RAIRO. RECHERCHE OPÉRATIONNELLE

GILBERT LAPORTE

Solving a family of permutation problems on 0-1 matrices

RAIRO. Recherche opérationnelle, tome 21, n° 1 (1987), p. 65-85

<http://www.numdam.org/item?id=RO_1987__21_1_65_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 21, n° 1, février 1987, p. 65 à 85)

SOLVING A FAMILY OF PERMUTATION PROBLEMS ON 0-1 MATRICES (*)

by Gilbert LAPORTE [1]

Abstract. – *This paper examines a family of permutation problems on* 0-1 *matrices. These problems arise in scheduling and in archaeological seriation. Three exact algorithms and two heuristic procedures are described and compared.*

Keywords : Permutations; seriation; archaeology.

Résumé. – *On décrit dans cet article une famille de problèmes de permutation définis sur des matrices* 0-1. *Ces problèmes sont souvent associés à la fabrication d'horaires et à la sériation archéologique. L'article décrit et compare trois algorithmes exacts et deux procédures heuristiques pour leur résolution.*

Mots clés : Permutations; sériation; archéologie.

1. INTRODUCTION

Consider the following problem recently posed by Telgen [37]. A company department has 12 employees, each of whom is involved in one or more of 22 projects. It is proposed to hold, on the same day, 22 meetings, one for each of the projects. Since many employees will have to attend more than one meeting, the schedule should be such that the number of movements in and out of the meeting room is minimized.

As noted by Telgen, this problem can be formulated as a travelling salesman problem (TSP): given m points and $C=(c_{ij})$, the associated distance matrix, determine the shortest Hamiltonian circuit through the m points. Let $A=(a_{ik})$ be a binary matrix whose entries indicate whether employee k is involved in project i ($a_{ik}=1$) or not ($a_{ik}=0$). It can be assumed that A contains at least one non-zero entry in every column. In a general situation, A will be an

(*) Received November 1985.
(2) École des Hautes Études Commerciales de Montréal, 5255 avenue Decelles, Montréal H3T 1V6, Canada.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/87/01 65 21/$ 4.10

$m \times n$ matrix. Adjoin to A an artificial row of zeros: $a_{m+1,k}=0 (k=1, \ldots, n)$. Now consider a permutation $(\varphi(1), \ldots, \varphi(m+1))$ of the $m+1$ matrix rows and the circular string of 0's and 1's in column k. This string contains:

s_1 sequences (0, 1) corresponding to entries in the meeting room;

s_2 sequences (1, 0) corresponding to exits from the meeting room;

s_3 sequences (1, 1) corresponding to two consecutive periods spent in the meeting room;

s_4 sequences (0, 0) corresponding to two consecutive periods out of the meeting room.

Observe that (i) $s_1=s_2$; (ii) $s_1+s_3=\sum_{i=1}^{m+1} a_{ik}$ since every 1 is preceded either by a 0 or by a 1; (iii) $s_1+s_2+s_3+s_4=m+1$.

The number of times employee k will enter or leave the meeting room is equal to $s_1+s_2=2s_1$. Minimizing this number is equivalent to maximizing s_3 or to minimizing $s_1+s_2+s_4$.

The "distance" between any two rows i and j of A can then be defined as the total number of occurences of the (0, 1), (1, 0) and (0, 0) patterns, i. e.

$$c_{ij}=n-\sum_{k=1}^{n} a_{ik}\, a_{jk} \tag{1}$$

Telgen's problem then consists of determining a permutation $(\varphi(1), \ldots, \varphi(m+1))$ of the $m+1$ matrix rows in order to minimize

$$c_{\varphi(m+1),\,\varphi(1)}+\sum_{i=1}^{m} c_{\varphi(i),\,\varphi(i+1)}.$$

This problem can be solved directly by means of any of the known TSP algorithms (*see* [23] for a recent survey and [17, 24] for two similar applications). Without loss of generality, we can assume that $\varphi(m+1)=m+1$ so that every entry of the last row of the reordered matrix $A'=(a'_{ij})$ is equal to zero. A' is such that the number of gaps of zeros between two consecutive ones in any column is minimized. Equivalently, the number of pairs (i, k) for which $i \leqq m$ and $a'_{ik}=a'_{i+1,k}=1$ is maximized. If z^* denotes the optimal value of the TSP solution, the minimum number of movements in and out of the meeting room is equal to

$$v^*=2(z^*-n(m+1)+\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}). \tag{2}$$

The optimal solution to Telgen's example is displayed in table I. This solution contains 7 gaps of zeros, but not all of them have the same importance. If all 22 meetings have the same duration and are scheduled in the same 8 hour day, each of them will last approximately 22 minutes. Then, small gaps (such as the first one for employee G) may represent pure wastes of time whereas larger ones may be used efficiently. However, the TSP algorithm does nothing to control the lengths of the various gaps: it can only minimize their number.

TABLE I

Telgen's final schedule. Only 5 employees have to enter the meeting room more than once. The schedules of employees E, G and H contain unusable pieces of free time.

Project numbers	Employees											
	A	*B*	*E*	*F*	*G*	*I*	*H*	*L*	*K*	*J*	*D*	*C*
1	1	1										
4		1	1									
6		1		1	1							
5				1	1							
20				1								
10					1	1						
16						1						
11						1	1	1				
13								1	1	1		
15									1	1		
22									1		1	
14							1		1			
9									1			
12					1				1			
17					1			1				
18			1					1				
21								1				
19								1			1	
3											1	
7			1								1	
2											1	1
8												1

The situation may be different in a university context: time spent by students between classes is rarely conducive to fruitful study. Most students would rather have a schedule in which the time spread between the first and last lecture is minimized. If the rows of A represent lectures and the columns, students, a suitable objective in this case would be to permute the lectures in order to minimize the sum over all columns of A' of the spread between the first and last 1, in other words, the sum of lengths of all gaps.

This problem was addressed by Adelson *et al.* [1] and by Norman [31] in connection with orchestra rehearsal:

"Not every player in an orchestra is required for every piece that is to be practised at a rehearsal. If every player arrives just in time for the first piece for which he is required and

leaves immediately after the last piece for which he is required, in what order should the pieces be played so as to minimize the total time (in man-hours) spent by the orchestra in rehearsal?" ([31], p. 58)

In this problem, the rows of A represent pieces of music, and the columns, musicians.

A similar ordering problem arises in the context of archaeological seriation. One of the problems frequently encountered by archaeologists is that of chronologically ordering graves on the basis of the objects they contain. It is often hypothesized that a particular object will appear in graves only over a limited time period. Letting $a_{ij}=1$ if object j is found in grave i and $a_{ij}=0$ otherwise, "the chronological best permutation of the rows of the matrix is that which minimizes $\sum_j r_j$ where the summation is over the columns of the matrix and where r_j is the difference between the row numbers of the first and last non-zero entries in column j" [7]. Extensive surveys on the seriation problem can be found in [14, 27, 34].

The archaeological seriation problem has counterparts in various other fields related to social and behavioral sciences. "For instance, a political scientist may wish to place legislators along a liberal-conservative dimension, a psychologist may attempt to seriate subjects along a moral or development continuum..." ([16], p. 133), etc.

2. DETERMINING THE OBJECTIVE FUNCTION

All of the above problems can be formulated in a unified way: determine the permutation $(\varphi(1), \ldots, \varphi(m))$ which yields the lowest value of some objective function z. In order to define z, consider the "gaps of zeros" between successive 1's in the columns of A'. These can be indexed by s. Every gap s has a length l_s and can be assigned a weight w_s. The objective is then to minimize $z=\sum_s w_s l_s$.

(i) In the orchestra scheduling and archeological seriation problems, w_s is generally equal to 1.

(ii) In Telgen's scheduling problem, $w_s=1/l_s$ since every gap in A' makes a contribution of 1 to the objective function. In this problem, it would make sense however, to weight the smaller gaps more heavily since they are more likely to be wasted, for example

$$w_s=\begin{cases} 2^{T-l_s} & (l_s \leqq T) \\ 0 & (l_s > T) \end{cases} \tag{3}$$

where T is a suitably chosen constant. To illustrate, setting $T=6$ would mean that only those gaps lasting no more than 132 minutes would be considered in the objective function.

(iii) Laporte and Desroches [21] recently treated a course scheduling problem. There were 9 one hour periods per day in which lectures could be scheduled. Every gap between classes generated a penalty of 2 and so did every hour of free time between classes. Then w_s was equal to $2+2/l_s$.

Finally, it may prove useful, in some instances, to assign column dependent weights to the various gaps in order to take into account the fact that two people's time may be valued differently.

3. CLASSIFICATION OF ALGORITHMS

In all but some very particular instances, the problem under consideration is NP-hard. Some of the "easiest" cases correspond in fact to a TSP. One instance of the problem is however relatively easy to solve. Exact algorithms are known to have been developed for the following classes of problems.

Class 1: There exists a permuted matrix A' such that in every column, all 1's are consecutive. In this case, irrespective of the weights w_s, the value of the objective function is equal to zero and all problems considered above are equivalent. Fulkerson and Gross [9] present a polynomial algorithm which determines whether A possesses the "consecutive 1's property" and if so, which identifies the optimal permuted matrix A'. Bartholdi *et al.* [2] apply a slight variation of this property to the construction of cyclic staff schedules.

Class 2: $w_s=1/l_s$ for all s. This is Telgen's original problem which can be solved by means of any TSP algorithm.

Class 3: $w_s=1$ for all s. We refer to this problem as the seriation problem (SP) as it arises directly from the archaeological context. This case appears to be the most arduous of all. A description of some exact algorithms for the SP is provided in section 4. These include

(i) a TSP guided search procedure;

(ii) a linear programming (LP) based algorithm;

(iii) dynamic programming.

In the current state of knowledge, problems which fall in neither of the above categories and problems which are simply too large to be solved

exactly, should be handled by means of a suboptimal heuristic approach. Such examples are given in section 5.

Computational results using both the exact and approximate algorithms are reported in section 6.

4. THREE EXACT ALGORITHMS FOR THE SERIATION PROBLEM

The first two classes of problems described in section 3 present no real interest insofar as they can be solved in a polynomial number of steps or that they reduce to a TSP. The SP is the problem that offers the real challenge. Here is a description of three exact algorithms for the SP, totally or partially developed by the author.

(i) A TSP guided search procedure

This method exploits the relationship between the SP and the TSP. Let z^*_{TSP} and z^*_{SP} be the optimal values of the TSP (minimizing the number of gaps of zeros) and of the SP (minimizing the sum of their lenghts) associated with A. Also, let $\underline{z}^*_{TSP}$ and $\underline{z}^*_{SP}$ be lower bounds on z^*_{TSP} and z^*_{SP} respectively. Now suppose the TSP is solved by a branch and bound algorithm which successively fixes arcs (i, j) at 0 or 1 in a search tree. The Little *et al.* method [26] constitutes one of the earliest examples of this type of algorithm. More efficient algorithms belonging to the same family have later been proposed by several authors (*see* for example Miliotis [29] and Carpaneto and Toth [3]). In such algorithms, a lower bound $\underline{z}^*_{TSP}$ on z^*_{TSP} is derived at every node of the search tree: the TSP solution obtained by exploring the current branch will contain at least $\underline{z}^*_{TSP}$ gaps of zeros; but since each of these gaps has a length of at least one, $\underline{z}^*_{TSP}$ also constitutes a valid lower bound on the total gap length in the optimal SP solution associated with the current branch. It is therefore valid to set $\underline{z}^*_{SP}$ equal to $\underline{z}^*_{TSP}$. But this bound can be improved. At the current node of the search tree, the partial solution contains paths of arcs fixed at 1. Let v_h be the number of nodes of path h and consider only those paths for which $v_h \geqq 4$. For a path h with the ordered sequence of nodes $i_1, \ldots, i_{v_h}$, define B_h, the matrix consisting of rows $i_1, \ldots, i_{v_h}$ of A. Define

$z_{TSP}(B_h)$: the number of gaps of 0's (between successive 1's) in columns of B_h;

$z_{SP}(B_h)$: the sum of lengths of these gaps.

Then, $\underline{z}^*_{SP}$ can be increased by $\sum_h (z_{SP}(B_h) - z_{TSP}(B_h))$. For example, let

$$A = \begin{array}{ccccccc|c} & & & & & & & \textit{Row number} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 4 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 5 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 6 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 8 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 9 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 10 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 11 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 12 \end{array}$$

and suppose that at a given node of the search tree, the following arcs have been fixed at 1: (4, 8), (8, 1), (1, 11), (11, 6), (3, 7), (7, 10), (10, 5). To these, correspond the two paths (4, 8, 1, 11, 6) and (3, 7, 10, 5) and the two submatrices

$$B_1 = \begin{array}{ccccccc|c} & & & & & & & \textit{Row number} \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 4 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 11 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 6 \end{array}$$

and

$$B_2 = \begin{array}{ccccccc|c} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 3 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 10 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 5 \end{array}$$

The values of $z_{SP}(B_h)$ and $z_{TSP}(B_h)$ are then computed: $z_{SP}(B_1) = 13$, $z_{SP}(B_2) = 6$, $z_{TSP}(B_1) = 7$, $z_{TSP}(B_2) = 4$. Therefore $\underline{z}^*_{SP}$ can be set equal to $\underline{z}^*_{TSP} + (13-7) + (6-4) = \underline{z}^*_{TSP} + 8$.

(ii) An LP based algorithm

Doran and Powell [7] proposed the following LP formulation for the SP. Define

$1+x_i$: the position in A' of row i of A;

$1+\alpha_k$, $1+\beta_k$: the row numbers of the first and last non-zero entry of column k of A'.

Then the problem is

$$\text{(P 1)} \qquad \text{minimize} \sum_{k=1}^{n} (\beta_k - \alpha_k)$$

subject to

$$\sum_{i=1}^{m} x_i = \frac{1}{2} m(m-1) \tag{4}$$

$$|x_i - x_j| \geqq 1 \qquad (i<j;\ i, j=1, \ldots, m) \tag{5}$$

$$\alpha_k \leqq x_i \leqq \beta_k \qquad (a_{ik}=1) \tag{6}$$

$$\beta_k - \alpha_k \geqq -1 + \sum_{i=1}^{m} a_{ik} \qquad (k=1, \ldots, n) \tag{7}$$

$$\alpha_k, \beta_k \geqq 0 \qquad (k=1, \ldots, n) \tag{8}$$

$$0 \leqq x_i \leqq m-1 \qquad (i=1, \ldots, m). \tag{9}$$

In (P 1), constraints (4) and (5) ensure that $(x_1, \ldots, x_m)$ constitute a permutation of $(0, \ldots, m-1)$ so that there is no need to specify integrality conditions on the x_i's. Constraints (9) are in fact redundant. Constraints (5) can be replaced by the following dichotomy:

$$x_i - x_j \geqq 1 \qquad \text{or} \qquad x_i - x_j \leqq -1. \tag{10}$$

A strategy for solving (P 1) could then be to replace each of these disjunctions by

$$\begin{aligned} x_i - x_j + m\,\delta_{ij} &\geqq 1 \\ x_i - x_j + m\,\delta_{ij} &\leqq m-1 \\ \delta_{ij} &= 0 \quad \text{or} \quad 1 \end{aligned} \tag{11}$$

as suggested by Dantzig [4]. Alternatively, one could avoid introducing the binary δ_{ij} variables by branching directly on the two alternatives of (10). In both cases the results are likely to be disappointing since the relaxation of

(P1) obtained by dropping constraints (5) will yield a solution in which

$$\beta_k - \alpha_k = -1 + \sum_{i=1}^{m} a_{ik} \qquad (k = 1, \ldots, n) \tag{12}$$

In other words, the gap between the objective value of (P1) and that of its relaxation will be large and slow to fill.

We now propose some alterations to (P1). These do not remove the difficulty just mentioned, but yield totally unimodular subproblems which can be easily handled by a network based algorithm. This formulation is obtained by removing constraints (4) and (9) from (P1):

(P2) $$\text{minimize} \sum_{k=1}^{n} (\beta_k - \alpha_k)$$

subject to

$$x_i - \beta_k \leqq 0 \qquad (a_{ik} = 1) \tag{13}$$

$$x_i - \alpha_k \geqq 0 \qquad (a_{ik} = 1) \tag{14}$$

$$\beta_k - \alpha_k \geqq -1 + \sum_{i=1}^{m} a_{ik} \qquad (k = 1, \ldots, n) \tag{15}$$

$$x_{m-1} - x_m \leqq -1 \tag{16}$$

$$\left.\begin{array}{l} x_i - x_j \leqq -1 \quad \text{or} \quad x_j - x_i \leqq -1 \\ (i < j;\ i = 1, \ldots, m-2;\ j = 1, \ldots, m) \end{array}\right\} \tag{17}$$

In (P2), all variables are unrestricted in sign. Since the coefficient matrix contains only two non-zero elements in every row, a "1" and a "-1", it is totally unimodular. And since all constraints have an integer right-hand side, all basic solutions will be integer. We will also make use of the following result.

PROPOSITION: *If A contains at least two non-zero elements in every column, and no row of zeros, then the optimal solution to (P2) is such that $(x_1, \ldots, x_m)$ constitutes a permutation of m consecutive integers.*

Proof: First note that the conditions imposed on A are not restrictive since the value of the objective function is unaffected by the order of elements in columns having less than two 1's. Such columns can in fact be removed from the matrix. Similarly, all rows of zeros will be positioned at the beginning or at the end of the matrix in the optimal solution. These too can be removed.

In (P2), constraints (17) imply that the distance between any two x_i's is at least 1. Consider a feasible solution $(\bar{x}_1, \ldots, \bar{x}_m; \bar{\alpha}_1, \ldots, \bar{\alpha}_n; \bar{\beta}_1, \ldots, \bar{\beta}_n)$ to (P2) such that two consecutive values $\bar{x}_j$ and $\bar{x}_l$ differ by more than 1. We will show that this solution is not optimal, by displaying a better solution $(\bar{x}'_1, \ldots, \bar{x}'_m; \bar{\alpha}'_1, \ldots, \bar{\alpha}'_n; \bar{\beta}'_1, \ldots, \bar{\beta}'_n)$.

Let $\bar{x}_l - \bar{x}_j = 1 + \Delta$ where $\Delta > 0$. Also let $K = \{k: \bar{\alpha}_k \leqq \bar{x}_j \text{ and } \bar{\beta}_k \geqq \bar{x}_l\}$. The set K is non-empty since row j and row l of A each contain at least one non-zero element. Then we can set

$$\begin{aligned}
\bar{x}'_i &= \bar{x}_i \qquad (\bar{x}_i \leqq \bar{x}_j) \\
\bar{x}'_i &= \bar{x}_i - \Delta \qquad (\bar{x}_i \geqq \bar{x}_l) \\
\bar{\alpha}'_k &= \bar{\alpha}_k \qquad (\bar{\alpha}_k \leqq \bar{x}_j) \\
\bar{\alpha}'_k &= \bar{\alpha}_k - \Delta \qquad (\bar{\alpha}_k \geqq \bar{x}_l) \\
\bar{\beta}'_k &= \bar{\beta}_k \qquad (\bar{\beta}_k \leqq \bar{x}_j) \\
\bar{\beta}'_k &= \bar{\beta}_k - \Delta \qquad (\bar{\beta}_k \geqq \bar{x}_l)
\end{aligned} \tag{18}$$

This new solution satisfies all constraints of (P2) and the value of its objective function is reduced by $\Delta |K|$. ■

In (P2), the value of $1 + x_i$ does not necessarily represent the position of row i of A in A', as was the case in (P1). But since the x_i's are consecutive integers, these positions are now given by

$$x'_i = 1 + x_i - \min_i \{x_i\}. \tag{19}$$

Finally, constraint (16) has been introduced in order to avoid considering solutions which are merely symmetries of one another.

The dual of (P2) will only have a "1" and a "-1" in every column and can be represented by a directed network with nodes associated with constraints and arcs with variables. The dichotomy $x_i - x_j \leqq -1$ or $x_j - x_i \leqq -1$ now corresponds to a choice between arc (i, j) and arc (j, i) on the network. As before, this can be handled by branch and bound. But now, all the subproblems are totally unimodular: each consists of determining a minimum cost flow. For this, any standard network package such as an out-of-kilter code [8] or RNET [12] can be used.

Formulations similar to (P2) arise in a variety of other permutation problems. Consider for example the "sequencing through a junction" problem described by Nicholson [30]: m routes intersect at a common point P. From each route i, a unit arrives at P at time u_i and takes a time v_i to cross P. A minimum delay c_{ij} is incurred between the passage of i and the passage of j

through P (e. g. trains crossing a junction, m jobs having to be processed through a machine with change-over times, etc.). We want to minimize the time at which the last unit finishes passing through P. Define

x_i: the time at which unit i starts crossing P;

y: the time at which the last unit finishes crossing P.

We then formulate the problem as follows:

(P3) minimize y

subject to

$$x_i \geqq u_i \qquad (i=1, \ldots, m) \tag{20}$$

$$y - x_i \geqq v_i \qquad (i=1, \ldots, m) \tag{21}$$

$$x_i - x_j \geqq c_{ji} + v_j \quad \text{or} \quad x_j - x_i \geqq c_{ij} + v_i \qquad (i<j\leqq m). \tag{22}$$

If $u_i = v_i = 0$ $(i=1, \ldots, m)$ in (P3), and if C satisfies the triangle inequality, i. e. $c_{ik} \leqq c_{ij} + c_{jk}$ $(i, j, k = 1, \ldots, m)$, the problem reduces to that of determining a shortest Hamiltonian open path.

Both these problems can be solved by using the branch and bound approach with network flow subproblems suggested for the SP. However, in the case of the shortest Hamiltonian path problem, it is more efficient to use one of the known TSP algorithms (see for example [3]).

(iii) A dynamic programming algorithm

Adelson *et al.* [1] and Norman [31] provide a dynamic formulation for the orchestra rehearsal problem. This formulation can be specialized to the SP. Any state can be described by a vector $(p_1, \ldots, p_m)$ where $p_i = 1$ if piece i has already been played and $p_i = 0$ otherwise. If $p_i = 0$, define $p_i' = 1$. Let $f(p_1, \ldots, p_m)$ be the lowest increase in the objective function, attainable from state $(p_1, \ldots, p_m)$. The aim is to compute $f(0, \ldots, 0)$. Define:

d_i: the duration of piece i and

$$l_k = \begin{cases} 1 & \text{if } \sum_{\substack{i \\ p_i = 1}} a_{ik} > 0 \text{ and } \sum_{\substack{i \\ p_i = 0}} a_{ik} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

i. e. $l_k = 1$ if an only if musician k has already played and still has another piece to play.

Then the solution can be obtained by considering all 2^m states through the following backward recursion, starting from state $(1, \ldots, 1)$ and with $f(1, \ldots, 1)=0$:

$$f(p_1, \ldots, p_m)= \min_{\substack{i \text{ such} \\ \text{that } p_i=0}} \{d_i[\sum_k \max(a_{ik}, l_k)]+f(p_1, \ldots, p_i', \ldots, p_m)\} \quad (24)$$

In this expression, d_i is included in the sum (i) if musician k plays in piece i or (ii) if he has already played and still has at least one piece to play, i. e. if he has to remain in the rehearsal room. The optimal solution is obtained by going through the sequence of optimizing states [i. e. those yielding the minimum in (24)], starting from $(0, \ldots, 0)$. The SP formulation is easily derived by setting $d_i=1$ $(i=1, \ldots, m)$.

In order to illustrate the computations, consider the following example.

Piece i	Duration d_i	Musician 1	2	3	4	5
1	2	1	0	0	1	0
2	4	1	1	0	0	1
3	8	0	1	1	0	0
4	5	1	0	0	1	1

First set $f(1, 1, 1, 1)=0$. The values of the next four states are relatively easy to compute and will not be explicited: $f(0, 1, 1, 1)=4$, $f(1, 0, 1, 1)=12$, $f(1, 1, 0, 1)=16$ and $f(1, 1, 1, 0)=15$. We now compute $f(0, 0, 1, 1)$. At state $(0, 0, 1, 1)$, $p_1=p_2=0$ and $p_3=p_4=1$, i. e. only pieces 3 and 4 have already been scheduled. From this state, it is possible to move to state $(1, 0, 1, 1)$ and to state $(0, 1, 1, 1)$. First consider state $(1, 0, 1, 1)$: this state is attained from $(0, 0, 1, 1)$ by setting $p_1=1$. Only musicians 1 and 4 play in piece 1; therefore, $a_{1,1}=a_{1,4}=1$ and $a_{1,2}=a_{1,3}=a_{1,5}=0$. Moreover, player 2 has already played (in piece 3) and still has a piece to play (piece 2): therefore $l_2=1$. Similarly, we compute $l_5=1$. Musician 3 has already played in piece 3, but has no more piece to play: therefore $l_3=0$. This information is sufficient to compute

$$d_1 \sum_{k=1}^{5} \max(a_{1k}, l_k)=2(1+1+0+1+1)=8. \quad (25)$$

The cost of going from state (0, 0, 1, 1) to state (1, 0, 1, 1) is therefore equal to $8+f(1, 0, 1, 1)=20$. Similarly, the cost of going from state (0, 0, 1, 1) to state (0, 1, 1, 1) is equal to $16+f(0, 1, 1, 1,)=20$.

It is easy to verify that the optimal solution consists of going through the following sequence of states: (0, 0, 0, 0), (1, 0, 0, 0), (1, 0, 0, 1), (1, 1, 0, 1), (1, 1, 1, 1), i. e. the pieces should be played in the order 1-4-2-3 and the total rehearsal time is equal to 47.

5. HEURISTIC ALGORITHMS

Seriation problems which are either too large or too difficult to solve by an exact algorithm have traditionally been tackled by means of interchange heuristics (see for example [6, 15, 19, 35]). These algorithms often take into account some particular features of the SP. Since the class of problems described in section 1 is more general, we suggest the use of a relatively general procedure such as Lin's 3-opt algorithm [25] or, if the value of m becomes too large, Or's Or-opt algorithm [32].

The 3-opt procedure is relatively powerful. It dominates (as far as the value of the objective is concerned) those presented in [6, 15, 19, 35].

(i) Consider an initial permutation and compute the associated objective.

(ii) Consider all permutations of rows taken 3 at a time.

(iii) Implement those permutations which yield an improvement in the objective function.

(iv) Repeat until no further improvement can be achieved by this process.

Several independant runs can be made, starting from R different initial solutions. Lin shows that the probability that this procedure will yield the global optimum is approximately equal to

$$p = 1 - (1 - 2^{-m/10})^R. \tag{26}$$

But since each step of the 3-opt algorithm requires $O(m^3)$ comparisons, this method may prove infeasible for large scale problems. Instead, we suggest the use of the Or-opt algorithm, a simplified version of the 3-opt algorithm. The Or-opt algorithm consists of successively inserting in all positions of the matrix all blocs of 3 *consecutive* rows and of implementing the profitable insertions. This procedure is then repeated with blocs of 2 rows and 1 row. The complexity of each step the Or-opt procedure is only $O(m^2)$. According to a recent study by Golden and Stewart [10], when applied to the TSP, the Or-opt and the 3-opt procedures perform just as well, but the former is much quicker. In the following section, we present some comparisons of our own for the SP.

6. COMPUTATIONAL RESULTS

6.1 Small and medium size seriation problems

The three exact algorithms described in section 4, the 3-opt and the Or-opt procedures were programmed in FORTRAN and tested on 30 classical seriation problems extracted from the literature on anthropology, archaeology and epigraphy [36]. The precise references are provided in table II. The three exact algorithms were tested on a CDC 7600 while the 3-opt and Or-opt searches were implemented on a VAX/VMS computer, a much slower machine.

The TSP based search was constructed from the Little *et al.* [26] algorithm for the TSP. The algorithm fared pretty well in problems in which z^*_{SP} was close to z^*_{TSP} (problems 4, 6-13, 19-20, 25-30) but was more disappointing in other cases. It failed to reach a global optimum within 600 seconds in 7 problems out of 30. The use of a more efficient TSP algorithm (for example [29]) would certainly have reduced the computation time but would have done little to reduce significantly the growth of the search tree in problems in which the gap between z^*_{SP} and z^*_{TSP} was too large.

For the LP based algorithm, we used for the solution of the sub-problems a version of the out-of-kilter algorithm [8] adapted to our particular problem. The computational times given in table II indicate that this algorithm is dominated by the other two. Despite its poor performance on the seriation problem, we believe that this type of algorithm could constitute a valuable tool in problems possessing a similar structure (such as the "sequencing through a junction" problem [30]) but in which the lower bound on the optimum obtained at the root of the search tree would be higher.

The DP algorithm had without any doubt the best performance. Problems with $m \leqq 16$ and a reasonable value of n can be solved in $O(mn\,2^m)$ operations in a relatively short time with this algorithm. The size of the array required to store the 2^m states prevented us from solving larger problems. Note that contrary to the previous two algorithms, computation times are here directly related to the number of columns.

The 30 problems used for testing the three exact algorithms were then attempted by performing 10 independent runs of the 3-opt and of the Or-opt procedures. The best optimal value obtained in each case was then compared with the global optimum determined by the dynamic programming algorithm. As can be seen from table III, each algorithm found the global optimum at least once in each of the 30 problems: 5.4 times on the average for the 3-opt algorithm and 5.57 times on the average for the Or-opt algorithm. The latter

TABLE II

Small and medium size seriation problems: execution times in seconds (CDC 7600) for three exact algorithms

Problem	Reference	m	n	TSP based	LP based (network model)	Dynamic programming
1	[5], p. 500	16	26	>600	>600	83.687
2	[20], p. 58	12	11	78.782	>600	1.790
3	[15], p. 99	13	57	>600	>600	18.526
4	*Ibid.*	11	23	4.934	>600	1.646
5	*Ibid.*	12	16	244.634	>600	2.559
6	*Ibid.*	5	3	0.003	0.005	0.006
7	[13], pl. 123(*)	5	5	0.003	0.013	0.007
8	*Ibid.*	6	6	0.003	0.126	0.013
9	*Ibid.*	7	7	0.004	0.389	0.029
10	*Ibid.*	8	8	0.028	1.143	0.067
11	*Ibid.*	9	8	0.207	8.305	0.141
12	*Ibid.*	10	9	0.141	35.911	0.335
13	*Ibid.*	11	13	8.046	507.863	1.002
14	*Ibid.*	12	13	40.268	>600	2.159
15	*Ibid.*	13	16	132.293	>600	5.597
16	*Ibid.*	14	19	498.010	>600	14.016
17	*Ibid.*	15	21	>600	>600	32.865
18	*Ibid.*	16	23	>600	>600	76.065
19	[33], p. 293	8	8	0.049	28.941	0.066
20	[18], p. 74	6	6	0.003	0.007	0.014
21	[11], p. 207	16	17	>600	>600	57.342
22	*Ibid.*	13	16	>600	>600	5.686
23	[36], p. 274	15	49	>600	>600	73.205
24	[28], fig. 201	8	20	0.086	10.580	0.145
25	*Ibid.*	11	20	98.280	>600	1.473
26	[28], p. 629	9	7	0.102	27.821	0.126
27	*Ibid.*	8	9	0.035	3.027	0.074
28	[28], p. 626	8	5	0.010	3.213	0.046
29	[15], p. 103	13	11	0.764	>600	3.805
30	[15], p. 101	14	13	65.215	>600	9.706

(*) Problems 7 to 18 are extracted from Hodson's matrix (pl. 123) by taking the first 5, 6, ..., 16 rows only.

procedure was also 35% faster than the 3-opt algorithm, the comparison becoming more favourable to the Or-opt algorithm for larger values of m. Thus, despite of performing a less thorough search than the 3-opt algorithm, the Or-opt algorithm seems just as likely to find the global optimum and has a lower time complexity. This confirms the conclusions of Golden and Stewart [10].

6.2. A large scale seriation problem

The 3-opt and the Or-opt procedures were also tested on a relatively large scale seriation problem described by Hodson [13]. Rows of zeros and columns

TABLE III

Small and medium size seriation problems: computational results for the 3-opt and the OR-opt algorithms

				3-opt algorithm			OR-opt algorithm		
Problem	m	n	(A)	(B)	(C)	(D)	(E)	(F)	(G)
1	16	26	254	254	1	74.16	254	3	46.60
2	12	11	85	85	1	10.60	85	1	7.87
3	13	57	376	376	5	67.01	376	4	57.96
4	11	23	154	154	10	13.86	154	10	11.67
5	12	16	98	98	2	15.24	98	1	9.84
6	5	3	6	6	10	0.20	6	9	0.31
7	5	5	7	7	10	0.28	7	10	0.37
8	6	6	11	11	10	0.58	11	10	0.69
9	7	7	14	14	6	1.13	14	10	1.20
10	8	8	19	19	6	2.05	19	3	1.93
11	9	8	23	23	7	3.20	23	2	2.82
12	10	9	25	25	6	5.39	25	4	3.85
13	11	13	32	32	3	11.94	32	3	7.21
14	12	13	39	39	5	15.62	39	3	10.43
15	13	16	46	46	8	26.31	46	6	15.39
16	14	19	48	48	4	41.18	48	5	22.52
17	15	21	55	55	4	61.15	55	8	33.54
18	16	23	58	58	7	94.49	58	1	47.92
19	8	8	38	38	7	1.75	38	8	1.72
20	6	6	12	12	4	0.57	12	10	0.83
21	16	17	34	34	4	71.59	34	5	34.17
22	13	16	40	40	2	27.35	40	5	18.85
23	15	49	315	315	4	114.24	315	10	79.01
24	8	20	82	82	10	4.43	82	9	4.64
25	11	20	94	94	3	13.49	94	4	9.33
26	9	7	32	32	4	2.61	32	2	2.43
27	8	9	29	29	10	2.07	29	4	2.48
28	8	5	21	21	5	1.32	21	10	1.47
29	13	11	90	90	3	13.65	90	6	9.84
30	14	13	91	91	1	21.69	91	1	14.78

(A) Global optimum.
(B) Best heuristic solution value.
(C) Number of runs for which global optimum was found.
(D) Total execution time in seconds (VAX/VMS) for 10 runs.
(E) Best heuristic solution value.
(F) Number of runs for which global optimum was found.
(G) Total execution time in seconds (VAX/VMS) for 10 runs.

containing less than two non-zero elements were first removed as in Doran [6]. This resulted in a 63×69 matrix. Interested readers are referred to Laporte [21] for a full description of the matrix.

On this matrix, it was impossible to complete the 3-opt algorithm in less than 3,000 CPU seconds on the VAX/VMS computer. The Or-opt search took between 1,100 and 2,670 seconds.

TABLE IV

Solution having a very low total spread. Some employees, for example D, G and L have several short interruptions.

Project numbers	Employees											
	A	*B*	*E*	*F*	*G*	*I*	*H*	*L*	*K*	*J*	*D*	*C*
20				1								
5				1	1							
6		1		1	1							
1	1	1										
4		1	1									
10					1	1						
16						1						
11						1	1	1				
14							1		1			
17					1			1				
12					1				1			
13								1	1	1		
15									1	1		
9									1			
22									1		1	
18			1					1				
7			1								1	
19								1			1	
21								1				
3											1	
2											1	1
8												1

Starting from 10 random orderings, the Or-opt procedure yielded solutions having the following values (computed as in the objective function of (P1)): 415, 523, 416, 521, 430, 432, 441, 412, 432, 563. The solution provided by Hodson [13] has a value of 456 whereas the best known solution for this problem had a value of 414 (see Doran [6]). In one instance, the Or-opt procedure produced a better solution (value of 412).

We then applied the Or-opt procedure to Doran's ordering: a better solution with an objective value of 411 was determined after 1,152 seconds.

6.3. Telgen's problem

Finally, Telgen's problem was solved by the Or-opt procedure using two different objectives. In the first case, w_s was set equal to 1 for all s (see section 1), in other words, the problem was treated as an SP. The solution is shown in table IV: while this solution may be very good for the oechestra rehearsal problem, it is not very appropriate in a business environment: a lot of time is wasted between meetings. Employees *D*, *G* and *L*, in particular,

TABLE V

Solution in which no employee has short interruptions between successive meetings

Project numbers	Employees											
	A	*B*	*E*	*F*	*G*	*I*	*H*	*L*	*K*	*J*	*D*	*C*
12					1				1			
5				1	1							
6		1		1	1							
10					1	1						
11						1	1	1				
18			1					1				
7			1								1	
2											1	1
3											1	
22									1		1	
9									1			
1	1	1										
16						1						
20				1								
21								1				
17					1			1				
19								1			1	
13								1	1	1		
15									1	1		
14							1		1			
4		1	1									
8												1

have several short interruptions. From this point of view, Telgen's schedule was much better.

In order to obtain the schedule displayed in table V, weights were assigned to small interruptions according to formula (3) and with $T=6$. The solution shows that several employees have broken days, but short interruptions are infrequent. Assuming all meetings have the same duration, there is no gap of less than 2 hours.

7. CONCLUSION

A familly of permutation problems on 0-1 matrices were considered. These have applications in several fields and can be formulated in a unified way. According to the objective function and to the structure of the 0-1 matrix, several instances of the problems can be distinguished:

(i) problems having the "consecutive l's property" for which there exists a polynomial algorithm [9];

(ii) problems which reduce to a TSP;

(iii) the seriation problem which poses the greatest challenge; for $m \leqq 16$, dynamic programming appears to constitute the best exact algorithm.

For seriation problems having a large number of rows or for problems which do not fall into one of the above three categories, we recommend the use of the Or-opt procedure.

ACKNOWLEDGMENTS

The author is grateful to the Canadian Sciences and Engineering Research Council (grant A4747) and to the Quebec Government (FCAR grant 86EQ3084) for their financial support. Thanks are also due to Fabien Chauny, Gilles Savard and Serge Taillefer for their assistance with programming, and to an anonymous referee for his valuable comments.

REFERENCES

1. R. M. Adelson, G. Laporte et J. M. Norman, *A Dynamic Programming Formulation with Diverse Applications*, Operational Research Quarterly, Vol. 27, 1976, pp. 119-121.
2. J. C. Bartholdi III, J. B. Orlin et H. D. Ratliff, *Cyclic Scheduling via Integer Programs with Circular Ones*, Operations Research, Vol. 28, 1980, pp. 1074-1085.
3. G. Carpaneto et P. Toth, *Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem*, Management Science, Vol. 26, 1980, pp. 736-743.
4. G. B. Dantzig, *On the Significance of Solving Linear Programming Problems with Some Integer Variables*, Econometrica, Vol. 28, 1960, pp. 30-44.
5. P. Dempsey et M. Baumhoff, *The Statistical Use of Artifact Distributions to Establish Chronological Sequence*, American Antiquity, Vol. 28, 1963, pp. 496-509.
6. J. E. Doran, *Computer Analysis of Data from the La Tène Cemetery at Münsingen-Rain*, Mathematics in the Archaeological and Historical Sciences, F. R. Hodson *et al.* Eds., Edinburgh University Press, 1971, pp. 422-431.
7. J. E. Doran et S. Powell, *Solving a Combinatorial Problem Encountered in Archaeology*, Some Research Applications of the Computer, Atlas Computer Laboratory, 1972.
8. L. R. Ford et D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.
9. D. R. Fulkerson et O. A. Gross, *Incidence Matrices and Interval Graphs*, Pacific Journal of Mathematics, Vol. 15, 1965, pp. 835-855.
10. B. L. Golden et W. R. Stewart, *Empirical Analysis of Heuristics*, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, E. L. Lawler *et al.* Eds., Wiley, 1985, pp. 207-250.
11. K. Goldmann, *Some Archaeological Criteria for Chronological Seriation*, Mathematics in the Archaeological and Historical Sciences, F. R. Hodson *et al.* Eds., Edinburgh University Press, 1971, pp. 202-208.

12. M. D. GRIGORIADIS et T. HSU, *RNET-The Rutgers Minimum Cost Network Flow Subroutines*, Rutgers University, New-Brunswick, N.J., 1979.

13. F. R. HODSON, *The La Tène Cemetery at Münsingen-Rain*, Stämpfli, 1968.

14. F. R. HODSON, D. G. KENDALL et P. TAUTU, *Mathematics in the Archaeological and Historical Sciences*, Edinburgh University Press, 1971.

15. F. HOLE et M. SHAW, *Computer Analysis of Chronological Seriation*, Rice University Studies, Vol. 53, 1967.

16. L. J. HUBERT, *Some Applications of Graph Theory and Related Non-Metric Techniques to Problems of Approximate Seriation: the Case of Symmetric Proximity Measures*, British Journal of Mathematical and Statistical Psychology, Vol. 27, 1974, pp. 133-153.

17. L. J. HUBERT et F. B. BAKER, *Applications of Combinatorial Programming to Data Analysis: the Travelling Salesman Problem and Related Problems*, Psychometrika, Vol. 43, 1978, pp. 81-91.

18. D. G. KENDALL, *Some Problems and Methods in Statistical Archaeology*, World Archaeology, Vol. 1, 1969, pp. 68-76.

19. I. KIVU-SCULY, *On the Hole-Shaw Method of Permutation Search*, Mathematics in the Archaeological and Historical Sciences, F. R. HODSON *et al.* Eds., Edinburgh University Press, 1971, pp. 253-254.

20. R. S. KUZARA, R. G. MEAD et K. A. DIXON, *Seriation of Anthropological Data: a Computer Program for Matrix Ordering*, American Anthropologist, Vol. 68, 1966, pp. 1442-1455.

21. G. LAPORTE, *A Comparison of Two Norms in Archaeological Seriation*, Journal of Archaeological Science, Vol. 3, 1976, pp. 249-255.

22. G. LAPORTE et S. DESROCHES, *The Problem of Assigning Students to Course Sections in a Large Engineering School*, Computers and Operations Research, Vol. 13, 1986, pp. 387-394.

23. E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN et D. B. SHMOYS, *The Travelling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

24. J. K. LENSTRA et A. H. G. RINNOOY KAN, *Some Applications of the Travelling Salesman Problem*, Operational Research Quarterly, Vol. 26, 1975, pp. 717-734.

25. S. LIN, *Computer Solution of the Travelling Salesman Problem*, Bell System Technical Journal, Vol. 44, 1965, pp. 2245-2269.

26. J. D. C. LITTLE, K. G. MURPHY, D. W. SWEENEY et C. KAREL, *An Algorithm for the Travelling Salesman Problem*, Operations Research, Vol. 11, 1963, pp. 972-989.

27. W. H. MARQUARDT, *Advances in Archaeological Seriation*, Advances in Archaeological Method and Theory, Vol. 1, M. B. SCHIFFER Ed., Academic Press, 1978, pp. 257-314.

28. B. J. MEGGERS et C. EVANS, *Archaeological Investigation in the Mouth of the Amazon*, Bulletin 167, Smithsonian Institute, Bureau of American Ethnology, 1957.

29. P. MILIOTIS, *Integer Programming Approaches to the Travelling Salesman Problem*, Mathematical Programming, Vol. 10, 1976, pp. 367-378.

30. T. A. J. NICHOLSON, *Permutation Programming and its Applications*, Ph.D. Thesis, University of London, 1970.

31. J. M. NORMAN, *Elementary Dynamic Programming*, Crane, Russak & Company, Inc., New York, 1977.

32. I. OR, *Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking*, Ph.D. Thesis, Northwestern University, Evanston, IL, 1976.

33. W. S. ROBINSON, *A Method for Chronologically Ordering Archaeological Deposits*, American Antiquity, Vol. 16, 1951, pp. 293-301.

34. A. SHUCHAT, *Matrix and Network Models in Archaeology*, Mathematics Magazine, Vol. 57, 1984, pp. 3-14.

35. R. SIBSON, *Some Thoughts on Sequencing Methods*, Mathematics in the Archaeological and Historical Sciences, F. R. HODSON *et al.* Eds., Edinburgh University Press, 1971, pp. 263-266.

36. A. STEFAN, *Applications of Mathematical Methods to Epigraphy*, Mathematics in the Archaeological and Historical Sciences, F. R. HODSON *et al.* Eds., Edinburgh University Press, 1971, pp. 267-275.

37. J. TELGEN, *How to Schedule Meetings with a Travelling Salesman Q & D, and Why we Didn't*, Interfaces, Vol. 15, 1985, pp. 89-93.