Control
and Cybernetics
VOL. 19 (1990) No. 3–4

An algorithm for finding an ordered sequence of values of a discrete linear function

by

Z. IWANOW

Institute of Fundamental Technological Research
Polish Academy of Sciences
ul. Świętokrzyska 21
00-049 Warszawa, Poland

The algorithm for finding an ordered sequence of discrete linear function values is presented.It is possible to start with any value.The applied method does not need much memory and is effective.The algorithm can be treated as useful tool in discrete optimisation.

Introduction

The problem of structure optimization while considering for instance its weight or cost and taking into account the selection of its elements from a given finite catalogue leads to a discrete problem with a linear objective function and nonlinear constraints. An effective method for solving discrete problems is the enumeration of the solution set [1], [2], [3]. In most cases the full survey of the set of solutions has no practical value because the number of solutions is too large. Thus certain strategy for eliminating some solutions is necessary. A possible method [1] may consist in examining the solutions in such a way

that monotonicity of the values of the objective function is ensured. The first encountered solution satisfying the admissibility conditions is the solution of the problem. If we start our searching from the objective value corresponding to the continuous solution, we will be able to solve on Personal Computer even such problems for which the full enumeration method would be impossible to perform on large computers [2]. To achieve this we need a sufficiently effective algorithm for finding the aforementioned sequence of solutions. The algorithm presented in [1] is practically, except for very small problems, of no use because of its memory requirements and time consumption. Its essential weakness is the necessity to determine the whole sequence starting from the smallest value. The algorithms presented in this paper do not require to store intermediate solutions so they do not need much memory. They allow us to start with any value of the objective function. That means if we give any value we obtain these solutions (not necessary unique) for which the function takes its first value greater than the given one. The efficiency of such a method for finding solutions is implied by the fact that we browse some groups of solutions for which we can easily calculate the minimum of the corresponding function values. In practice in some cases the problems of structure optimization lead to problems in 0-1 variables. Therefore in this paper we describe separately the algorithm for 0-1 variables and for variables from a certain catalogue, although main assumptions of both algorithms are similar.

I. 0-1 variables

1 Problem statement

Define a function:

$$\begin{array}{ll} & f(\mathbf{x}) = \sum_{i=1}^{N} c_i x_i \\ \mathbf{x} = [x_1, \cdots, x_N]^T & x_i \in \{0,1\} \qquad c_i \in R \qquad i = 1, \cdots, N. \end{array} \tag{1}$$

Assume, that coefficients are orderd and distinct i.e.

$$\begin{array}{l} c_i \neq c_j \quad i \neq j. \\ c_i < c_{i+1} \quad i = 1, \cdots, N-1. \end{array} \tag{2}$$

The case of equal coefficients, see 2e.

Let us state the following problem :

Problem 1. Let a real number $w \in R$ be given. Find all the vectors $\bar{\mathbf{x}}$ satisfying the following conditions:

$$(f(\bar{\mathbf{x}}) > w) \wedge \bigwedge_{\mathbf{x} \neq \bar{\mathbf{x}}} (f(\mathbf{x}) \leq w \quad \vee \quad f(\mathbf{x}) \geq f(\bar{\mathbf{x}})) \tag{3}$$

This problem is equivalent to finding all the points, for which the function (1) takes the smallest possible value exceeding the number w fixed before.

2 An algorithm to find the numerical solution for the problem 1

a. A model and its certain properties

The function (1) is defined in 2^N different points. These points can be assigned to the vertices of a suitable graph which has a tree structure.

The graph construction and this assignment enable us to estimate the minimal and maximal values of the function corresponding to some groups of points and thus also to vertices. This property will be essentially used in the algorithm.

The tree corresponding to the problem 1 is constructed recursively in the following manner:

1. Choose the root and assign to it an index equal to 0.
2. The vertex with the index i is the parent of the vertices with the indices $i+1, i+2, \cdots, N$. The vertex with the index N has no children, it is a leaf.
3. If the index of a child is equal to i then weight of the edge between it and the parent is equal to c_i.

An example of such a tree for $N = 5$ is shown in Fig. 1.

The set of graph vertices can be divided into subsets called layers, according to the following rule:
to the i–th layer belong all the vertices which can be reached from the root in i steps.

From the construction of the tree follows, that there are exactly $N+1$ layers namely $0, 1, \cdots, N$. If a vertex lies in the L–th layer then a path from the root to this vertex passes through exactly L vertices including that of destination. To every vertex k_L of any layer with the number L we can assign a unique sequence of indices of vertices lying on the unique path from the root to k_L.

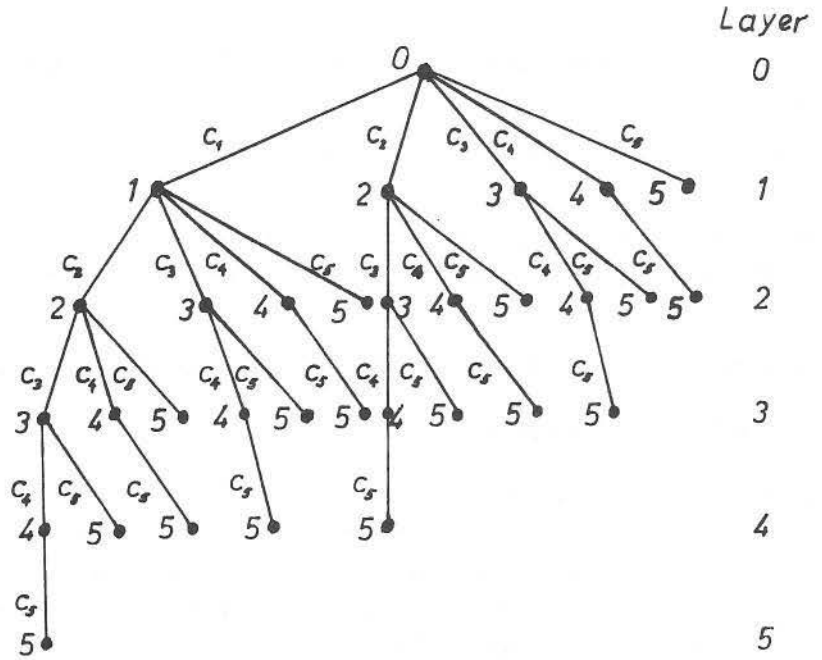

Figure 1

$$j_1^{k_L}, j_2^{k_L}, \cdots, j_L^{k_L} \tag{4}$$

Using (4) we can assign to every vertex a unique vector:

$$\mathbf{x}^{k_L} = \left[x_1^{k_L}, \cdots, x_N^{k_L}\right]^T \tag{5}$$

with the coordinates:

$$x_r^{k_L} = \begin{cases} 1 & r \in \left\{j_1^{k_L}, \cdots, j_L^{k_L}\right\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$r = 1, \cdots, N.$$

Using (6) we can assign to every vertex a unique value of the function (1), namely $f(\mathbf{x}^{k_L})$.

This value is also equal to the length of the path from the root to this vertex. From (4) - (6) follows that a vector assigned to any vertex from the L–th layer has exactly L components equal to 1, the remaining equal to 0. From the manner in which the graph has been constructed one can inductively prove that in the L–th layer there is exactly $\binom{N}{L}$ vertices. Otherwise one can say that there are

as many vertices in this layer as different vectors (5) with exactly L coordinates equal to 1. From the equality $\sum_{i=0}^{N} \binom{n}{i} = 2^N$ we deduce that the vertices of the constructed graph represent all the points belonging to the domain of the function (1). The assignment of the sequence of indices (4) to a vertex allows us to order lexicographicaly vertices of a layer. Let us consider two different vertices of the same layer L with their corresponding sequences

$$\begin{array}{l} j_1^{k_L}, j_2^{k_L}, \cdots, j_L^{k_L} \\ j_1^{l_L}, j_2^{l_L}, \cdots, j_L^{l_L} \end{array} \tag{7}$$

Define:

$$n := min\left(i | j_i^{k_L} \neq j_i^{l_L}\right) \tag{8}$$

Let us assume that this vertex is earlier which has smaller index with the number n, i.e.

$$\begin{array}{l} j_n^{k_L} < j_n^{l_L} \Longrightarrow k_L \prec l_L \\ j_n^{l_L} < j_n^{k_L} \Longrightarrow l_L \prec k_L \end{array} \tag{9}$$

The above ordering implies that the first vertex in the layer L is 1_L with the sequence of indices:

$$1, 2, \cdots, L \tag{10}$$

At the same time the value of the function (1) which corresponds to this vertex is the smallest one in the layer L. It is implied from the construction of the graph condition 2 and by (10), (6). Every vertex later than (10) must have one of the indices greater than (10). All the next indices have to be also greater than the corresponding indices from (10). Since the coefficients c_j, according to (2), form an increasing sequence, the value of the function (1) corresponding to the vertex (10) is the smallest value corresponding to the vertices in this layer. The same considerations imply that the last vertex in the layer $L > 0$ is the vertex whose sequence of indices is:

$$N - L + 1, \cdots, N - 1, N \tag{11}$$

The value of the function (1) corresponding to this vertex is the greatest of all the values assigned to the vertices of the layer L.

From (10) and (4)–(6) we get the following properties of the sequence of the values of the function (1) assigned to the first vertex in a layer:

1. if $c_j > 0$ $(j = 1, \cdots, N)$ – then this is an increasing sequence.

2. if $c_j < 0$ $(j = 1,...,\bar{i})$, $c_j > 0$ $(j = \bar{i}+1,\cdots,N)$ then the values corresponding to the layers $1,\cdots,\bar{i}$ form a decreasing sequence, while these corresponding to the layers $\bar{i}+1,\cdots,N$ – an increasing sequence.

Similarly, the sequence of the values of the function (1) assigned to the last vertices in the layer, have the following properties:

1. if $c_j > 0$ $(j = 1,\cdots,N)$ – then this is an increasing sequence.
2. if $c_j < 0$ $(j = 1,\cdots,\bar{i})$, $c_j > 0$ $(j = \bar{i}+1,\cdots,N)$ then the values corresponding to the first $(N-\bar{i})$ layers form an increasing sequence, while these corresponding to the layers $N-\bar{i}+1,\cdots,N$ a decreasing sequence.

Let us take any vertex from the layer L_1 which is earlier than the layer L, that is

$$L_1 < L \tag{12}$$

and a sequence corresponding to this vertex:

$$j_1^{m_{L_1}},\cdots,j_{L_1}^{m_{L_1}} \tag{13}$$

By the graph construction principle 2. and by the definition of the order (9), this vertex has its descendants in the layer L if

$$N - j_{L_1}^{m_{L_1}} \geq L - L_1 \tag{14}$$

By the order rules (9) in the layer the earliest descendant is the vertex whose corresponding sequence of indices is:

$$j_1^{m_{L_1}},\cdots,j_{L_1}^{m_{L_1}},j_{L_1}^{m_{L_1}}+1,\cdots,j_{L_1}^{m_{L_1}}+L-L_1 \tag{15}$$

while the latest one:

$$j_1^{m_{L_1}},\cdots,j_{L_1}^{m_{L_1}},N-(L-L_1)+1,\cdots,N \tag{16}$$

The first and the last vertices in a layer are special cases of the earliest and the latest descendant of the root $(L_1 = 0)$ in the layer L. The same reason as for the first vertex in a layer leads us to the following conclusion : let $v1$ be the vertex with the index sequence (15) and $v2$ - with the index sequence (16). Then $v1$ corresponds to the smallest and $v2$ to the greatest value of the function (1) among the values of all the descendants of the vertex (13) in the layer L. If the subsequent vertices $m_{L_1}, m_{L_1}+1,\cdots$ of the layer L_1 have their

descendants in L, then we get the following increasing sequences of values: the sequence corresponding to the earliest and the sequence corresponding to the last descendants of the subsequent vertices.

This property is substantial for the enumeration of vertices in a given layer. The properties we just described are essentially used in the algorithm. They allow us to treat and to examine the whole groups of vertices in the layer L as descendants of the vertex belonging to the previous layer.

The algorithm for finding the desired points consists of three main parts:

1. – determining the layer from which we start our search

2. – determining the first vertex in the layer for which the corresponding function value is greater than w.

3. – examining the subsequent vertices in the layer in order to find ones for which the excession of the value w is less than in the found vertex – then the modification of the vertex proceeds – or in order to find and to include the equivalent vertices.

The first two steps which are not time consuming, allow us to exclude some groups of vertices whose number is in certain cases quite considerable. Whereas the search in step 3 is performed in groups and there is a stopping criterion for it, so we do not need to continue the search process until the set of vertices becomes empty. Hence the reduced time consumption of the algorithm. On the other hand the storage requirements are determined by the number of equivalent vertices which should be kept in the memory. For the search itself we need actually to store only the sequence (4) defining the path to the first vertex of the group examined subsequently.

b. A layer with which we begin our search

The determination of the layer with which we begin our search and the strategy of the order of the search of the further layers depend on the specifics of a given problem.

If optimization concerns the problem in mechanics which has been modeled as a problem with $0-1$ variables then usually the number n of variables equal to 1 is known, e.g. in the optimization of bar structures n is the number of bars. In this case it suffices to examine the level with the number equal to n.

In the more general case the suitable layer can be found while considering the properties of the smallest and the greatest values assigned to vertices in the respective layers described in part a.

c. An algorithm for finding the first vertex in the determined layer with the function value exceeding a given value.

Let us fix a layer number L and a real number w. Let us state the following problem: find the earliest vertex with respect to the order (9) for which the corresponding value of the function (1) is greater than w.

After finding such a vertex we can exclude from our search process all the earlier vertices. In certain of the cases it can substantially reduce the number of the remaining vertices.

In the construction of the algorithm the properties (12)–(16) have been used. The algorithm has the folowing form:

fix : w - a given real number. We are looking for a vertex with the smallest possible value greater than w.

L - the number of the layer in which we are looking for this vertex.

$sp(i) = \sum_{j=1}^{i} c_j \; i = 1, ..N$ $sp(i)$ – the minimal value in the i-th layer

$$sk(i) = \begin{cases} \sum_{j=i}^{N} c_j \; i = 1, \cdots, N \\ 0 \qquad i = N+1 \end{cases}$$

$sk(i)$ – the maximal value in the layer $N-i+1$ or the maximal possible contribution of $(N-i+1)$ layers to the length of the path.

if $(w \geq sp(L))$ go to 2 if the smallest value in the layer L is greater than w then the vertex we are looking for is the first vertex in L, otherwise proceed to the proper search.

$pw(i) = i \, (i = 1, \cdots, L)$ determine the path to the vertex we are looking for and go to the end.

$a1 = sp(L)$
go to 8

2 i1=N-L+1
if$(w < sk(i1))$ go to 3 if $w \geq$ maximal value in the layer L then the problem has no solution.

stop

3	i=1	the counter - *it* denotes at the same time the number of the layer to which belongs the subsequent just determining vertex on the path leading to the desired vertex.
	a1=0.0	a1 – the contribution in the length of the path to the desired vertex of the subpath alredy found. After the end of the procedure it becomes a value corresponding to the desired vertex.
101	i2=i1	the maximal admissible number of the index of a vertex in the i-th layer (with respect to the layer L; see (14)
	i1=i1+1	– the same for the $i+1$-layer
	w1=w-a1-sk(i1)	a1 – the length of the path from the root to the last already determined vertex on the path to the desired vertex sk(i1) – the maximal possible length of a path from vertices of the i-th and L-th layer. w1 – the minimal length of an edge between $(i-1)$-th and i-th layer needed to ensure that the path in the L-th layer could achieve the value w.
	$nr = min\,[j\|c_j > w1]$	the choice of an edge with the index nr, that is with the length c_{nr}. It ensures that the path (with the subpath already determined) ending in the layer L exceeds the length w. From the definition of $w1$ it follows that no path (ending in the layer L) can be of the length greater than w, when the index in the i-th layer is less than nr.
	if($nr < i2$) go to 4 go to 5	if the index nr is smaller than the maximum admissible one in the i-th layer with respect to the L-th, we proceed to the further part of the algorithm.If this index is equal to that maximum one, the task is finished (go to 5) because the remaining indices of the path ending in the layer L have to be also maximal according to the layers (cf.(16)).

Index nr can not be greater than the maximum admissible one because otherwise it would mean that the problem has no solution. This possibility has been excluded earlier

4 if $(i < 2)$ go to 9
if $(nr > pw(i-1))$ go to 9
$nr = pw(i-1)+1$

according to the construction of the graph the index must be greater than the preceding one. Thus if nr is less or equal to the previous index (for $i-1$) then to the variable nr an adequate value is assigned.Checking this inequality makes sense for $i \geq 2$.

9 pw(i)=nr — in the matrix pw the indices of the subsequent vertices on the path leading to a vertex in the L-th layer with the desired property are stored.

$a1 = a1 + c_{nr}$ — actualize the value $a1$.

$i = i+1$

if(i $\leq$ L) go to 101 — if not all the layers up to the desired one has been examined, repeat the loop.

stop — otherwise the path has been found.

5 $a1 = a1 + sk(i2)$
do 102 $i3 = i, L$
$pw(i3) = i2$
102 $i2 = i2+1$
8 stop.

d. An algorithm for examining the vertices of the given layer

After finding the first vertex in the given layer to which a value of the function (1) greater than the given value has been assigned, we have to examine remaining vertices in the layer in order to find out if there is a vertex with the value greater than the given value but smaller than the value assigned to the vertex just found. Since the desired vertex may not be uniquely defined, all the equivalent vertices should also be found. The search algorithm consists in visiting some groups of vertices, descendants of a certain vertex, which will be changed during the performance of the algorithm. There is a manner in which we can estimate whether a vertex which is the prospective one from the point of view of our search criterion belongs to such a group. In this case we

examine a group of descendants of an immediate descendant of the vertex whose descendants have been examined before and which is the earliest one in its layer. If there is no such a prospective vertex in a given group, we examine the next group of vertices - descendants of a vertex defined anew with respect to the previous group. The algorithm will be presented similarly to that in c.

Input data

	w	the assumed real number. We are looking for a vertex with the smallest possible value greater than w.
	L	the number of the examined layer.
	$pw(i)$ $(i = 1, \cdots, L)$	the path to the earliest vertex in the layer L for which the value w is exceeded - the standard vertex.
	a1	the value assigned to this vertex.
	lp=1	the counter of equivalent vertices.
	$i4 < L$	the number of the greatest such index that $pw(i4)$ is less than the maximum admissible value for this index i.e. the vertex with the path $pw(1), \cdots, pw(i4)$, $pw(i4+1)+1$ does not exist or it has no descendants in the layer L, while $pw(1), pw(i4-1), pw(i4)+1$ does exist and has its descendants in the layer L (cf.(12)-(15)). If $pw(L)$ is less than the maximum admissible value we set $i4 := L-1$. In practice $i4$ is being determined in the procedure described in c., so to say parallelly to the determination of a desired vertex. For the clarity of the description we have skipped this problem there.
1	if $(i4 < 1)$ stop	if $i4 = 0$, then it means that all the vertices in the layer have been examined (it follows from the further description of the algorithm)
	i1=pw(i4)+1	the group of the descendants of the vertex $pw(1)$, $\cdots, pw(i4-1)$, $i1$ in the layer L is the group of vertices which has been examined subsequently. All the vertices earlier than the vertices of this group have been examined in the procedure described in c or during the subsequent iterations of the current procedure.

	$r = \sum_{i=1}^{i4-1} c_{pw(i)} + c_{i_1} + \sum_{i=i4+1}^{L} c_{i_1+(i-i4)}$	the smallest value in the examined group of vertices (it corresponds to the earliest of them (15))
	if $(r > w)$ go to 3	if the smallest value in the group of vertices is greater than w , this and $a1$ must be compared. If $r = a1$ then the equivalent vertex has been found and it has to be included into the list. If $r < a1$ then we have found a vertex with the value closer to w than the one with the corresponding value $a1$, so the list is initialized anew. If $r > a1$ - proceed to the examination of the next group.
	go to 4	if $r \leq w$ then find in the current group a vertex with the value greater than w.
3	if(r < a1) go to 5 if $(r = a1)$ go to 6 if $(r > a1)$ go to 7	compare r and $a1$.
5	$lp = 1$	the newly found vertex is better than the previous one. The counter of vertices set to 1.
	a1=r	change of a pattern value
	go to 8	proceed to the recording of the path
6	lp=lp+1	the equivalent vertex has been found
8	pw(i4)=i1 pw(i)=pw(i-1)+1 for $i = i4 + 1, \cdots, L$	in the matrix pw a path to the earliest descendant of the vertex defined by the path $pw(1), \cdots, pw(i4-1)$, $i1$ (layer $i4$) in the L-th layer.

	i4=i4-1	to all the vertices from the layer $i4$ which are the descendants of the vertex with the path $pw(1), \cdots, pw(i4-1)$ correspond values greater than r. So we have to examine the group of descendants of the vertex in the layer $i4-1$, which is the next to the vertex with the path $pw(1), \cdots, pw(i4-1)$, in the layer L.
	go to 9	proceed to the recording
7	i4=i4-1	see the previous comment.
	go to 1	
4	pw(i4)=i1	in the group of the descendents of the vertex from
	$r = \sum_{i=1}^{i4} c_{pw(i)}$	the layer i4 with the path $pw(1), \cdots, pw(i4-1)$, $i1$ we are looking for the first vertex whose corresponding value is greater than w. The search algorithm is similar to the procedure described in part c.
i1=N-L+i4+1		the maximum admissible value of the $(i4+1)$-th element of the path which ends at the layer L with the element of the value N.
	i=i4+1	
101	i2=i1	we store the maximum admissible value of the
	i1=i1+1	index in the layers i and $i+1$.
w1=w-r-sk(i1)		the minimal length of an edge between $(i-1)$-th and i-th layer needed to ensure that the path in the L-th layer could achieve the value w.
$nr = min\,[j \mid c_j >$ w1]		the smallest value of the index ensuring that $w1$ is exceeded.
	if ($nr < i2$) go to 14	if the index is equal to the maximum admissible
	go to 15	then the remaining indices should attain the given values according to (16).
14	if (nr > pw(i-1))	the value of the subsequent index must be
	go to 19	greater than that of the previous one

nr=pw(i-1)+1

19 r=r+c_{nr} the subsequent vertex of the path to the vertex with the demanded property has been found. Modify the length of the subpath.

$pw(i) = nr$ store the index of the vertex found.

i=i+1

if (i ≤ L) go to 101 if still some of the L layers have not been used, proceed to the next layer.

i4=L-1 actualize the value of the parameter $i4$.

go to 16 the vertex is fixed, go to the comparison with the patern value.

15 i4=i-1 actualize the value of the parameter $i4$.

pw(i)=i2 the subsequent vertices of the path to the ver-

pw(j)=pw(j-1)+1; tex with the demanded property (in the case

$j = i + 1, \cdots, L$ when this is the latest descendant of the ver-

r=r+sk(i2) tex $pw(1), \cdots, pw(i-1)$), and the length of this path are stored.

16 if (r > a1) go to 1 if the value corresponding to the found vertex is

if (r=a1) go to 21 greater than the pattern value , we skip it, proceeding to the next branch. If this value is equal to the pattern, we include the vertex into the list as a vertex equivalent to the pattern vertex. If this value is less than the patern we abandon the list of the equivalent vertices and the newly found vertex becomes the pattern vertex.

lp =1 the counter of equivalent vertices is set to 1.

a1=r the new pattern value is fixed.

go to 9 proceed to the recording of the new pattern vertex.

21 lp=lp+1 if the vertex we found is equivalent to the pattern, increase the counter of equivalent vertices by 1 and add it to the list.

9 record in the suitable set the path to the found vertex on the position determined by lp.

go to 1 proceed to the search of subsequent branches.

e. An algorithm to determine vertices equivalent to the fixed one in the case of

equal coefficients.

If in the sequence of indices determining the vertex we change one of the indices and at the same time the coefficients corresponding to the old index and to the new one are equal then the same value of the function (1) corresponds to the vertex determined by the new sequence and to the initial one. If we substitute a new index , we have to take into account the principles of the construction of the graph given in 2a., especially the second condition. The application of this fact allows us to skip certain groups of vertices while searching in the algorithm presented in d. The algorithm described below enables us to determine equivalent vertices by the suitable change of indices.

$pw(i)$, $i = 1, \cdots, L$ the defined sequence of indices describing a vertex in the layer L. For this vertex we have to determine equivalent vertices taking into account the equality of coefficients.

pw1(i)=pw(i) store pw.

$$pw2(i) = \begin{cases} N+1 \quad \text{if} c_{pw(i)} = c_{pw(i)+1} = \cdots = C_N \\ min[j|\, c_j > c_{pw(i)}] \end{cases}$$

$$i = 1, \cdots, L$$

the ranges of the variation of the respective indices are stored. They ensure that the respective coefficients c_j are constant.

```
   i=L
1  if (pw(L)+1 < pw2(L)) go to 4
   go to 3
```

the loop beginning with the control variable i equal to L is beeing performed. For every change of an index on any position, all the permitted changes of indices on the subsequent position are performed. i is the pointer to the position of the vector pw, where the equivalent indices are substituted. If an index on the i-th position has attained its upper limit, we proceed to the previous position.

```
4   pw(L)=pw(L)+1        pw defines a path to the vertex equivalent to
    store pw             the initial one. We store (or we use it in an-
    go to 1              other way) this path and we proceed to the
                         further execution of the loop for the control
                         variable equal to L.

3   i=i-1                we proceed to the previous position.
    if (i< 1) go to 7    if i = 0 then the set of all the positions have
                         been exhausted.

    if (pw(i)+1 < pw2(i) ) go to 6   we check whether the loop on the i-th
    go to 3                          position is exhausted. If it is the case,
6   pw(i)=pw(i)+1                    proceed to the earlier position, if not -
                                     proceed to the substitution on the later
                                     positions.
      do 101 i5=i+1,L                on the positions i + 1 to L the indices
      i2=pw1(i5)                     from the initial (pattern) path are sub-
      if(i2 > pw(i5-1)) goto 9       stituted, they are stored in pw1. The
      i2=pw(i5-1)+1                  condition 2. of the construction of
      if(i2 < pw2(i5)) goto 9        the graph is checked. We check also
      go to 3                        whether the admissible value of an in-
9     pw(i5)=i2                      dex on a given position (pw2(i5))is ex-
101   continue                       ceeded. If it is unable to fulfil the re-
                                     spective conditions, we proceed to the
                                     position i − 1.

        i=L
        store pw
        go to 1                      proceed to the execution of the loop
                                     for the control variable equal to L.
7     stop.
```

f. An algorithm to solve the problem1.

Using the algorithm described above, for a given value w, it is possible to find a sequence of solutions such that its corresponding sequence of values of function (1) is nondecreasing. The algorithm has the following form:

1. define w. If w is greater than or equal to the maximum admissible value then the problem has no solutions. Otherwise go to 2.

2. define layers in which the search will be performed (cf. part b.)

3. define as a pattern value w_w maximal value of the function (1).

4. To each of the layers defined in 2 do 4.1 - 4.2.

   4.1 find the earliest vertex in the layer, whose corresponding value is greater than w, denote it $\bar{w}_w$ (the algorithm from part c.).

   If $\bar{w}_w < w_w$ modify the pattern vertex and the pattern value and skip the list of the equivalent vertices.

   If $\bar{w}_w = w_w$ add the vertex to the list of the equivalent vertices.

   If $\bar{w}_w > w_w$ go to 4.2.

   4.2 visit the remaining vertices in the layer according to the algorithm described in part d. During this algorithm the pattern vertex and the pattern value w_w are modified.

5. for every vertex from the list of the found equivalent vertices with the same value w_w , find all equivalent vertices taking into account the equality of the coefficients.

6. set $w = w_w$, if subsequent elements of the sequence are needed ,go to 1.

II. Any bounded variables.

3 Problem statement

Define the function:

$$\begin{aligned} f_1(\mathbf{x}) &= \textstyle\sum_{i=1}^{N} x_i \\ \mathbf{x} &= [x_1, \cdots, x_N]^{\mathsf{T}} \\ x_i &\in \{c_1^i, \cdots, c_{N_i}^i\} \qquad i = 1, \cdots, N \\ c_j^i \in R; \quad i &= 1, \cdots, N; \qquad j = 1, \cdots, N_i \end{aligned} \tag{17}$$

Assume, without loss of generality, that c_j^i are ordered for the corresponding variables and distinct.

$$c_j^i < c_{j+1}^i \qquad i = 1, \cdots, N; \qquad j = 1, \cdots, N_i - 1 \tag{18}$$

The case when the values are equal needs a simple trick: we "stick the branches together" . It will be described separately.

For such a function an algorithm solving the problem 1 from the part 1. will be described.

4 An algorithm for the numerical solution of the problem with any variables

a. A model.

Consider a graph with the tree structure (Fig. 2) constructed recursively as follows:

1. Choose the root and assign to it a pair of indices $(0, 0)$.

2. A vertex with (i, j) is a parent of the vertices with the indices $(i + 1, k)$ for $k = 1, \cdots, N_{i+1}$. The vertex with the first index equal to N has no children.

3. If the indices of a vertex which is a child are equal to (i, j) then the weight of the only edge between this child and its parent node is equal to c_j^i.

The vertices of the graph can be devided into layers as in the previous case. However the interpretation of this division is different. Namely from the point of view of the current problem which is the general one, all the vertices of the previously constructed graph would belong to the layer N.

The cardinality of this layer is equal to $N_1 * N_2 * \cdots * N_N$. To every vertex a unique sequence of second indices of the vertices on the unique path from the root to this vertex can be assigned. We can apply (4)-(5) and assign to every vertex the value $f_1(\mathbf{x}^{k_L}) = \sum_{i=1}^{L} c^i_{j_i^{k_L}}$. The vertices in each layer can be ordered using (7)-(8). According to this order, the first vertex in the layer L is the vertex with the sequence of indices:

$$\underbrace{1, 1, \cdots, 1}_{L \text{ times}} \tag{19}$$

and the last one:

$$N_1, \cdots, N_L \tag{20}$$

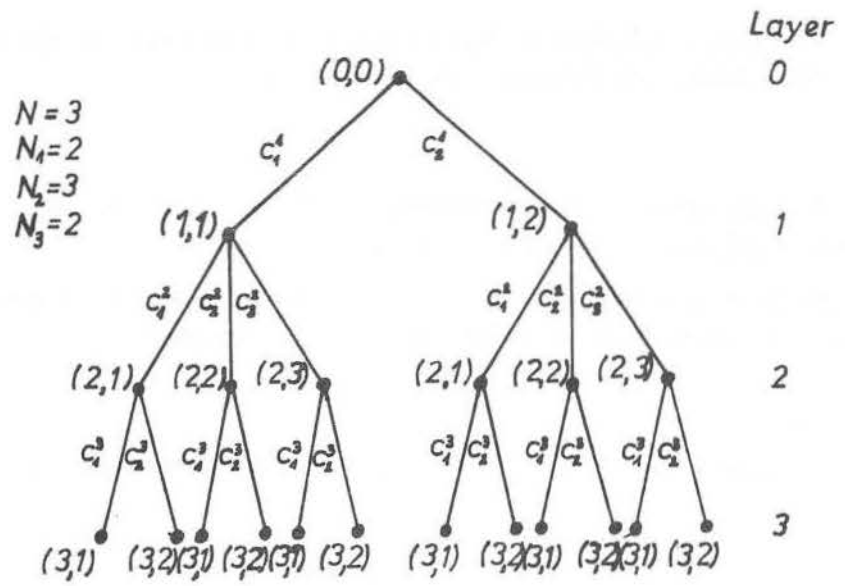

Figure 2

To the first vertex corresponds the smallest value of the function f_1 in the layer L and to the second vertex - the greatest one.

Let us take any vertex from the layer L_1 - this layer is earlier than the layer L.

Let the following sequence of indices corresponds to this vertex:

$$j_1, \cdots, j_{L_1} \tag{21}$$

The earliest descendant of this vertex in L is:

$$j_1, \cdots, j_{L_1}, \underbrace{1, \cdots, 1}_{L - L_1 \text{ times}} \tag{22}$$

and the last one;

$$j_1, \cdots, j_{L_1}, N_{L_1+1}, \cdots, N_L \tag{23}$$

From the construction of the graph and from the ordering of the variables (18) it follows that to the vertex (22) corresponds the smallest value of the function f_1 and to the vertex (23) the greatest one among the values assigned to the descendants of (21) in the layer L.

The relations described in this part allowed us to construct the algorithm for determining succesive values of the function (17).

b. An algorithm for finding the first vertex in the layer N with a value of the function (17) exceeding the given value

For fixed w we have to find in the layer N the first vertex, with respect to the defined order, to which a value greater than w is assigned.

$$sk(i) = \begin{cases} 0 & i = N \\ \sum_{j=i+1}^{N} c_{N_j}^{j} & i = 1, \cdots, N-1 \end{cases}$$ the given auxilary values.

$$sp(i) = \begin{cases} 0 & i = N \\ \sum_{j=i+1}^{N} c_{1}^{j} & i = 1, \cdots, N-1. \end{cases}$$

	ips=0	the control variable of the loop - the number of the layer of the currently appointed vertex (its first index)
	isw=0	the length of the subpath from the root to the vertex preceding the vertex found.
1	ips=ips+1	beginning of the loop; we determine *ips*-th vertex on the path to the desired vertex.
	ia=w-isw-sk(ips)	isw — the length of the subpath found. sk(ips) — the maximum possible length of the subpath from the layer *ips* to the layer N. ia — the minimum value which must be exceeded by the weight of an edge to the vertex we are looking for. It guarantees that the path which ends in the layer N , attains the length greater than w.
	$i = min\left[j \mid c_j^{ips} > ia\right]$	the minimum index of an edge joining the layers (*ips* − 1)-th and *ips*-th with the weight greater than *ia* (cf. the comment concerning *ia*). The desired vertex in the layer *ips* has indices (ips, i)

$isw = isw + c_i^{ips}$	the modified length of the subpath found.
pw(ips)=i	we store the index of the succesive vertex of the path we are looking for.
if(ips < N) go to 1	if the layer N has not yet been reached, proceed to the search for the next vertex.
stop	$pw(\cdot)$ maintains a sequence of indices which identifies the vertex with the desired property.

c. An algorithm for searching the layer N

After finding the first vertex with the value greater than w, we have to perform a survey of the remaining vertices in order to find the vertices equivalent to this vertex or better than this one, i.e. with the smaller value. Searching for the first vertex is useful because it excludes from the survey a certain set of vertices and is not time-consuming.

The search algorithm is analogous to the case of $0-1$ variables with regard to the differences in the construction of the adequate graphs.

	pw(i)	$i = 1, \cdots, N$ the path to the pattern vertex.
	isw > w	the value corresponding to this vertex.
	sp(i), sk(i)	see part b.
	lp=1	the counter of vertices equivalent to the pattern vertex (pattern vertex included).
11	l=N	the movable pointer pointing to the layer of which the vertex, whose descendants in the layer N are being examined in the actual loop, is a member.
3	l=l-1	after arriving at this place for the first time we get a vertex of the layer N and its assigned value. We have to answer the following question: is there, among the descendants of some vertices of the layer $N-1$, a vertex better than the pattern. Afterwards we have to proceed to the examination of the descendants of some vertices from the previous layer. The explanation of the details we can get by the analysis of the steps from which the procedure jumps to the label 3.

if(l < 1) stop — all the vertices in the layer N have been examined.

i=pw(l)+1 — the index of the first neighbour, next to the vertex identified by the path $pw(1), \cdots, pw(l)$, in the current layer.

if($i > N_l$) go to 3 — if there is no such a neighbour, proceed to the previous layer.

pw(l)=i

$isw1 = \sum_{j=1}^{l-1} c^j_{pw(j)} + c^l_i$ the value assigned to such a vertex.

i1=isw1+sp(l) — the value corresponding to the earliest descendant in the layer N of the aforementioned vertex.

if (i1 > w) go to 6 — if this value exceeds w, compare it with isw. If it is not greater than w find the first descendant of the vertex determined above of the layer l in the layer N to which corresponds the value greater than w. Such a vertex must exist, it exists among the descendants of the vertex predecessing the aforementioned vertex in the layer l (the pattern vertex). In the next part we look for such a vertex.

ips=l

7 ips=ips+1 — similar to the part described in b. After its performance $pw(\,)$ maintains indices of the path to the desired vertex, $isw1$-its corresponding value.

ia=w-isw1-sk(ips)

$i = min\left[j | c^{ips}_j > ia\right]$

$isw1 = isw1 + c^{ips}_i$

pw(ips)=i

if (ips < N) go to 7

if (isw1 < isw) go to 9 — the vertex better than the pattern one has been found. Its value is greater than w but less than that of the pattern vertex. Modify the pattern vertex.

	if (isw1=isw) go to 10	a vertex equivalent to the pattern vertex has been found, store it, modify the value of l.
	if (isw1 > isw) go to 11	the vertex found is worse than the pattern. Its value is greater than w and also than isw. Examine later vertices.
6	if (i1 > isw) go to 3	if the earliest descendant, this one with the smallest value, is worse than the pattern one then there is no better vertex among the descendants of the next neighbour of the described vertex in the layer l. Thus we will visit the descendants of the suitable vertex in the layer $l-1$.
	pw(j)=1 $j = l+1, \cdots, N$	fix a path to the aforementioned earliest descendant of the current vertex. $i1$ - its corresponding value.
	if(i1 < isw) go to 18	if the found vertex is better than the pattern, modify the pattern.
	if(i1=isw) go to 15	if it is equivalent, store it (label 15).
18	lp=0	
	isw=isw1	
	go to 15	
9	lp=0	
	isw=isw1	
10	l=N	
15	lp=lp+1	
	store $pw(i)$ $i = 1, \cdots, N$.	
	go to 3 proceed to the further survey.	

d. The case when the admissible values of the elements of individual cataloques are equal

We assumed until now that every variable can take different discrete values (18). It makes sense, from the practical point of view, to consider also the case of multiple values. Namely, in the case of structure optimization , if the variable represents the cross-section area of a bar, it may happen that to the same cross-section area correspond different parameters e.g. moment of inertia.

Assume that:

$$c_j^i = c_{j,1}^i = \cdots = c_{j,k_j^i}^i \tag{24}$$

Consider the function (17). Consider a given vertex in the layer N with the path:

$$j_1, \cdots, j_N \tag{25}$$

Let its corresponding value be w. From (24) follows that the same value corresponds to all the vertices with the following paths defined by the loops:

```
      do 100 i1=0, k^1_{j_1}
      do 100 i2=0, k^2_{j_2}
      ⋮
      do 100 iN=0, k^N_{j_N}
      j_1 = j_1 + i1
      ⋮
      j_N = j_N + iN
100   continue.
```

e. An algorithm to perform the problem 1 in the case of discrete variables

Using the procedures described above, we can construct the following algorithm building the sequence of solutions whose corresponding values of the function (17) is nondecreasing:

1. define w. If w is greater than or equal to the maximum value in the layer N, the problem has no solution. Otherwise go to 2.
2. find the earliest of the vertices (the pattern) in the layer N with the value of the function (17) greater than w. Let us denote it by w_w - the algorithm from the part b.
3. visit other vertices in the layer N - the algorithm from the part c. During this algorithm the pattern vertex and the value w_w are modified.
4. for every vertex from the list of the found equivalent vertices with the same value w_w, find equivalent vertices taking into account the equalitis of certain values from the discrete catalogue.
5. $w = w_w$; if the subsequent elements are needed, go to 1.

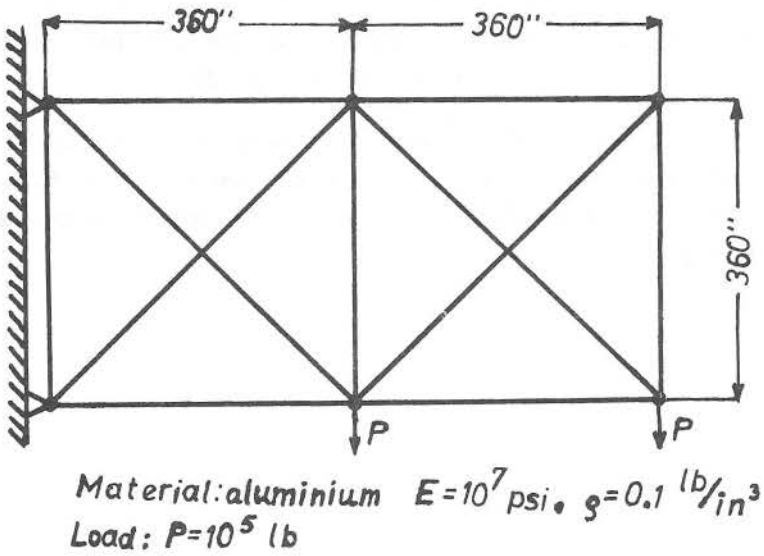

Figure 3

Numerical example

The example problem is a ten-member planar truss often presented by the authors discussing elastic structure optimization (Fig. 3). We consider the following case of this structure. Each of ten structural memebers can be selected separately. The structure is subjected to the displacement constraint $u \leq 2in$ imposed on the vertical displacements of all nodes and stresses in all rods $\sigma \leq 2.5 * 10^4\ lb/in^2$. The catalogue of available sections contains ten following section areas $[in^2]$ [0.1, 0.5, 1.0, 2.0, 4.0, 7.0, 12.0, 19.0, 27.0, 36.0].

The search was performed on PC XT. It has started with the value w equal to the continuous solution $w = 5026.39[lb]$. The found discrete solution was equal to $5356.11[lb]$. The full survey of the set of solutions (10^{10} cases) would be impossible to perform even on large computers.

References

[1] GREENBERG H. Integer Programming. Academic Press, New York, London, 1971.

[2] GUTKOWSKI W., BAUER J., IWANOW Z. Minimum Weight Design of Space Frames From a Catalogue; Shells, Membranes and Space Frames; *Proceedings IASS Symposium, Osaka*, **3** (1986), p. 229.

[3] IWANOW Z. The Method of Enumeration According to the Increasing Value of the Objective Function in the Optimization of Bar Structures, *Bulletin de L'Academie Polonaise des Sciences Serie des sciences techniques*, **29**, No. 9-10, (1981).