde{x}_i^{**} \leftarrow x^{(j-1)}$; **fi**.
  fi.
od;

GL3. [Apply local minimization]
Apply univariate local minimization:$LoMin1v(\cdot)$.
$(f_i^{**}, x_i^{**}) \leftarrow LoMin1v(F_b, X_b, f_i, \tilde{f}^{**}, \tilde{x}^{**}, \mathtt{TOL})$;
return (f_i^{**}, x_i^{**});

In a one-dimensional global search $Go\_1DimSrch(\cdot)$, the following property holds[13].

Property 1 Let the lower unimodal region of the global minimum $\underline{x}^{**}$ of $f(x)$ on an interval $[L,U]$ be $Ru(\underline{x}^{**})$, where $Ru(\underline{x}^{**})$ is defined as the maximum region. Then, if $h \le 1/2 \cdot \min\{\underline{x}^{**} - a, b - \underline{x}^{**}\}$ holds, the algorithm $Go\_1DimSearch(\cdot)$ always finds the global minimum $\underline{x}^{**}$ of function $f(x)$.

An example of unimodal region $Ru(\underline{x}^{**})$ and step size h is shown in Fig.5.

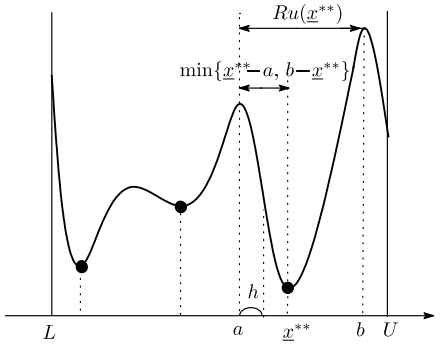

Fig. 5: Unimodal region $Ru(\underline{x}^{**})$ and step size h

4.2. Modified Inductive Search Algorithm

The algorithm finds the global minimum $\underline{x}^{**}$ and its function value $\underline{f}^{**}$ for an objective function $f(x)$ of n-variables with searching region $D^n \equiv \prod_{j=1}^{n} D_j \equiv [L_j, U_j]$ for a given step size h. The steps of the algorithm are as follows.

$(\underline{f}^{**},\ \underline{x}^{**}) \leftarrow Go\_nDimSrch(f,\ n,\ D^n,\ h,\ \mathtt{TOL})$;

G1. [Perform one-dimensional global search]

for $n_v = 1$ **to** n **do**

$(\tilde{f}^{**}_{n_v},\ \tilde{x}^{**}_{n_v}) \leftarrow Go\_1DimSrch(f_{n_v},\ D_{n_v},\ h,\ \tilde{\boldsymbol{x}}^{**}_{n_v-1})$;

$\tilde{\boldsymbol{x}}^{**}_{n_v} \equiv (\tilde{x}^{**}_1, \ldots, \tilde{x}^{**}_{n_v}) \leftarrow \tilde{\boldsymbol{x}}^{**}_{n_v-1} \cup \{\tilde{x}^{**}_{n_v}\}$; **od**;

G2. [Apply local minimization]

Apply local minimizer:$LoMin(\cdot)$ to starting point $\tilde{\boldsymbol{x}}^{**}_{nv}$ and its function value $\tilde{f}^{**}_{nv}$.

$(\underline{f}^{**}, \underline{x}^{**}) \leftarrow LoMin(f, n_v, \tilde{f}^{**}_{nv}, \tilde{\boldsymbol{x}}^{**}_{nv}, \mathtt{TOL})$;

return $(\underline{f}^{**}, \underline{x}^{**})$;

4.3. Comparisons between the Original Inductive Search and Our Modified Search

We show comparisons between the original inductive search algorithm (O) and our algorithm (M) as follows.

(O) { Number of calling `oracle(..)` (see Fig.1) : n
Number of calling `bren(..)` (see Fig.3) : $n\cdot(\mathtt{STP\_C}+1)$

(M) Number of calling $Go\_1DimSrch(..)$: n

(O) Number of calling `local_learn(..)` : $n \cdot \mathtt{LEARN}$,
where the value of `LEARN` is 0 or 1.

(M) Number of calling LoMin`(..)` : 1

5. NUMERICAL EXAMPLES

5.1. Benchmark Functions of 1st ICEO[9]

The following five objective functions with two cases of dimensions ($n = 5, 10$) at the 1st ICEO are presented.

(1) Sphere (unimodal, separable):

$$(P^n_S)\begin{cases} \text{min.:} & f(\boldsymbol{x}) = \sum_{i=1}^{n}(x_i - s)^2, \\ \text{s.t.:} & \boldsymbol{x}\in[-5,5]^n;\ s=1,\ n=5,10. \end{cases}$$

(2) Griewank (multimodal with convex skeleton, non-separable):

$$(P^n_G)\begin{cases} \text{min.:} & f(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n}(x_i-s)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i-s}{\sqrt{i}}\right)+1, \\ \text{s.t.:} & \boldsymbol{x}\in[-600,600]^n;\ s=100,\ n=5,10. \end{cases}$$

(3) Shekel (multimodal, non-separable):

$$(P^n_S)\begin{cases} \text{min.:} & f(\boldsymbol{x}) = -\sum_{i=1}^{30}\frac{1}{\|\boldsymbol{x}-\boldsymbol{a}_i\|^2 + c_i}, \\ \text{s.t.:} & \boldsymbol{x}\in[0,10]^n;\ n=5,10. \end{cases}$$

(4) Michalewicz (multimodal, separable):

$$(P^n_M)\begin{cases} \text{min.:} & f(\boldsymbol{x}) = -\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right), \\ \text{s.t.:} & \boldsymbol{x}\in[0,\pi]^n;\ n=5,10. \end{cases}$$

(5) Langerman (multimodal, non-separable):

$$(P^n_L)\begin{cases} \text{min.:} & f(\boldsymbol{x}) = -\sum_{i=1}^{30} c_i\left(e^{\frac{1}{\pi}\|\boldsymbol{x}-\boldsymbol{a}_i\|^2}\cos\left(\pi\|\boldsymbol{x}-\boldsymbol{a}_i\|^2\right)\right), \\ \text{s.t.:} & \boldsymbol{x}\in[0,10]^n;\ n=5,10. \end{cases}$$

5.2. Implementation

The inductive search and its modified search are implemented in programming language C (MinGW gcc 3.4.5). All numerical experiments for which results are shown in this paper were carried on an Lenovo note PC Think Pad X250 (2.6GHz Intel Core i7-5600) with double precision.

Setting parameters of inductive searches for five test functions are shown the Table 1.

Table 1: Setting parameters for inductive searches

Function	`STP_C`	`LEARN`	h
Sphere	0	0	1
Griewank	0	0	1.2
Shekel's foxholes	1	1	0.2
Michalewicz	10	0	0.2
Langerman	2	1	0.2

In this table, if the parameter `STP_C` = 0, then *global learning* becomes a *local search* because of one-dimensional local search `bren(..)` executing only one for each calling `oracle(nv)`, (nv=1,2,...,n) from **sect. 3.2**. From **sect. 4.3**, the number of calling `bren(..)` is n in this case.

5.3. Comparison between Inductive Searches and the Other Methods

Comparisons between the number of callings in the original inductive searches and the number of callings in the other methods is shown in Table 2.

Table 2: Benchmark results for the inductive search and the other 7 methods in the 1st ICEO w.r.t. the number of callingw (N^c:$\tilde{N}^c$: our implemented inductive search, the original inductive search and N^c_r of the other 7 methods)

Function	n	$\tilde{N}^c/N^c$	N^c_r
Sphere	5	25/20	243-12,218
Sphere	10	50/40	243-85,692
Griewank-1	5	68/41	5,765-2,977,996
Griewank-2	10	140/79	6,446-2,110,889
Shekel	5	415/74	6,318-451,992
Shekel	10	853/120	6,075-4,440,948
Michalewicz	5	183/120	1,877-60,219
Michalewicz	10	448/501	10,083-20,233,341
Langerman	5	471/176	4,131-232,496
Langerman	10	892/372	26,973-15,727,653

We can see that the number of callings for both inductive searches are much smaller than those of the other 7 methods.

5.4. Result for Original Inductive search in ICEO and Our Implemented Inductive Search

Benchmark result of the original inductive search in the 1st ICEO[9] and our inductive search are shown in Table 3.

Table 3: Benchmark results of two inductive searches w.r.t. obtained minimal function values and the number of callings ($\underline{f}^{**}(N^c)$: original and $\tilde{\underline{f}}^{**}(\tilde{N}^c)$: our implementation.).

Function	n	$\underline{f}^{**}(N^c)$	$\tilde{\underline{f}}^{**}(\tilde{N}^c)$
Sphere	5	3.88×10^{-15} (20)	0 (25)
Sphere	10	7.10×10^{-15} (40)	0 (50)
Griewank	5	7.99×10^{-6} (41)	1.09×10^{-1} (68)
Griewank	10	1.31×10^{-6} (79)	6.20×10^{-1} (140)
Shekel	5	−10.327 (74)	−10.400 (415)
Shekel	10	−10.101 (120)	−10.208 (853)
Michalewicz	5	−4.69 (120)	−4.49 (183)
Michalewicz	10	−9.66 (501)	−8.75 (448)
Langerman	5	−1.499 (176)	−0.965 (471)
Langerman	10	−1.499 (372)	−0.076 (892)

The original inductive search finds global minima for all ten problems, but our implemented inductive search cannot find global minima for five problems.

Since Griewank's problem is a multimodal function, and the parameter `STP_C` = 0 and `LEARN` = 0 (i.e., *global and local learning* of an inductive search is not a valid.), our implemented search fails to find the global minimum $\boldsymbol{x}^{**} = (1,\ldots,1)$. On the other hand, Original code"t2.c" of Griewank's function set $s = 0$, the original search `oracle(..)` can find the global minimum `xmin` = 0 because `bx=(L+U)/2=0`.

5.5. Results of Inductive Search and Modified Inductive Search

The result between our implemented inductive search and our modified inductive search are shown Table 4.

Table 4: Benchmark results of implemented inductive search and modified inductive searches w.r.t. obtained minimal function values and the number of calling ($\tilde{\underline{f}}^{**}(\tilde{N}^c)$: inductive search and $\underline{f}^{**}_M(\tilde{N}^c_M)$: modified inductive search.).

Function	n	$\tilde{\underline{f}}^{**}(\tilde{N}^c)$	$\tilde{\underline{f}}^{**}_M(\tilde{N}^c_M)$
Sphere	5	0 (**25**)	0 (61)
Sphere	10	0 (**50**)	0 (121)
Griewank	5	1.09×10^{-1} (**68**)	$\mathbf{3.20\times10^{-11}}$ (5,043)
Griewank	10	6.20×10^{-1} (**140**)	$\mathbf{2.12\times10^{-10}}$ (10,167)
Shekel	5	−10.404 (415)	−10.404 (**315**)
Shekel	10	−10.208 (853)	−10.208 (**510**)
Michalewicz	5	−4.490 (183)	**−4.689** (**169**)
Michalewicz	10	−8.745 (448)	**−9.160** (**407**)
Langerman	5	−0.01 (362)	**−0.820** (**355**)
Langerman	10	-3.4×10^{-5} (807)	**−0.813** (**682**)

The number of function evaluations of an implemented inductive search for Sphere and Griwank functions is very small, because the parameters of Sphere and Griewank functions are set to `STP_C` = 0 and `LEARN` = 0 (i.e., *global and local learning* of an inductive search is not a valid.)

Our modified inductive search finds better minima for six problems and with a smaller the number of function evaluations for six problems than does the implemented inductive search.

6. Conclusion

In this paper, we showed a clearer and more detailed algorithm based on C-code or C++-code of an *an inductive search* by Bilchev's implementation. We propose an algorithm for a modified inductive search using a one-dimensional global search and we evaluated the algorithm. Both algorithms become deterministic methods because a random number generator is not used. Moreover, the one-dimensional global search of our modified method has a theoretical guarantee that the method can find a global minimum. Numerical examples show that our modified method can be reliably find a global minimum.

References

[1] Neumaier, A.: "Global Optimization,"2015.
1) `www.mat.univie.ac.at/~neum/glopt/`
2) `www.mat.univie.ac.at/~neum/glopt/software_g.html`
3) `www.mat.univie.ac.at/~neum/glopt/bilchev/Readme`

[2] C.A. Floudas: "Deterministic Global Optimization: theory, methods and applications," Kluwer, 2000.

[3] Törn, A. and Žilinskas, A. : "Global Optimization," Lecture Notes in Computer Science. 350, Springer-Verlag, 1989.

[4] Bazaraa, M. S. et al.: "Nonlinear Programming —Theory and Algorithms— (3rd ed.)," Wiley-Intersci., 2006.

[5] Kirkpatrick, S. et al. : "Optimization by Simulated Annealing," Science, Vol.220, 671–680, 1983.5.

[6] Goldberg, D.E. : "Genetic Algorithms in Search" Addison-Wesley, 1989.

[7] Kennedy, J., Eberhart, R.C: "Particle Swarm Optimization," IEEE, Proc. of IEEE International Conference on Neural Networks, pp.1942–1948, 1995.

[8] Bilchev, G. and Parmee, I.: "Inductive Search," Proc. of 1996 IEEE International Conference on Evolutionary Computation (ICEO '96), pp.832–836, 1996.

[9] Bersini, H. et el. : "Results of the First International Contest on Evolutionary Optimisation," Proc. of 1996 IEEE International Conference on Evolutionary Computation (ICEO '96), pp.611–615, 1996.

[10] Yuret, D: "From GA's to efficient optimization," MSC thesis MIT May 1994.

[11] Brent, R. P.: "Algorithms for Minimization without Derivatives," Prentice Hall, 1973.

[12] Algorithm Solutions GMBH
`www.algorithmic-solutions.com/leda/eval/`

[13] Kanemitsu, H et al. A Global Minimization Method for Separable Multimodal Functions Using a One-Dimensional Global Search, Proc. of 2006 International Symposium on Nonlinear Theory and its Applications, pp.875-878, 2006.