STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING MATHEMATISCHE STATISTIEK

S 398

AN OPTIMAL PARTITIONING

by

Jac.M. Anthonisse.

July 1968.

Introduction

An example of the type of problems to be discussed here is found in connection with the construction of time-tables [1].
Suppose that n lectures (of one hour) must be given within m hours. There is , in general, a large number of ways to assign an hour to each lecture. An assignment is infeasible if the same hour in assigned to different lectures to be given by the same person. An assignment is undesirable if the same hour is assigned to different lectures that are of interest to a common set of students. An assignment is good if there is only a small number of cases in which a student must make a choice between lectures, that he would like to attend, because they are given simultaneously.

This leads to the following problem (in which a_{ij} may be interpreted as the number of students liking to attend both lectures i and j):

To each pair of elements (i,j) from a set
$X=\{1,2,\ldots,n\}$ a number a_{ij} with $a_{ij} \geq 0$,
$a_{ij} = a_{ji}$ and $a_{ii} = 0$ has been assigned.
Find a prescribed number of disjoint, non-empty subsets $P_1,\ldots,P_m$ from X such that

$$X = \bigcup_{h=1}^{m} P_h$$

and

$$z = \sum_{h=1}^{m} z_h \qquad \text{is minimized,}$$

where

$$z_h = \sum_{i<j} (a_{ij} \mid i \in P_h, \ j \in P_h).$$

To solve this problem a 'double' branch and bound algorithm was constructed. In the first stage of the algorithm a 'representation' is generated, that is an 'incomplete partitioning' in which each set P_h contains one element only. A lower bound for the z-values of all 'complete partitionings' that are extensions of the representation is computed. If this lower bound exceeds the smallest upper bound for the minimal value of z known at that moment a new representation is generated.

In the second stage of the algorithm a lower bound for the z-values of all complete partitionings that are extensions of a given incomplete partitioning is computed. The value of this lower bound is used to determine whether the incomplete partitioning must be extended or not. If not, the algorithm returns to an incomplete partitioning of which the current incomplete partitioning in an extension and generates a new extension.
After all extensions of a representation have been enumerated (explicitly or implicitly) stage 1 is activated again.

The order in which the representations are generated and the lower bound that is used to detect whether a respresentation may be skipped or must be extended leads, by heuristic reasoning, to a re-indexing (permutation) of the rows and columns of (a_{ij}) which may be applied before the proper algorithm is performed.

A problem related to the one defined above is that of finding the minimal value of m for which a partitioning with z=0 exists. The algorithm given in [2] is similar to the algorithm presented in this report.

Numerical Results

Matrices (a_{ij}) were generated with the help of the following distribution:

$$\text{Prob}\ (\underset{\sim}{x} = 0) = \rho$$

$$\text{Prob}\ (\underset{\sim}{x} = i) = \frac{1-\rho}{n^2} \qquad (i=1, \ldots, n^2).$$

For each of the combinations n=10, ρ= .2,.4,.6,.8 20 matrices were generated, for each matrix the partitioning problems with m=2,4,6,8 were solved.

For each of the combinations n=15, ρ = .2,.4,.6,.8 10 matrices were generated, for each matrix the partitioning problems with m=3,6,9,12 were solved.

In all cases the Algol-procedures given in the last section of this report were used.

Table 1 contains the means of the observed times that were used to rearrange the matrices.

Tables 2 and 3 contain the means of the observed times that were used to solve the partitioning problems for n=10 and n=15 respectively. All times are given in seconds.

The computations were performed by the ELECTROLOGICA X-8 computer.

ρ / n	.2	.4	.6	.8
10	.53	.43	.40	.31
15	1.44	1.18	.94	.81

table 1: mean time to rearrange the matrix.

ρ \ m	.2	.4	.6	.8
2	2.00	1.37	.73	.21
4	1.00	.34	.13	.12
6	.18	.08	.08	.08
8	.05	.05	.05	.05

table 2: mean time to solve a problem with n=10.

ρ \ m	.2	.4	.6	.8
3	171.70	39.93	2.56	.49
6	8.50	.76	.25	.25
9	.26	.15	.16	.17
12	.08	.08	.08	.09

table 3: mean time to solve a problem with n=15.

Definitions

For each non-empty subset P_h from $X=\{1,\dots,n\}$ the 'representative' P_h is defined as

$$P_h = \min_i (i \mid i \in P_h)$$

The non-empty and disjoint subsets $P_1,\dots, P_m$ from X constitute a 'complete partitioning' if

$$X = \bigcup_{h=1}^{m} P_h .$$

The non-empty and disjoint subsets $P_1,\dots, P_m$ from X constitute an 'incomplete' partitioning' if

$$X \supset \bigcup_{h=1}^{m} P_h.$$

If $P_1,\dots, P_m$ is an incomplete partitioning then $p_1,\dots, p_m$ is said to be the 'representation' of the incomplete partitioning, where p_h is the representative of P_h $(h=1,\dots,m)$.

It will be assumed in the sequel that the subsets constituting an incomplete partitioning are indexed in such a way that

$$p_1< p_2< p_3< \dots <p_m.$$

The incomplete partitioning $P_1,\dots,P_m$ is said to be an 'extension' of the incomplete partitioning $Q_1,\dots,Q_m$ if

$$Q_h \subset P_h \quad (h=1,2,\dots,m).$$

As a representation $p_1,\dots,p_m$ may be identified with the incomplete partitioning $\{p_1\},\dots,\{p_m\}$ it may be stated that each incomplete partitioning is an extension of its representation.

It is easily seen that the relation 'having the same representation' is an equivalence relation on the class of all incomplete partitionings. Each equivalence-class is identified by a representation and contains all incomplete partitionings that are extensions of that representation.

Obviously, an equivalence class contains a complete partitioning if and only if $p_1=1$.

Stage 1.

As X must be partitioned into disjoint non-empty subsets $P_1,\ldots,P_m$ with

$$X = \bigcup_{h=1}^{m} P_h$$

and

$$p_1 < p_2 < \ldots < p_m$$

it follows that

$$h \leq p_h \leq n-m+h$$

and

$$p_1 = 1$$

The representations having these properties are generated by application of the following rules:

'first' representation:

$p_h = h \qquad (h=1,2\ldots,m)$

'next' representation:

let $\quad p_1,\ldots,p_m$ be the present representation.

step

1 $\quad$ compute $h^* = \max_h (h \mid p_h < n-m+h)$,

2 $\quad$ if $h^*=1$ all (relevant) representations have been generated, terminate the algorithm,

3 $\quad$ the new representation is defined by:

$$p^1_h = \begin{cases} p_h & \text{for } h=1,2,\ldots,h^*-1 \\ p_h+1 & \text{for } h=h^* \\ p^1_{h^*}+h-h^* & \text{for } h=h^*+1,\ldots,m. \end{cases}$$

Let $P_1,\ldots,P_m$ be a complete partitioning, with representation $p_1,\ldots,p_m$.

As $p_1<p_2<\ldots<p_m$ it follows

from the definition of a representative that

$$j \notin \bigcup_{h=1}^{l} P_h$$

if

$$p_l< j< p_{l+1} \quad (l=1,\ldots,m-1).$$

Define

$$g_l = \sum_{p_l<j<p_{l+1}} \min (a_{p_h j} \mid h=1,\ldots,l) \text{ for } l=1,2,\ldots,m-1,$$

and

$$g_m = \sum_{p_m<j} \min (a_{p_h j} \mid h=1,\ldots,m).$$

Then, for each $h=0,1,2,\ldots,m$

$$G_h = \sum_{l=1}^{h} g_l$$

is a lower bound for the z-values of all complete partitionings that are extensions of the representation $p_1,\ldots,p_m$.

Now suppose that a complete partitioning with $z = z^*$ was found already.

If $G_m > z^*$ extension of the present representation will not lead to a better complete partitioning, thus a next representation may be generated.

The rules given above generate all representations, and a sequence of representations with $G_m > z^*$ may be found. The following rules, applied to each representation found in step 3, will skip representations for which it is evident that no extension to a complete partitioning with $z < z^*$ exists.

let $p_1,\dots,p_m$ be the representation found in step 3

```
step
4          $k = \max_h (h | G_h < z^*,\ h=0,1,\dots,m)$,

5          if $k \neq m$     define $h^* = k+1$, return to step 2,

           if $k = m$     the present representation may not be skipped.
```

The contents of steps 4,5 may be clarified with the following remarks. Suppose that the representation (1,4,7,9) has been generated in a problem with n=10 m=4. If steps 4,5 were not applied the next representations would be (1,4,7,10) etc.

	1	2	3	4	5	6	7	8	9	10
$\longrightarrow$	x			x			x		x	
	x			x			x			x
	x			x				x	x	
	x			x				x		x
	x			x					x	x
$\longrightarrow$	x				x	x	x			
	x				x	x		x		
	x				x	x			x	
	x				x	x				x
	x				x		x	x		

If, for representation (1,4,7,9), $G_1 < z^*$ but $G_2 > z^*$, this will also be the case for the next 4 representations because g_2 does not decrease. Thus the 5 representations may be skipped. The next representation to be considered is (1,5,6,7).

<u>Stage 2.</u>

Let $P_1, \ldots, P_m$ be an incomplete partitioning (e.g. a representation). Define

$$F = X - \bigcup_{h=1}^{m} P_h$$

and

$$b_{hj} = \begin{cases} \sum_i (a_{ij} \mid i \in P_h) \text{ if } j > p_h \\ + \infty \quad \text{otherwise,} \end{cases}$$

for $h = 1, 2, \ldots, m$ and each $j \in F$.

Define

$$z_h = \sum_{i<j} (a_{ij} \mid i \in P_h,\ j \in P_h) \quad (h = 1, \ldots, m),$$

and

$$c_j = \min_h b_{hj} \quad \text{for each } j \in F.$$

Then

$$\bar{z} = \sum_{h=1}^{m} z_h + \sum_{j \in F} c_j$$

is a lower bound for the z-values of all complete partitionings that are extensions of the present incomplete partitioning.*)

*) The value of $\bar{z}$ might be increased by adding a term depending on the a_{ij} with $i \in F$ and $j \in F$.

If $\bar{z} > z^*$ the present incomplete partitioning may be abandoned, otherwise a $j^* \in F$ is selected and the incomplete partitionings

$$P_1, \ldots, P_h \cup \{j^*\}, \ldots, P_m \quad (h=1,\ldots,m)$$

are generated, as will be specified below.
Now stage 2 of the algorithm may be described with the help of the recursive procedure

$$T(P_1, \ldots, P_m)$$

which consists of the following steps:

```
step
1       determine F
        if $F \neq \emptyset$ proceded with step 3,

2       copy the current complete partitioning, replace the current
        value of $z^*$ by the z-value of the complete partitioning, if
        $z^* = 0$ terminate the algorithm,
        if $z^* > 0$ terminate this instance of T,

3       determine $b_{hj}$, $c_j$ and $\bar{z}$
        if $\bar{z} > z^*$ terminate this instance of T,

4       determine $j^*$ by: $c_{j^*} > c_j \quad j \in F$,

5       determine $h_1, \ldots, h_m$ such that
        $b_{h_1 j^*} \leq b_{h_2 j^*} \leq \ldots \leq b_{h_m j^*}$ ,
        perform
        $T(P_1, \ldots, P_{h_l} \cup \{j^*\}, \ldots, P_m)$

        for $l=1,2,\ldots$ with $\bar{z} - c_{j^*} + b_{h_l j^*} < z^*$.
```

(Notice that the value of z^* may be decreased by a call of T.)

Stage 2 of the algorithm is equivalent with the call

$$T\left(\{p_1\}, \ldots, \{p_m\}\right) \quad ,$$

where $p_1, \ldots, p_m$ is a representation generated by stage 1.

It should be noted that the matrix (b^1_{hj}) associated with the incomplete partitioning

$$P_1, \ldots, P_{h^*} \cup \{j^*\}, \ldots, P_m$$

may be obtained from the matrix (b_{hj}) associated with the incomplete partitioning

$$P_1, \ldots, P_{h^*}, \ldots, P_m$$

by

1 deleting the column corresponding with j^* ,

2 re-computing the remaining elements of the h^*-th row:

$$b^1_{h^*j} = b_{h^*j} + a_{jj^*} \quad (j \in F \setminus \{j^*\}).$$

<u>Stage 0.</u>

The algorithm sketched above may be applied to any symmetric matrix (a_{ij}) with $a_{ij}>0$ and $a_{ii}=0$.

The problem defined in the first section remains the same if the rows and columns of (a_{ij}) are permuted. This freedom to re-arrange the elements of (a_{ij}) may be used to accelerate the algorithm.

Let i_1 be an index i for which

$$\sum_{j=1}^{n} a_{ij}$$

is maximal.

Select i_k $(k=2,\ldots,n)$ as an index i

with

$$i \notin \{i_1,\ldots,i_{k-1}\}$$

for which

$$\min (a_{ii_l} \mid l=1,\ldots,k-1)$$

is maximal, if this maximum is attained for several indices an index i for which

$$\sum_{j=1}^{n} a_{ij}$$

is maximal may be chosen from that set of indices.
If the rows and columns of (a_{ij}) are permuted according to the permutation

$$(i_1,\ldots, i_m)$$

it may be expected that, in general, the optimal solution of the partitioning problem will be found in a small number of iterations. As z^* then reaches its minimal value only a relatively small number of representations will be presented to stage 2.

The permutation $(i_1,\ldots, i_m)$ obtained in this way is optimal in the sense that for fixed $i_1,\ldots,i_{k-1}$, selection of $i^*_k \neq i_k$, as k-th index will lend to a smaller lower bound G_{k-1} in stage 1, for all representations containing $i_1,\ldots, i_{k-1}$.

Algol procedures.

The algorithms sketched above will now be given in the form of Algol-60 procedures. All details of the algorithms may be found in these procedures.

The procedure PERMUTE contains a call of the procedure SUM, which is equivalent with

```
real procedure SUM (i,a,b,x); value b; integer i,a,b; real x;
begin real s; s: = 0;
      for i: = a step 1 until b do  s: = s + x;
      SUM: = s
end;
```

```
procedure PERMUTE(A, I, n); value n; integer n; integer array A, I;
comment PERMUTE re-arranges the elements of A[1 : n, 1 : n] in
preparation of the procedure PARTITION. I[j]:= original index of
the j-th row and column in the new array;

begin integer i, j, k, a, b;
      integer array B[1:n];

      procedure change(i, j); value i, j; integer i, j;
      begin integer a, k;
            a:= I[i]; I[i]:= I[j]; I[j]:= a;
            for k:= 1 step 1 until n do
            begin a:= A[i,k]; A[i,k]:= A[j,k]; A[j,k]:= a end;
            for k:= 1 step 1 until n do
            begin a:= A[k,i]; A[k,i]:= A[k,j]; A[k,j]:= a end;
            a:= B[i]; B[i]:= B[j]; B[j]:= a
      end change;

      for i:= 1 step 1 until n do
      begin B[i]:= SUM(j, 1, n, A[i,j]); I[i]:= i end;
      for i:= 1 step 1 until n do
      begin a:= B[i]; k:= i;
            for j:= i + 1 step 1 until n do
            begin b:= B[j]; if b > a then
                  begin a:= b; k:= j end
            end;
            if k ≠ i then change(i, k)
      end;
      b:= - 1;
      for i:= 2 step 1 until n do
      begin a:= B[i]:= A[i,1]; if a > b then
            begin b:= a; k:= i end
      end;
      for i:= 2 step 1 until n do
      begin for j:= k - 1 step - 1 until i do change(j, j + 1);
```

```
          b:= - 1;
          for j:= i + 1 step 1 until n do
          begin a:= A[i,j]; if a < B[j] then B[j]:= a else a:= B[j];
                if a > b then
                begin b:= a; k:= j end
          end
      end
  end PERMUTE;
```

```
procedure PARTITION(A, P, n, m, zstar, output); value n, m;
integer zstar, n, m; integer array A, P; procedure output;
comment PARTITION solves the following problem: the set (1,
2,...,n) must be partitioned into m disjoint non-empty subsets,
such that the cost of the partitioning is minimized. The cost of
the partitioning is defined as the sum of all A[i, j] with i < j
for which i and j are in the same subset. It is assumed that: A[i,
j] = A[j, i] ≥ 0, A[i, i] = 0 for 1 ≤ i, j ≤ n, 1 < m < n, zstar >
the minimal cost of a partitioning. The procedure generates a
sequence of partitionings with decreasing cost. Each time an
improved solution is found zstar:= cost of the partitioning, P[i]:=
j if i is in subset j, for i = 1,...,n, a call of the procedure
output is made;

begin integer z, max;
      integer array F[1:n], C, H[1:n - m], B[1:m,1:n - m], p[1:m];

      procedure nextrep;
      begin integer h, i, j, k, a, b, g;
      s1:   for h:= m step - 1 until 1 do if p[h] < n - m + h
            then goto s2;
      s2:   if h = 1 then goto exit;
      s3:   i:= p[h]:= p[h] + 1;
            for j:= h + 1 step 1 until m do p[j]:= i + j - h;
            for j:= 1 step 1 until n do F[j]:= 0;
            for j:= 1 step 1 until m do F[p[j]]:= j; k:= 0; g:= 0;
            for j:= 1 step 1 until n do if F[j] = 0 then
            begin k:= k + 1; a:= max;
                  for h:= 1 step 1 until m do
                  begin i:= p[h];
                        b:= B[h,k]:= if i < j then A[i,j] else max;
                        if b < a then a:= b
                  end;
                  H[k]:= j; C[k]:= a; g:= g + a;
                  if g < zstar then goto nextj;
            for h:= m step - 1 until 1 do if p[h] < j then
            goto s2;
      nextj:
      end;
      z:= 0;
end nextrep;


procedure iter;
begin integer h, i, j, k, a, b, zl;
s1:   b:= - 1; k:= 0; zl:= 0;
      for j:= 1 step 1 until n - m do if F[H[j]] = 0 then
      begin a:= C[j]; if a > b then
            begin b:= a; k:= j end;
            zl:= zl + a
      end;
      if k ≠ 0 then goto s3;
s2:   for j:= 1 step 1 until n do P[j]:= F[j]; zstar:= z;
      output; if z = 0 then goto exit;
s3:   if z + zl ≥ zstar then goto term; zl:= zl - b;
s4:   a:= max;
      for i:= 1 step 1 until m do
      begin b:= B[i,k]; if b < a then
            begin a:= b; h:= i end
      end;
      if z + zl + a ≥ zstar then
      begin for i:= 1 step 1 until m do
            begin b:= B[i,k]; if b ≥ max then B[i,k]:= b - max
            end;
            goto term
      end;
s5:   F[H[k]]:= h; z:= z + B[h,k]; B[h,k]:= B[h,k] + max;
      for j:= 1 step 1 until n - m do if F[H[j]] = 0 then
      begin a:= C[j]; b:= B[h,j]; B[h,j]:= b + A[H[k],H[j]];
            if a < b then goto nextj; a:= max;
            for i:= 1 step 1 until m do
            begin b:= B[i,j]; if b < a then a:= b end;
```

```
                C[j]:= a;
            nextj:
            end;
            iter;
            for j:= 1 step 1 until n - m do if F[H[j]] = 0 then
            begin a:= C[j]; b:= B[h,j]:= B[h,j] - A[H[k],H[j]];
                if b < a then C[j]:= b
            end;
            z:= z - B[h,k] + max; F[H[k]]:= 0; goto s4;
        term:
        end iter;

init: max:= ₁₀6;
        for z:= 1 step 1 until m do p[z]:= z; p[m]:= m - 1;
stage1: nextrep;
stage2: iter; goto stage1;
exit:
end PARTITION;
```

Literature .

1. K. Kirchgässner
Die graphentheoretische Lösung eines
nicht-linearen Zuteilungsproblems.
Unternehmensforschung 9 (1965) 217-229.

2. Jac. M. Anthonisse
Determination of the Chromatic Number of a Graph, Report S 393,
Mathematisch Centrum, Amsterdam, March 1968.