*The 2013 World Congress on*
***Advances in Structural Engineering and Mechanics (ASEM13)***
*Jeju, Korea, September 8-12, 2013*

# New Direct Search Method for the Discrete Structural Optimization with Nonlinear Constraints and Integer Index variables

*Hongwoo Lee[1)]

[1)] *Department of Architecture, Seowon University, Cheongju, Chungbuk 361-742, Republic of Korea*
[1)] hwlee@seowon.ac.kr

## ABSTRACT

This study proposes a new direct search method for discrete structural optimization with nonlinear constraints, whose tentative name is Near-Up Search algorithm. The algorithm combines 'Near Points Search' module and 'Upward Direction Search' module. The algorithm uses the integer index variables which were newly defined. Near Points Search module is used for searching the adjacent design points to find a better solution. Upward Direction Search module is used for searching the design points which make the value of objective function bigger than the value by the current points to find a feasible solution. The combinational operation of the two modules, Near-Up Search algorithm can find the feasible and near optimal solution without gradient information. The effectiveness of the proposed algorithm is shown through the use of a classical truss optimization example.

## 1. INTRODUCTION

Most of the conventional numerical optimization techniques treat the design variables as continuous things. Those techniques are inadequate for the structural designs using commercially available fabricated sections as members which should be expressed by discrete variables. To use the continuous optimization method in such structural design the discreteness should be ignored by employing the round-off technique, they may result in solutions far from optimum or even result in infeasible values. Furthermore, optimization problems of structures are controlled by nonlinear constraints. Therefore, it is very difficult to solve these complex problems with conventional continuous optimization techniques(Hwang etc. 2001).

In this study, a new direct search algorithm for discrete structural optimization with nonlinear constraints, Near-Up Search, is proposed. The simplicity of direct search can make us overcome the complexity of structural optimization. A new type of variables,

[1)] Professor

‘integer index variables’, is introduced for the new algorithm to formulate the discrete structural design problem. To demonstrate the ability of proposed algorithm, it is applied to a classical truss design. The results are compared with those of ALGA in MATLAB.

## 2. FORMULATION OF OPTIMIZATION PROBLEM

Let us consider the optimization problem with continuous variables and nonlinear constraints which is :

$$\text{minimize} \quad f(x) \tag{1.1}$$
$$subject\ to \quad c_i(x) \le 0 \qquad \text{for i} = 1, \cdots, m_c \tag{1.2}$$
$$b_l \le x \le b_u \tag{1.3}$$

where $f(x)$ represents the objective function which is $f: R^n \to R$, $c_i(x)$ are the nonlinear constraints, $x$ is design variables, $b_l$ and $b_u$ are the lower and upper bound of variables. The variable is a vector $x \in R^n$ and expressed as a point in design space. In sizing optimization problems, the objective function is taken as the total weight of trial structure and the stress and displacement constraints are considered. The variables are the sectional area of members of structure.

In this paper I want to introduce a new type of variables, ‘integer index variables’, which has very important role in the new direct search method, Near-Up Search. Let us assume that the sectional areas in Eq. 2 can be used in the sizing optimization of a structure.

$$\text{A}(in^2) = \{1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09\} \tag{2}$$

The integer index variable $x_i$ can be defined as the index of a element in the area set in Eq. 2 for the sectional area of the $i$th member in the structure. So the variable must be an integer between 1 and 10 because the number of possible elements shown in the area set is 10.

$$1 \le x_i \le 10, \quad x_i \in I \tag{3}$$

In this definition, ‘$x_5 = 3$’ means that the area of 5th member of the structure is the 3rd element of the area set A in Eq. 2 and it is 1.99$(in^2)$.

Using this integer index variable, the sizing optimization problem of a structure can be expressed in the following form:

$$\text{minimize} \quad f(x) \tag{4.1}$$
$$subject\ to \quad c_i(x) \le 0 \qquad \text{for i} = 1, \cdots, m_c \tag{4.2}$$
$$x = \{x_1, x_2, x_3, \cdots, x_n\}, \quad x \in I^n \tag{4.3}$$
$$1 \le x \le x_u \tag{4.4}$$

where, the objective function $f(x)$ is $f: I^n \rightarrow R$, $x$ is integer index design variables, $n$ is the number of members which compose the structure, $x_u$ is the upper bound of variables. In this paper, this discrete structural optimization problem with integer index variables and nonlinear constraints expressed in Eq. 4 will be solved by Near-Up Search algorithm which will be very effective technique due to the problem's unique formulation using integer index variables.

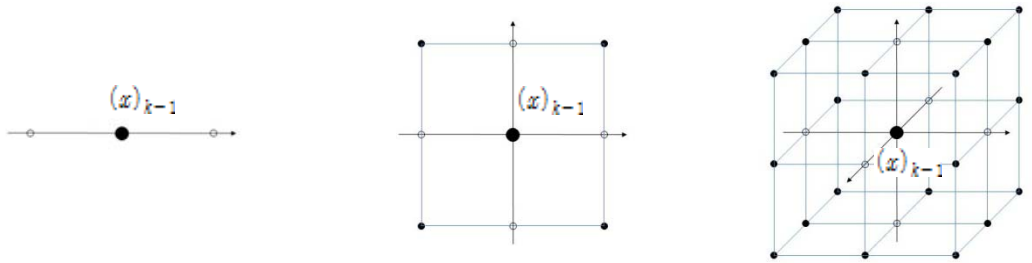
## 3. NEAR-UP SEARCH

Direct search methods have been known since at least the 1950s. However, by the early 1970s, these methods were largely dismissed by the mathematical optimization community and disappeared from most of its literature for the lack of theoretical background(Tamara etc. 2003).

But the simplicity of direct search method can be a powerful character to overcome the complexity of structural optimization. So, in this paper, a new direct search algorithm, Near-Up Search, is proposed for solving the discrete optimization problem with nonlinear constraints and integer index variables. Near-Up Search is a search algorithm combines 'Near Points Search' module and 'Upward Direction Search' module.

### 3.1 Near Points Search Module

Near Points Search is a process searching the neighbor points from the current design points. 'The near points' are defined as the neighbor points which can be obtained by increasing or decreasing the value of one or more components of the current point by only 1. By the characteristic of the integer index variables, changing the value of a component in a variable means changing the index of sectional area of a member in sizing optimization. The near points are illustrated in Fig. 1.




a. One dimension b. Two dimension c. Three dimension

Fig. 1 Spacial dimensions and the near points

A near point, $(x)_{near}$, can be obtained by Eq. 5.2 from the current point, $(x)_{k-1}$.

$$(x)_{k-1} \in I^n \tag{5.1}$$

$$(x)_{near} = (x)_{k-1} + d_j^{near} \tag{5.2}$$

$$d_j^{near} \in D_{near} \text{ and } d_j^{near} \in I^n \tag{5.3}$$

$$D_{near} = \{d_j^{near}\}\{d_j^{near} | d_j^{near} \; is \; random \; vector \; composed \; of \; -1, 0, 1.\} \qquad (5.4)$$

Where, $d_j^{near}$ called 'Near-vector' is a direction vector to find a neighbor point, $D_{near}$ is a set of Near-vector. The vector, $d_j^{near}$, can be created easily by using MATLAB function, such as 'fix(rand(1,n)*3)-1', but it requires special care in handling the Near-vector because the near points should be in the design space between the upper and lower bound.

After a near point is created, the two points $(x)_{near}$ and $(x)_{k-1}$ may be compared with the function values as follows: $(x)_{near} \prec (x)_{k-1}$ if ( $f((x)_{near}) < f((x)_{k-1})$ and $c_i((x)_{near}) \leq 0 \;\; for \; all \; i$ ). That is, $(x)_{near}$ is "better" than $(x)_{k-1}$ because it yields a lower objective function value and satisfies all constraints. If the better point is found the current point is updated as $(\boldsymbol{x})_{\boldsymbol{k}} = (\boldsymbol{x})_{\boldsymbol{near}}$, and the best point is also changed, $(x)_{best} = (x)_k$. If not, then try another Near-vector, $d_j^{near}$, and another near point $(x)_{near}$, to compare with the current point. The pseudo-code for the Near Points Search process is shown in Table. 1.

Table. 1 Pseudo-code for the Near Points Search

```
For nearID = 1, n_near
    (x)_near = (x)_{k-1} + d_j^near
    If ( f((x)_near) < f((x)_{k-1}) and c_i((x)_near) ≤ 0  for all i )
        While  c_i((x)_near) ≤ 0  for all i
            (x)_k = (x)_near
            (x)_near = (x)_{k-1} + d_j^near
        End
        If ( (x)_best = 0  or  f((x)_k) < f((x)_best)
            (x)_best = (x)_k
        End
        Exit For loop
    Else if ( nearID = n_near )
        (x)_k = (x)_{k-1}
    End
End
```

Where, $n_{near}$ is the maximum number of iteration, nearID is an index of iteration in Near Points Search

3.2 Upward Direction Search Module

Upward Direction Search is a process searching for the point which satisfying all constraints from the current point. This process is performed when the current point is infeasible or the Near Point Search can't find the better point.

In structural optimization, if the bigger sectional areas are used, the possibility for satisfying the stress and strain constraints is increased. So the bigger value of design variables has more possibility to satisfy the constraints. That's the reason why in Upward Direction Search process the design variables are slowly increased by using Eq. 6 to find feasible points.

$$(x)_{k-1} \in I^n \quad (6.1)$$
$$(x)_{up,0} = (x)_{k-1} \quad (6.2)$$
$$(x)_{up,j} = (x)_{up,j-1} + d_j^{up} \quad (6.3)$$
$$d_j^{up} \in D_{up} \text{ and } d_j^{up} \in I^n \quad (6.4)$$
$$D_{up} = \{d_j^{up} | d_j^{up} \text{ is random vector composed of } 0 \text{ or } 1.\} \quad (6.5)$$

Where, $d_j^{up}$ called 'Up-vector' is a direction vector to find a feasible point and the vector increase the value of variables. The vector, $d_j^{up}$, can be created easily using MATLAB function, such as 'fix(rand(1,n)*2)', but it also requires special care in handling Up-vector that the new points, $(x)_{up,j}$, should be in the design space between the upper and lower bound.

This process will be terminated when a new points, $(x)_{up,j}$ is found which satisfy all constraints($c_i((x)_{near}) \leq 0 \quad for\ all\ i$ ). Then the current point is updated as $(\boldsymbol{x})_{\boldsymbol{k}} = (\boldsymbol{x})_{\boldsymbol{up,j}}$. The pseudo-code for the Upward Direction Search process is shown in Table. 2.

Table. 2 Pseudo-code for the Upward Direction Search

```
(x)_{up,0} = (x)_{k-1}
For j = 1, n_up
    (x)_{up,j} = (x)_{up,j-1} + d_j^up
    If (c_i((x)_near) ≤ 0  for all i )
        (x)_k = (x)_{up,j}
        If ( (x)_best = 0 )
            (x)_best = (x)_k
        End
        Exit For loop
    End
End
if ( j = n_up )
    (x)_k = (x)_{up,j}
End
```

This Upward Direction Search process can also be used to escape the local optimum points when the Near Points Search can't find a better point.

3.3 Near-Up Search Algorithm

Near-Up Search algorithm is illustrated in Fig. 2. Let $k$ serve as the index for each iteration and let $(x)_k \in R^n$ denote the design point of $k$th iteration, with $(x)_0$ denoting the initial guess. Generally $(x)_0$ is infeasible point and Upward Direction Search is performed by setting '$\mathrm{nearID} = n_{near}$' before Near Points Search is carried out at 1st iteration. Let $(x)_{best}$ denote the best point found at the current iteration. $(x)_k$ is updated at each iteration and $(x)_{best}$ is changed if a better point is found. The algorithm is

terminated when $k$ is bigger than the maximum number of iteration ($k \geq n_{iter}$) or the function value reaches the target value( $f((x)_k) < f_{target}$).

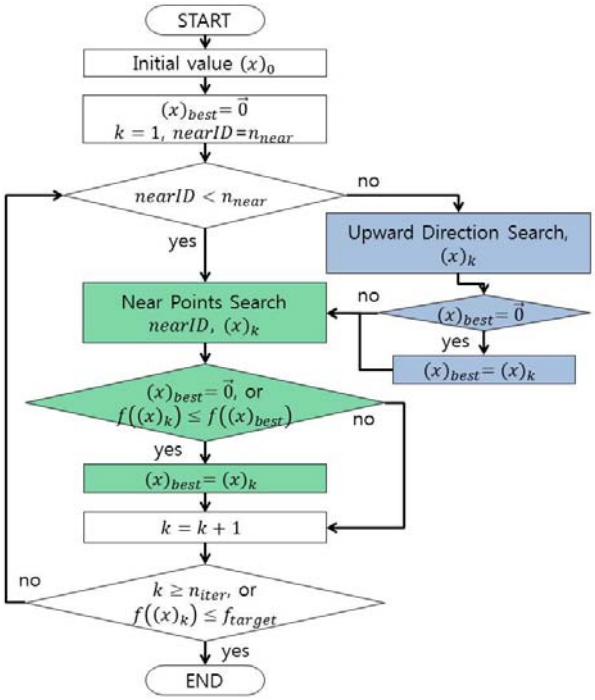



Fig. 2 Algorithm for the Near-Up Search

## 4. VERIFICATION THE VALIDITY OF NEAR-UP SEARCH

To verify the validity of Near-Up Search algorithm, it is applied to the optimization of 10-bar truss. The optimization results of Near-Up Search are compared with those of ALGA(Augmented Lagrangian Genetic Algorithm) included in the MATLAB(R2012a) program. The 6-node 10-bar truss model is shown in Fig. 3.

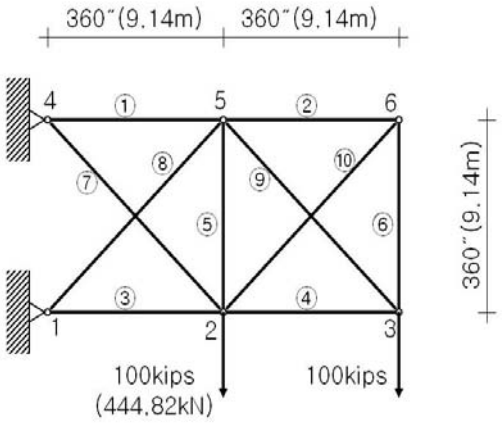



Fig. 3 10-bar truss

4.1 formulation of 10-bar truss design

In this problem the cross-section area for each 10 members in truss will be optimized towards the minimization of the total weight of truss. The cross-sectional areas can be used are:

$$\begin{aligned}\mathrm{A}(in^2) =\{&1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09,\\ &3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22,\\ &4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5,\\ &13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5,\\ &30.0, 33.5\}\end{aligned} \quad (7)$$

Let $x_i$ means the index variable for the area of the $i$th member in the truss and it would be an integer between 1 and 42 because the number of available area in the area set in Eq. 7 is 42.

$$1 \le x_i \le 42, \quad x_i \in I \quad (8)$$

In this definition, if $x_5 = 10$ means that the area of 5th member of the truss is 3.09$(in^2)$. The design variable $x$ is a vector that has 10 components because the truss is composed of 10 members.

Using this design variable $x$, the formulation of 10-bar truss design problem is as follows.

$$\text{minimize} \quad f(x) = \sum_{i=1}^{10} \rho v_i \quad (9.1)$$

$$subject\ to \quad g_j^{\sigma} \le 0 \quad \text{for j} = 1, \cdots, m_{\sigma} \quad (9.2)$$

$$g_k^{\delta} \le 0 \quad \text{for k} = 1, \cdots, m_{\delta} \quad (9.3)$$

$$x = \{x_1, x_2, x_3, \cdots, x_{10}\}, \quad x \in I^{10} \quad (9.4)$$

$$1 \le x_i \le 42 \quad \text{for i} = 1, \cdots, 10 \quad (9.5)$$

where, the density $\rho$ is the weight of the unit volume, $v_i$ is the volume of the $i$th member, $g_j^{\sigma}$ represents the stress constraints, $g_k^{\delta}$ represents the displacement constraints. The density is $\rho = 0.1 lb/in^3$, and the Young's modulus $E = 10000 ksi$. The allowable stress for each member is $\pm 25$ksi for both tension and compression. The allowable displacement on the nodes is $\pm 2$in, in the y direction.

The best result of 10-bar truss design found in Lee (2008) is shown in Table 3.

Table. 3 Solution of 10-bar truss design

| Member | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Area of Member$(in^2)$ | 33.5 | 1.62 | 22.9 | 14.2 | 1.62 | 1.62 | 7.97 | 22.9 | 22.0 | 1.62 |
| Integer index variable | 42 | 1 | 39 | 32 | 1 | 1 | 28 | 39 | 38 | 1 |
| Total weight of truss | 5490.7 lb | | | | | | | | | |

4.2 Results of design by ALGA

The value of parameters used in running ALGA in MATLAB : population size=50, maximum generations=500, stalling generation limit=100. The value of other parameters like the probabilities of crossover and mutation are using the default values assigned in MATLAB.

If the initial population is generated randomly in design space, ALGA would be terminated in 1st generation because ALGA couldn't find a feasible individual. So I assigned the maximum value of the variables to the initial population.

In 100 try, ALGA obtains nearly as good as the cross sections and total weight as shown in Table. 3, but can't find the exact same or better solution as that in Table. 3. The total weight of truss of best solution found by ALGA is 5503.93 lb. The solution was found in 229th generation after the functions were evaluated 1,171,536 times and the elapsed time is 940.47 seconds. The average weight of truss in 100 try was 5863.10 lb and the average elapsed time is 808.45 seconds. Almost every time ALGA was terminated by the stalling generation limit. The normal convergence history of ALGA is illustrated in Fig. 4.

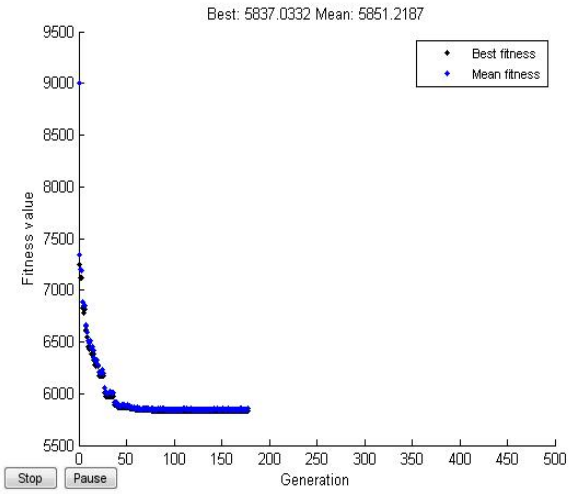



Fig. 4 Convergence history of ALGA

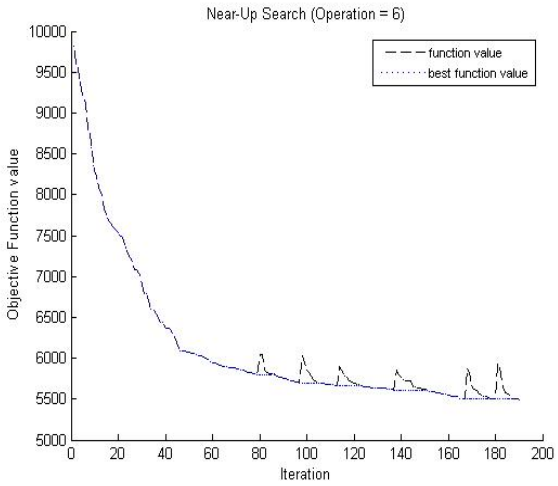



Fig. 5 Convergence history of Near-Up Search

4.3 Results of design by Near-Up Search

In the Near-Up Search algorithm, the initial values of design variables can be created randomly thanks to the Upward Direction Search module. The termination criteria is maximum number of iteration, $n_{iter} = 1000$, and this algorithm was performed 100 times to evaluate the average performance.

Every time in 100 try, Near-Up Search algorithm find the exact same solution shown in Table. 3. The average number of iteration to find the solution is 160 and the average elapsed time is 40.52 seconds. This results can be compared with those of ALGA and we can see that Near-Up Search is more effective than ALGA in this type of sizing optimization. The normal convergence history of Near-Up Search is illustrated in Fig. 5.

## 5. CONCLUSIONS

This study proposes a new direct search algorithm, Near-Up Search, is proposed for solving the discrete optimization problem with nonlinear constraints. The algorithm combines 'Near Points Search' module and 'Upward Direction Search' module, and it uses the integer index variables which were newly defined in this paper. The Near Points Search module is used for searching the adjacent design points to find a better solution. The Upward Direction Search is used to search for a feasible point from an infeasible current point and to escape the local optimum points. The combinational operation of the two modules, Near-Up Search algorithm can find the feasible and near optimal solution without gradient information and doesn't have to use the penalty function like augmented Lagrangian.

The proposed method is illustrated and tested by the minimum weight structural optimization of 10-bar truss. The results of the design show that the rate of convergence and the accuracy of solution of the proposed algorithm are much better than those of ALGA. It is obvious that, for the discrete structural optimization with nonlinear constraints, Near-Up Search algorithm is a very useful method.

## REFERENCES


Hwang, S., Cho, H., Han, S. (2001), Discrete Optimal Design of Truss Structures Using Genetic Algorithm, J. of the Computational Structural Engineering Institute of Korea, Vol. **14**(2), 97–106

Kolda, T. G., Lewis, R. M., Torczon, V. (2003), Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. SIAM review, Vol. **45**(3), 385–482

Lee, H. (2008), Direction Vector for Efficient Structural Optimization with Genetic Algorithm, J. of the Korean Association for Spatial Structures, Vol. **8**(3), 75-82

The MathWorks (2012), Genetic Algorithm and Direct Search Toolbox 2 : User's Guide, MATLAB ver. R2012a