POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title: On the one-dimensional space allocation problem

Auteurs: Authors: Jean-Claude Picard, & Maurice Queyranne

Date: 1978

Type: Rapport / Report

Référence: Citation: Picard, J.-C., & Queyranne, M. (1978). On the one-dimensional space allocation problem. (Rapport technique n° EP-R-78-48). https://publications.polymtl.ca/5944/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL: https://publications.polymtl.ca/5944/

Version: Version officielle de l'éditeur / Published version

Conditions d'utilisation: Terms of Use: Tous droits réservés / All rights reserved

Document publié chez l'éditeur officiel
Document issued by the official publisher

Institution: École Polytechnique de Montréal

Numéro de rapport: Report number: EP-R-78-48

URL officiel: Official URL:

Mention légale: Legal notice:

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

Génie Industriel

ON THE ONE-DIMENSIONAL SPACE ALLOCATION PROBLEM

JEAN-CLAUDE PICARD AND MAURICE QUEYRANNE

DEPARTEMENT DE GENIE INDUSTRIEL
ECOLE POLYTECHNIQUE DE MONTREAL

RAPPORT TECHNIQUE NO EP78-R-48

NOVEMBRE 1978.

Ecole Polytechnique de Montréal

CA2PQ
UP4
78R48

Campus de l'Université
de Montréal
Case postale 6079
Succursale 'A'
Montréal, Québec
H3C 3A7

8 DEC 1978

ON THE ONE-DIMENSIONAL SPACE ALLOCATION PROBLEM

Don

JEAN-CLAUDE PICARD AND MAURICE QUEYRANNE

À CONSULTER
SUR PLACE

DEPARTEMENT DE GENIE INDUSTRIEL
ECOLE POLYTECHNIQUE DE MONTREAL

RAPPORT TECHNIQUE NO
NOVEMBRE 1978.

ABSTRACT

The One-Dimensional Space Allocation Problem arises when locating rooms along a corridor, when setting books on a shelf, when allocating information items among the "cylinders" of a magnetic disk, or when storing products in certain warehouses. The rooms (objects) to be located have a known length (or width) and, for each pair of rooms, there is a given number of "trips" between them; the problem is to find a sequencing of the rooms which minimizes the total traveled distance. This paper deals first with two particular cases. In the rooted tree case, previous results obtained by Adolphson and T.C. Hu for the Linear Ordering Problem are extended and provide efficient solution methods. In addition, it is shown that most of the assumptions made by Adolphson and Hu are necessary, since that relaxing any of them leads to an NP-complete problem. In the case of independent destinations (solved by Bergmans and Pratt when all lengths are equal), an unimodality property is shown for an optimum solution, but this is not sufficient to lead to a "good" algorithm and the problem is shown to be NP-complete. For the general problem, an exact algorithm using dynamic programming is described and computationally tested.

1- INTRODUCTION

In this paper we consider a layout problem which arises in a variety of contexts. As a prototype example, consider the problem of designing the physical arrangement of rooms within an hospital department. In many cases, the available space is approximately rectangular, with one long edge of the rectangle having windows, and the following design is considered: set a corridor along the main dimension of the rectangle with patients rooms on one side of the corridor-the side with windows-and the service rooms on the other side. Once this type of arrangement has been decided, a next step would be to determine an efficient disposition of the service rooms. One plausible measure of efficiency is the distance traveled between these rooms. We intendedly let it undetermined by whom or which this distance is traveled: according to the activities of the department and to the objectives retained by the planner, it may be the distance traveled by the patients, by the nurses or by some other important entity: wheeling beds or chairs, blood containers. etc... The physical data concerns the space needed for each service room: since the width of the rooms will be given for the type of layout we are considering, this information will be in the case of rectangular rooms summarized in the length of the room. The data pertinent to the use of the rooms could be a single measure of this usage for each room, but, in order to more accurately represent the system, it appears more advisable to use the traffic, i.e. the number of trips per day or unit of time, between every pair of rooms. Thus one could measure the efficiency of a layout by the total traveled distance, which is defined as the sum, over all pairs of rooms, of the door-to-door distance of these rooms in the layout, weighted by the traffic between

the rooms. The problem is then to find a sequence of rooms which minimizes this total traveled distance.

The problem may appear in other contexts, with a very different interpretation. If we wish to arrange our books on a shelf, we would like to put together the books which are likely to be used at the same time; here the "room length" becomes thickness of the books, and the traffic the frequency (or probability) of picking a given book just after another one. This naive example cannot be directly extended to a whole library where the physical distance is no longer measured along one dimension, but arises again when it is desired to classify the books along one scale. Assume that we are given a scale from 0 to 10, and we wish to assign an interval to every subject. The width of the interval must reflect the importance of the subject and it is desired to have related subjects in close locations. In this problem, (called the "linear typological aggregation" in [26]) we are provided with some connectedness measure (see [21] for a good discussion) which corresponds to the traffic in the hospital example, and one possible measure of the efficiency of a scale could be the total weighted distance between all pairs of subjects (in [21], Morse uses another measure involving the squared distances).

The book and shelf problem arises in the computer domain when it is desired to efficiently assign information items to certain types of memories [24],[23],[11]. For instance, consider the problem of assigning files to the cylinders of a disk. Assume that every file requires a

given number of cylinders which are to be adjacent, and that the time necessary to reach a file is proportional to the number of cylinders between the present location of the read heads and the beginning of the file. Provided statistics are available for the frequencies of requiring a file just after another one (traffic), reducing the average access time is again a layout problem of the type discussed here. While our interest in the problem came from this memory allocation context, we met it again in a study of a warehouse layout for a brewery. In this company, the distribution process has been computerized and one bottleneck has been identified at the loading of the trucks. One of the solutions which had to be considered was the following: a forklift carries the different palettes to a main conveyor which takes them to the trucks. The forklift operator follows a loading sequence which has been determined by the computerized distribution system. The various palettes are picked at the end of other conveyors bringing them from the stocking area. Several conveyors may be assigned to the most frequent types of palettes, and these conveyors, coming from the same area in the warehouse, are to be adjacent. Given statistics on the orders, or more precisely on the loading sequences, the efficiency of the operations is increased when the various types of palettes are assigned to the conveyors in order to minimize the distance traveled by the forklift.

The general problem has been defined by Simmons [28] as the One Dimensional Space Allocation Problem (ODSAP). Several special cases of ODSAP have been studied and we refer to the recent paper by Chan and

Francis [7] for a good review and bibliography. In the present paper we have tried to delineate the limits between some solvable cases and presumably intractable ones. After presenting some mathematical formulations of the general problem in Section 2, we study in Section 3 the rooted tree case introduced by Adolphson and Hu [2] and show how the assumptions made by these authors appear necessary for obtaining an efficient solution method. In section 4, we show that the case of independent destinations, which is efficiently solved when all room length are equal [4], [23], [11], becomes also NP-complete for arbitrary room length. Since the two previous cases, as well as the general case, are important in practice, we propose an $0\ (n2^n)$ dynamic programming algorithm for solving the ODSAP with n rooms; this algorithm appears to be much more efficient than previously known solution methods. Due to memory requirements, the dynamic programming approach seems limited to problems with less than 20 rooms, and we briefly discuss, in the last section, some other possible, exact or heuristic, approaches for solving larger problems.

2- SOME FORMULATIONS OF THE ONE DIMENSIONAL SPACE ALLOCATION PROBLEM (ODSAP).

The following developments will be put in terms of rooms and lengths. Consider n rooms $r_1, r_2, \ldots, r_n$; let ℓ_i denote the length of the room r_i, and w_{ij} the number of trips from room r_i to room r_j (traffic intensity). Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ denote a permutation of the room indices; i.e. the first (leftmost) room is $r_{\pi(1)}$,

the second room is $r_{\pi(2)}$ and so on. Denote by π^{-1} the inverse of this permutation: $\pi^{-1}(i)$ is the position of r_i in the permutation π. Consider a permutation π and two distinct rooms r_i and r_j: the <u>distance</u> between r_i and r_j with respect to this permutation, assumed to be taken between their centers, is equal to the half-length of r_i, plus the lengths ℓ_k of all rooms r_k which are between r_i and r_j in π, plus the half-length of r_j:

$$d^c(i,j,\pi) = \frac{1}{2}\ell_i + \sum_{k \in B(i,j,\pi)} \ell_k + \frac{1}{2}\ell_j \qquad (2\text{-}1)$$

where
$$B(i,j,\pi) = \{k: \pi^{-1}(i) < \pi^{-1}(k) < \pi^{-1}(j)\} \text{ if } \pi^{-1}(i) < \pi^{-1}(j)$$
$$= \{k: \pi^{-1}(j) < \pi^{-1}(k) < \pi^{-1}(i)\} \text{ if } \pi^{-1}(j) < \pi^{-1}(i)$$

is the set of indices of the rooms between r_i and r_j in π (as usually, a sum over an empty set of indices is assumed to be zero).

The <u>One Dimensional Space Allocation Problem</u> (ODSAP) is the problem of finding a permutation π which minimizes the weighted sum of the distances between all pairs of rooms, i.e.

$$\text{(ODSAP)} \quad \min_{\pi} z(\pi) = \sum_i \sum_{j \neq i} w_{ij} d^c(i,j,\pi) \qquad (2\text{-}2)$$

A first remark [28] allows us to get rid of the half room lengths. Defining

$$d(i,j,\pi) = \sum_{k \in B(i,j,\pi)} \ell_k \tag{2-3}$$

it appears that

$$z(\pi) = \sum_i \sum_{j \neq i} w_{ij}\, d(i,j,\pi) + K \tag{2-4}$$

where $$K = \sum_i \sum_{j \neq i} \frac{1}{2} w_{ij} (\ell_i + \ell_j)$$

is a constant, independent of π. In the following, we will always consider this formulation, in which the distances are taken between the closest walls of the rooms. In particular, the distance between adjacent rooms is zero.

An immediate consequence follows: consider a room r_i and all of the permutations in which r_i is in first position: ℓ_i does not appear in the cost $z(\pi)$ of any such permutation π (in other words their costs remain the same when ℓ_i is changed to any arbitrary value). This remark will be extended to provide a basis to a dynamic programming approach described in section 5.

Another simple remark concerns the symmetry of the solutions to the problem. Let π' be the <u>symmetric</u> of π, i.e.

$$\pi'(t) = \pi(n-t+1) \text{ for all } t=1,\ldots, n$$

Then clearly $z(\pi') = z(\pi)$. In other words, we can exchange the right and the left hand sides in our definition. Hence, we could somewhat simplify the problem by only considering, for instance, the permutations in which

$\pi^{-1}(1) \leqslant \frac{1}{2}$ n, i.e. room r_1 is in the leftmost side of the corridor. This remark will be applied in order to reduce the computational requirements of the solution method. However, we might also be interested in instances of ODSAP in which no such symmetry is allowed: assume for instance that the corridor has one dead end, and all the patients arrive from the other end. This situation may be modelled by adding a dummy room r_o with the restraint that this room must be in the first position; this room r_o represents the initial location of the patients entering in the corridor; as pointed out above, the length ℓ_o is essentially unimportant, and the entries w_{oi} have an immediate interpretation. The situation in which there is some flow of patients through both ends of the corridor might be modelled by a similar device.

A third simple remark deals with an immediate symmetry in the objective function. Since $d(i,j,\pi) = d(j,i,\pi)$ for all pairs i,j and all permutations, the objective function may be restated as:

$$z(\pi) = \sum_{i} \sum_{j>i} (w_{ij}+w_{ji})d(i,j,\pi) \tag{5}$$

In lieu of the natural ordering of the indices, one could also use the permutation π to distinguish between the couples (i,j) and (j,i):

$$z(\pi) = \sum_{t=1}^{n-1} \sum_{u=t+1}^{n} (w_{\pi(t),\pi(u)}+w_{\pi(u),\pi(t)})d\left(\pi(t),\pi(u),\pi\right) \tag{6}$$

By developping the expression of $d(\pi(t),\pi(u),\pi)$, one obtains

$$z(\pi) = \sum_{t=1}^{n-1} \sum_{u=t+1}^{n} (w_{\pi(t),\pi(u)},+w_{\pi(u),\pi(t)}) \sum_{v=t+1}^{u-1} \ell_{\pi(v)} \tag{7}$$

(as a convention, we set a sum to zero when its argument is empty).

There is another possible formulation for the objective function, which could be called "length-directed": by regrouping all the occurences of $\ell_{\pi(v)}$ in (7), it comes:

$$z(\pi) = \sum_{v=2}^{n-1} \ell_{\pi(v)} \sum_{t=1}^{v-1} \sum_{u=v+1}^{n} w_{\pi(t),\pi(u)} + w_{\pi(u),\pi(t)} \qquad (8)$$

This formulation (8) will prove useful for some particular cases of ODSAP, see section 4.

The ODSAP can be formulated as an integer programming problem. Love and Wong [18] give such a formulation, together with some (deceptive) computational results with a standard integer programming code. We propose a more natural formulation of ODSAP as a <u>Cubic Assignment Problem</u>: for all i and t, define a 0-1 variable x_{it} which is equal to one if and only if room r_i is assigned to the tth position. Clearly there must be exactly one room per position, and one position for every room; the following formulation of the ODSAP follows from (7):

$$\left[\begin{array}{lll}
\min\limits_{x} & \sum\limits_{t=1}^{n-1} \sum\limits_{u=t+1}^{n} \sum\limits_{i=1}^{n} \sum\limits_{j\neq i} (w_{ij}+w_{ji}) x_{it} x_{ju} \sum\limits_{v=t+1}^{u-1} \sum\limits_{k\neq i,j} \ell_k x_{kv} & \\
\text{subject to} & \sum\limits_{i} x_{it} = 1 & \text{for all } t \\
 & \sum\limits_{t} x_{it} = 1 & \text{for all } i \\
 & x_{it} = 0 \text{ or } 1 & \text{for all } i,t.
\end{array}\right.$$

We call this a "Cubic Assignment Problem" since it appears as a generalization of the Linear and the Quadratic Assignment Problems [10], [16], [17] involving products $x_{it}x_{ju}x_{kv}$ of three variables. Such a problem appears

as even more difficult than the Quadratic Assignment Problem (QAP). In the conclusion, we point out that even a Linear Assignment Relaxation of ODSAP is much more difficult to solve than the corresponding relaxation [10], [16] of QAP. On the other hand, the ODSAP has properties which may be derived from its single-dimensionality, and which do not hold for 2-dimensional or general QAP.

An interesting location problem closely related to the ODSAP is the one-dimensional version of the QAP of the Koopmans-Beckman type [17]: given n points (locations) $P_1, P_2, \ldots, P_n$ with coordinate $a_1, a_2, \ldots, a_n$ on a line, and given n activities and a matrix W of traffic intensities between activities, find a one-to-one assignment of the activities to the locations in order to minimize the total weighted distance. This problem, which we call the <u>Generalized Linear Ordering Problem</u> (GLOP) can be stated in the following QAP format:

$$\left[\begin{array}{lll} \min \sum_i \sum_j w_{ij} \sum_t \sum_u |a_t - a_u| \, x_{it} x_{ju} & & \\ \text{s.t.} \quad \sum_i x_{it} = 1 & \text{for all } t & \\ \quad \sum_t x_{it} = 1 & \text{for all } i & \\ \quad x_{it} = 0 \text{ or } 1 & \text{for all } i,t. & \end{array}\right.$$

This problem seems to have received little attention in the literature, while it shares with ODSAP many properties owing to its single dimensionality.

The Linear Ordering Problem (LOP) [2] is at the "intersection" of ODSAP and GLOP: it is defined as an ODSAP with all

rooms having the same length, say unity, or equivalently as a GLOP with the location being regularly placed, say at coordonates 1, 2, ..., n. The LOP has been proven to be NP-complete [9], hence it follows that both ODSAP and GLOP are NP-complete. As a consequence, we can estimate that the existence of a "good" algorithm is unlikely for this whole class of problems. The following developments will explore the limits between tractable and untractable cases of ODSAP and will provide solution methods for the general problem.

3- THE ROOTED TREE CASE

In this section, we consider cases in which the trip matrix W has the following property: the first column is zero and every other column has exactly one non-zero element which is located above the main diagonal. For an illustration, think of the following process: a test performed in a room of the corridor determines in which room the patient will be moved, another test is applied in this room and so on until a diagnosis is completed. Here we specify that the patients must check-in first in room r_1 and that any other room may receive patients from only one given other room. We observe that this situation is more likely to arise when modeling a search among information items than actual clinical diagnosis.

Associated with the matrix W is a directed graph with nodes representing the rooms and arcs corresponding to every non-zero entry in W. The above condition on W is clearly equivalent to the following definition of a <u>rooted tree,</u> with root r_1: there is no arc with head r_1 and exactly one arc with head r_j for all $j > 1$. The unique room r_i such that there exists an arc (r_i, r_j) is called the <u>predecessor</u> of r_j and will be denoted by $p(r_j)$. A characteristic property of a rooted tree is that there is a unique path from the room r_1 to a given node; in other words, if two patients are in the same room, they have visited exactly the same rooms, and in the same order, since they checked in.

This problem has been studied by Adolphson and Hu [2] when all the lengths are equal, and we will extend their results to the case of arbitrary positive lengths. We state additional assumptions pertaining

to the model of Adolphson and Hu:

A1: the root r_1 must be located in the first position.

A2: the patients will stay in the last room they visit; in other words, we do not take into account their journey after this point.

A3: the predecessor of r_j $(j>1)$ must be located between the root r_1 and r_j.

These three assumptions will be discussed later in this section. For a room r_j $(j>1)$ the entry $w_{p(j)j}$ may be interpreted as the number of patients which visit r_j. According to this interpretation, it seems natural to impose the following condition

$$w_{p(j)j} \geqslant \sum_k w_{jk} \quad \text{for } j > 1 \tag{3-1}$$

In case of strict inequality, the difference may be interpreted as the number of patients which will stay in r_j. The following theorem shows that under this condition (3-1), the assumption A3 can always be verified.

<u>THEOREM I</u>:

If assumptions A1 and A2 are verified and if the matrix W satisfies condition (3-1), then there exists an optimum permutation which verifies assumption A3.

The proof of this proposition is given in Appendix.

Figure 2 gives an example of matrix W which does not satisfy condition (3-1) and the corresponding rooted tree. When all room lengths are equal, one can verify (by simple enumeration) that the optimum permutation is (1,3,2,4,5), which does not satisfy assumption (3-1).

	r_1	r_2	r_3	r_4	r_5
r_1	0	1	0	0	0
r_2	0	0	2	1	0
r_3	0	0	0	0	0
r_4	0	0	0	0	1
r_5	0	0	0	0	0

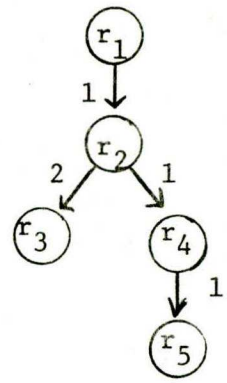

Figure 2: A matrix W and its associated rooted tree.

In [2, p. 419] Adolphson and Hu give an O(nlogn) algorithm for solving the Optimal Linear Ordering Problem (OLOP). It is not difficult to see that their problem is equivalent to ODSAP. Indeed, with each node j in their rooted tree are associated a weight and a length which correspond to the entries $w_{p(j)j}$ and ℓ_j in our problem. If y (π) denotes the objective function value of an arbitrary solution π in the OLO problem, that is,

$$y(\pi) = \sum_j w_{p(j)j} \sum_{t=1^{-1}(p(j))+1}^{\pi^{-1}(j)} \ell_\pi(t)$$

then the value $z(\pi)$ of the corresponding solution in the ODSA problem (with center-to-center distances) differs from $y(\pi)$ by a constant, independent of π:

$$z(\pi) = y(\pi) + \frac{1}{2} \sum_{j>1} (w_{p(j)j} - \sum_k w_{jk})\ell_j$$

Hence, the Adolphson and Hu's algorithm solves the rooted tree ODSAP under assumptions A1, A2 and either A3 or (3-1). Adolphson and Hu noted that the rooted tree ODSAP (under assumptions A1, A2 and A3) is closely related to the following one-machine scheduling problem: given n jobs $J_i (i=1,\ldots,n)$ with processing times p_i and deferral rates v_i, and a rooted tree with the jobs as nodes (plus an additional root), find a sequence π of these jobs on a single machine which satisfies the precedence relation induced by this rooted tree (assumptions A1 and A2), and minimizes the total cost $\sum_i v_i F_i(\pi)$ where $F_i(\pi)$ is the completion time of job J_i in the sequence π, i.e. the total processing time of all jobs requered before J_i, including J_i itself. The algorithm by Adolphson and Hu solves the problem by scanning the nodes (jobs) of the tree from the leaves to the root. In [1], Adolphson proposed an other O(nlogn) algorithm, which scans the jobs in nonincreasing order of the ratios v_i/p_i; this algorithm could also be applied to our problem as well as other algorithms by Garey [8] and Horn [12].

Now, consider variations of the assumption A2. If the patients go back to the registration desk (rooms r_1) after their diagnostic is completed and we want to take also in account this part of their trip, we obtain the assumption A2′:

> A2′: all the patients must go back to room r_1 after their diagnostic is completed.

Of course, this assumption makes sense only if condition (3-1) is satisfied; otherwise there would be some "magic" creation of patients in the process and we do not intend to study such systems! Before stating the main

result for this problem, we have to introduce another condition, now on the length of the rooms: we will say that condition (3-2) is satisfied if the length of any room r_j with $j \neq 1$ and $p(j) \neq 1$, is not smaller than the length of the room to be visited before:

$$\text{for all } j \neq 1 \qquad p(j) \neq 1 \Rightarrow \ell_j \geqslant \ell_{p(j)} \tag{3-2}$$

This could be interpreted by considering that the tests become more and more complicated, and thus require larger rooms, as the diagnosis progresses; or, equivalently, the simplest tests, those which require small rooms, are performed first. In particular, condition (3-2) is met when all room lengths are equal, i.e. in the OLO problem.

THEOREM 2

If assumptions A1 and A2′ are verified, if the matrix W satisfies condition (3-1) and if the lengths ℓ_j satisfy condition (3-2) then there exist an optimum permutation for the rooted tree ODSAP which verifies assumption A3.

The proof of this proposition is given in Appendix.

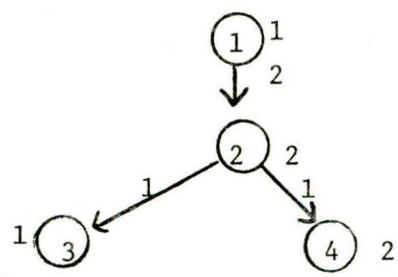

FIGURE 4.

Figure 4 gives an instance of ODSAP with a matrix W satisfying condition (3-1), but with lengths which do not satisfy condition (3-2). It appears immediately that the optimum solution of ODSAP under assumptions A1 and A2′ is given by the permutation (1,3,2,4), which does not verify assumption A3.

For a solution satisfying assumptions A1 and A3, it is not difficult to see that its objective value under assumption A2′ is exactly twice its value under assumption A2. Hence a rooted tree ODSAP under assumptions

A1, A2′ and either A3 or (3-1) and (3-2) can be solved using the O(nlogn) algorithm by Adolphson and Hu [2] On the other hand, the rooted tree ODSAP becomes trivial under assumptions A1, A3 and A2′′: all the patients must go to the rightmost end of the corridor when their diagnostic is completed. In this case, each patient must travel at least the whole length of the corridor and any permutation which satisfy assumption A3 will be optimal.

We now proceed to discuss the assumptions A1 and A3 and show that the rooted tree ODSAP becomes NP-complete when any of these assumptions is relaxed. To this purpose, we introduce an instance of the rooted tree ODSAP, which has applications in the assignment of disk cylinder to files in some operating systems: assume a disk contains several files, one of which, called the VTOC (Volume Table Of Content), keeping the address (position) of all the files in this disk; every request for a file is processed in the following way: the read head is positionned to the VTOC which is searched for the address of the desired file; then the head is moved to this position and the file is read. Given the number of cylinders required for each file and statistics for the frequency of calls to each file, the problem is to allocate the cylinders to the files in order to minimize the average access time. Here, we make the following simplifications:

- each file requires an integer number of cylinders, which must be adjacent

- the access time is mainly determined by the physical moving of the head between cylinders.

In addition we note that the VTOC usually uses one cylinder so that the total displacement between two accesses to the VTOC is roughly equal to the distance (number of cylinders) between the VTOC and the farmost end of the file (see Figure 5)

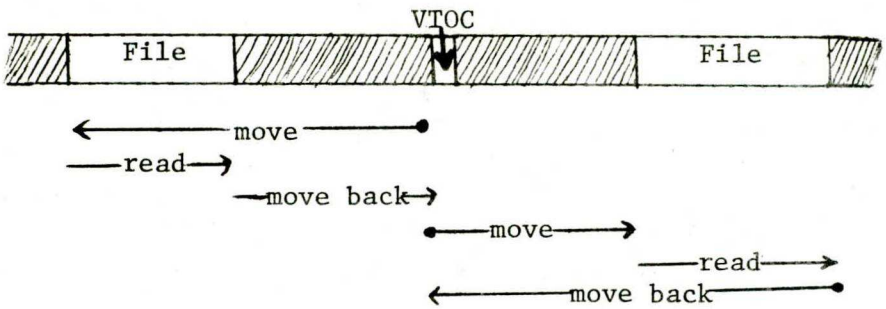

FIGURE 5: Distances for the VTOC problem

It appears that this problem is an instance of the rooted tree ODSAP in which the rooted tree has a particularly simple structure: one root (the VTOC) and all the other nodes (the files) attached to it; hence the assumption A3 is always verified. But here we do not impose the assumption A1: it is clear that the VTOC must be located somewhere "in the middle" of the cylinders. The objective function uses the "extreme" distances and assumption A2′; by an argument similar to the one used in the previous section, it is clear that we can reduce to either center-to-center or closest-walls distances; also, as noted above, the objective function under assumption A2 is equal to the half of its value under assumption A2′. (Note also that asumption A2′′ has less meaning here, since it implies that the root (VTOC) will be located at one end, reducing to the previous problem).

The VTOC problem will be shown to be equivalent to the following two-machines scheduling problem: given two parallel identical machines

and n jobs J_i $(i=1,\ldots,n)$ with processing times p_i and deferral rate v_i, assign each job to one machine and find the sequences on these machines in order to minimize the total cost $\sum_i v_i F_i$, where F_i is the completion time of job J_i, that is the total processing times of all jobs processed before J_i (including J_i itself) an the same machine as J_i. The equivalence with the VTOC problem can be seen by identifying files with jobs, lengths (number of cylinders) ℓ_i with processing times p_i, number of calls w_{1i} with deferral rates v_i and considering as processed on the first machine the set of jobs corresponding to files located before the VTOC, and on the second machine those located after the VTOC, the first job on a machine corresponds to the file closest to the VTOC and so on. Note that the objective function in the scheduling problem is equal to the objective function in the VTOC problem under assumption A2, that is the half of its value under the more natural assumption A2′.

A well-known result in Scheduling Theory (e.g. Theorem 2.4 in [3]) implies that in an optimum solution, the jobs must be processed, on each machine, according to a non-increasing order of the ratios v_i/p_i [3, pp. 119-120]. It follows that an optimum solution to the VTOC problem must possess the following <u>unimodality property</u>: the sequence of the ratios w_{1i}/ℓ_i must be unimodal, first non-decreasing up to the VTOC, then non-increasing. This property reduces the number of solutions to be considered from n! to 2^{n-1}. When all the deferral rates (frequencies) are equal, the problem can be efficiently solved using a simple dispatching rule (see [3, pp. 118-119]); a similar dispatching rule also applied when all the processing times (lengths) are equal. However for general

deferral rates, the problem has been shown to be NP-complete, see [6], [5]. According to this equivalence and since the VTOC problem is an instance of the rooted tree ODSA, we obtain the following theorem:

<u>Theorem 3</u> The following rooted tree ODSAP's under assumptions

(i) A3 and A2

(ii) A3 and A2′

(iii) A2

(iv) A2′

are NP-complete.

This theorem shows that it is unlikely that the approach of Adolphson and Hu could be extended, except perhaps for very particular cases which would not include the VTOC problem, without the assumption A1.

Now, we restore assumption A1 for the remainder of this section and discuss the assumption A3:

<u>Theorem 4</u>: The rooted tree ODSAP under assumptions A1 and A2 is NP-complete

<u>Proof:</u> it is clear that this problem is in NP. To prove it is NP-complete, we exhibit a reduction from the rooted tree ODSAP under only assumption A2 (proven NP-complete by Theorem 3) to the present problem: consider an instance of the rooted tree ODSAP with r rooms (including the roots r_1), an integer matrix W and integer lengths ℓ_i; define another rooted tree ODSAP by adding a dummy root r_o with an arbitrary length ℓ_o, such that p(1) = 0 and define w_{o1} to be small enough, specifically:

$$w_{o1} < 1/\sum_{i=1}^{n} \ell_i$$

(In order to insure the new matrix to be integer, it is for instance suffi-

cient to multiply W by $2\sum_i \ell i$ and to set $w_{o1}=1$). There is a one-to-one correspondence between solutions to the original problem and solutions to the associated problem we have constructed which verify assumption A1; their objective functions values, under assumption A2, differ only by w_{o1} times the distance between the leftmost end of the corridor and the center of r_1 (see Figure 5 for an example of this reduction). Since in the original problem, a non-optimal solution must have an objective function value at least one unit greater than the optimum and w_{o1} is small enough, it follows that an optimum solution to the associated problem must be also optimum for the original problem. Hence, if we had an efficient algorithm for solving the rooted tree ODSAP under assumptions A1 and A2, we could use it, through this reduction, to solve any instance of the NP-complete rooted-tree ODSAP under assumption A2 only.

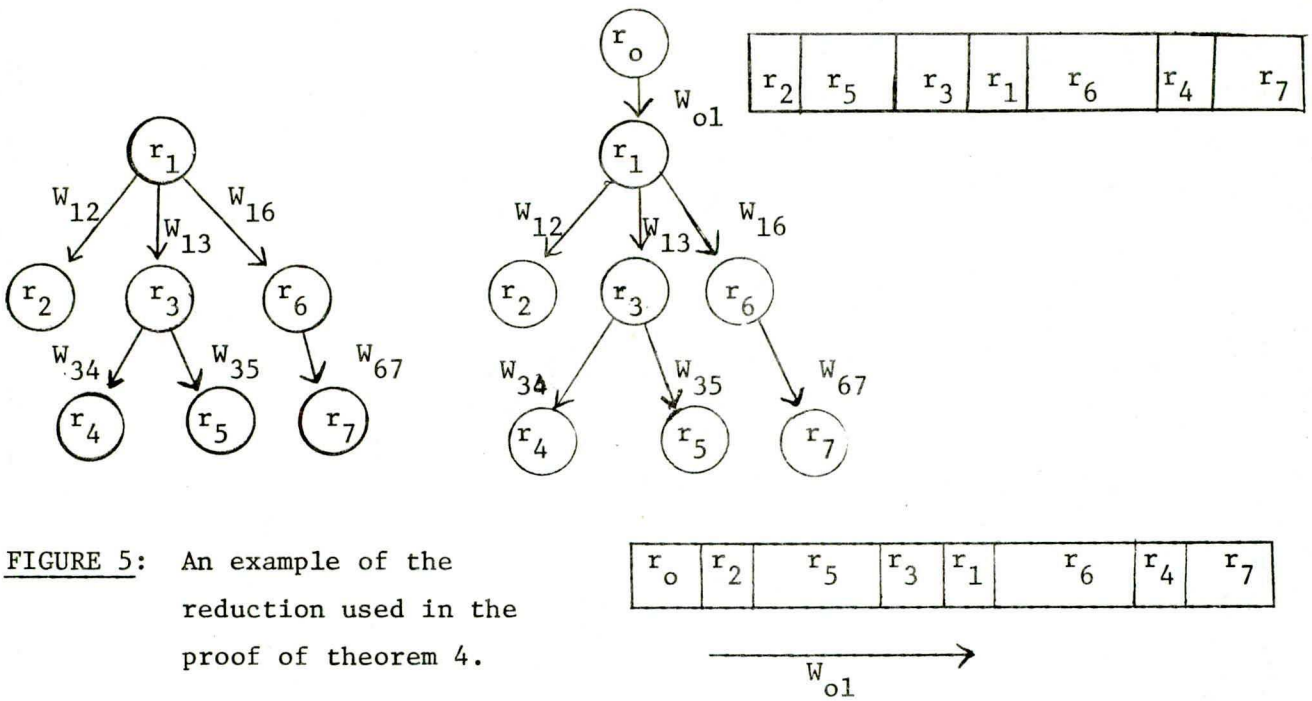

FIGURE 5: An example of the reduction used in the proof of theorem 4.

As it was shown by theorem 1, a rooted tree ODSAP under assumptions A1 and A2 can be efficiently solved when condition (3-1) holds; for the general case, the problem becomes NP-complete. Hence, it does not seem that the approach of Adolphson and Hu could be extended without assumption A3.

Finally, we leave as an open question the status of the rooted tree ODSAP under assumptions A1 and A2$'$, when condition (3-1) or condition (3-2) does not hold. However, in sight of the previous theorems, we strongly suspect it is again an NP-complete problem.

4- INDEPENDENT DESTINATIONS

Consider the particular case of the ODSAP in which n nonnegative numbers $p_1, p_2, \ldots, p_n$ are given such that

(ODSAPID) $$w_{ij} = p_i p_j \text{ for all } i, j. \qquad (4\text{-}1)$$

This case is named the case of <u>independent destinations</u> according to the following interpretation: consider that p_j represents the probability that the room j will be the next destination of a trip, this probability being, independent from the rooms visited before j. This defines a Markov chain and (assuming all p_i positive) the steady-state probabilities are equal to p_i for state (room) i and the transition probabilities are equal to $p_i\ p_j$ for the transition from state i to state j. Note that the logical constraint

$$\sum_{i=1}^{n} p_i = 1$$

can be derived from (2-1) by a simple global rescaling the w_{ij}'s.

If, in addition to (2-1), all room lengths are equal, the resulting Linear Ordering Problem can be solved by an ordering algorithm with a running time in O(n logn): after sorting the rooms in a nondecreasing order of the p_i's, alternately assign them to the leftmost and the rightmost free position. The procedure, due to Bergmans [4] and Pratt [23] produces an <u>alternating order</u> (see [7]) with respect to the p_i's, that is an order π verifying

$$p_{\pi(1)} \leqslant p_{\pi(n)} \leqslant p_{\pi(2)} \leqslant p_{\pi(n-1)} \leqslant \ldots$$
$$p_{\pi(n)} \leqslant p_{\pi(1)} \leqslant p_{\pi(n-1)} \leqslant p_{\pi(2)} \leqslant \ldots$$

(this last condition would be satisfied when exchanging "leftmost" and "rightmost" in the above description).

For the general ODSAPID with arbitrary room lengths, we obtain some insight into the solution structure:

THEOREM 4:

There is an optimum solution to ODSAPID which satisfies the following unimodality property: There is an index q such that, for all $i, j \in \{1, \ldots, n\}$

(i) $\pi(i) < \pi(j) \leqslant q$ implies $p_i/\ell_i \leqslant p_j/\ell_j$

(ii) $\pi(i) < \pi(j) \leqslant q$ implies $p_i/\ell_i \leqslant p_j/\ell_j$

Proof:

Let π be an optimum permutation and consider the unique index q such that

$$\sum_{r<q} p_{\pi(r)} \leqslant \frac{1}{2} P < \sum_{r \leqslant q} p_{\pi}(r)$$

where $P = \sum_{i=1}^{n} p_i$

We will show (i) by an adjacent pairwise exchange: suppose the set$\{I = \pi(i) \leqslant \pi(q)\}$ is not in nondecreasing order of the number p_i/ℓ_i; then consider two elements i in position r and j in position r+1 such that $p_i/\ell_i > p_j/\ell_j$ and the permutation π' deduced from π by exchanging i and j, that is

$$\pi'(i) = r+1$$

$$\pi'(j) = r$$

$$\pi'(k) = \pi(k) \text{ for all } k \neq i, j$$

Then $z(\pi') = z(\pi) + p_i \ell_j \left(\sum_{u<r} p_u - \sum_{v>r+1} p_v \right)$

$$+ p_j \ell_i \left(\sum_{v > r+1} p_v - \sum_{u < r} p_u \right)$$

$$= z(\pi) + (p_i \ell_j - p_j \ell_i) \left(\sum_{u < r} p_u - \sum_{v > r+1} p_v \right)$$

Since $r \leqslant q-1$, $\sum_{u < r} p_u < \frac{1}{2} P$

and $\sum_{v > r+1} p_v \geqslant \frac{1}{2} P$

and since, by assumption $p_i/\ell_i > p_j/\ell_j$ it follows that

$$z(\pi') < z(\pi)$$

a contradiction with the assumption of π being optimal.

The relation (ii) is proven by a symmetric argument. #

In sight of this result, it might be reasonable to expect that an ordering approach would solve the ODSAPID. The following theorem indicates that this is unlikely:

<u>Theorem 5:</u> The ODSAPID with arbitrary probalities p_i and lengths ℓ_i is NP-complete.

The proof of Theorem 5 is given in appendix 3.

A particular case of ODSAPID, defined by

$$p_i = 1/n \text{ for all } i=1, \ldots, n$$

is easily solved by an ordering procedure (Property 4 of [7]). We give here a simpler proof than that of [7], by using the relation (8): it comes

$$z(\pi) = \sum_{v=2}^{n} \ell_{\pi(v)} (v-1)(n-v)$$

It is well known that a problem of minimizing, over all the permutations

π, the quantity

$$z(\pi) = \sum_{v=1}^{n} \ell_{\pi(v)} a_v$$

is solved by an ordering procedure which matches the ℓ_j's in a non-increasing order with the a_v's in a nondecreasing order (this is readily established by a simple adjacent pairwise exchange argument). Since here

$$a_v = (v-1)(n-v) = a_{n-v+1} \qquad \text{for } v=1,\ldots,n$$

and

$$a_u < a_v \qquad \text{for } u < v \leqslant \tfrac{1}{2} n,$$

it follows that any permutation π in <u>generalized alternating order</u> [7] of the ℓ_j's, i.e. verifying

$$\max(\ell_{\pi(k)}, \ell_{\pi(n-k+1)}) \leqslant \min(\ell_{\pi(k+1)}, \ell_{\pi(n-k)}) \text{ for all } k \leqslant \tfrac{1}{2} n,$$

is optimal.

5- A DYNAMIC PROGRAMMING APPROACH TO THE GENERAL PROBLEM.

The computational performance of exact solution methods for the general ODSAP has been so far very deceptive: the algorithms of Simmons [28] and Love and Wong [18] have been unable to optimally solve an 11-room problem within rather large time and core limits. In contrast, the dynamic programming algorithm to be presented below solves optimally any 11-room problem in less than a second and within less than 100 K of memory on an IBM 360/75. Since our first computational experiments (1976), the same method has been independently discovered by B. Harris (1977, private communication) and Z. Drezner (1978, private communication from R. Francis). The algorithm presented here is an extension of an algorithm devised by Held and Karp [14] for the Linear Ordering Problem (see also [29]). The present algorithm could be developped using Held and Karp's formalism, but we prefer a simple presentation.

Consider a subset S of $N = \{1,2,\ldots,n\}$, with s elements ($1 \leqslant s \leqslant n$) to be located in the s first positions. The cost $z(\pi)$ of any permutation with $\pi(i) \in S$ for all $i=1,\ldots,s$ can be decomposed as

$$z(\pi) = \sum_{u=1}^{s-1} \sum_{v=u+1}^{s} w_{\pi(u)\pi(v)} \; d(\pi(u),\pi(v),\pi)$$

$$+ \sum_{u=1}^{s} \sum_{v=s+1}^{n} w_{\pi(u)\pi(v)} \; d(\pi(u),\pi(v),\pi)$$

$$+ \sum_{u=s+1}^{n-1} \sum_{v=u}^{n} w_{\pi(u)\pi(v)} \; d(\pi(u),\pi(v),\pi)$$

Let us denote by P the point between S and $\bar{S}$, that is the rightmost end of the last room in S and also the leftmost end of the first room in $\bar{S}$. If the first room starts at the abscissa 0, then P is

located at the abscissa $\ell(S)=\sum_{i\in S}\ell_i$, independendly of the arrangement within S and within $\bar{S}$. Then we can split the distance between a room in S and a room in $\bar{S}$ as:

$$d(\pi(u),\ \pi(v),\ \pi) = d(\pi(u),P,\pi) + d(P,\pi(v),\pi)$$
$$\text{for all } u=1,\ldots,s,\ \text{all } v=s+1,\ldots,n$$

and $z(\pi)$ becomes

$$z(\pi)=\sum_{u=1}^{s-1}\left(\sum_{v=u+1}^{s} w_{\pi(u)\pi(v)}d(\pi(u),\pi(v),\pi) + W_{\pi(u),\bar{S}}\, d(\pi(u),P,\pi)\right)$$
$$+\sum_{u=s+1}^{n-1}\left(\sum_{v=u+1}^{n} w_{\pi(u)\pi(v)}d(\pi(u),\pi(v)+\pi)+W_{\pi(u),S}d(P,\pi(u),\pi)\right)$$

where, for a subset A of N and $i\in N-A$ we note

$$W_{iA}=\sum_{j\in A} w_{ij}$$

Then $z(\pi)$ is decomposed in a term which depends only on the arrangement within S, and a term which depends only on the arrangement within $\bar{S}$. If π is an optimum permutation, then for any s=1,...n the arrangement of the first rooms must be an optimum permutation to a "special" ODSAP with s+1 elements, the s first elements being $\pi(1),\ldots,\pi(s)$ with their own lengths ℓ_i and interactions w_{ij} and a "artificial" (s+1) element which we denote $\bar{S}$, with an arbitrary length (see first remark in section 2) and iteractions $w_{i\bar{S}}$ with the s first elements and which must be located in last position. Of course, the same is true, mutatis mutandis, for $\bar{S}$.

The above observation is the key of the development of the dynamic programming algorithm. Consider any subset $S\subsetneq N$ and denote by f(S) the value of the special ODSAP defined above. Clearly, we have the <u>boundary conditions</u>

$$f(\{i\}) = 0$$

and $\quad f(N) = z(\pi^*)$ where π^* is an optimum ODSAP solution.

For any S such that $2 \leqslant |S| \leqslant n-1$, there must be an element of S in

s-th position and the above decomposition still holds, yielding the <u>recurrence relation</u>

$$f(S) = \min_{i \in S} \left\{ f(S-\{i\} + \ell_i \, (W_{S-\{i\}}, \bar{S}) \right\}$$

where for two disjoint subset A and B of N, we note

$$W_{AB} = \sum_{i \in A} \sum_{j \in B} w_{ij}$$

Then ODSAP can be solved in $O(2^n)$ applications of the recurrence relation, provided that the subsets S are generated in an order compatible with the inclusion relation. We have written a code DYNSA1 which perfoms this task by using $2^n+n^2+5n+cst$ memory locations, each subset $S=\{i_1,i_2,\ldots i_s\}$ being unambignous by represented by its binary expansion $2^{i_1-1}+2^{i_2-1}+\ldots+2^{i_s-1}$ and generated in ascending order of these numbers. While the application of the recurrence relation seems to require $O(n^3 2^n)$ operations at first glance, this number is cut down to $O(n2^n)$ in DYNSA1. Such a performance has been announced by Lawler [17] for the Held and Karp's dynamic programming algorithm applied to the Linear Ordering Problem. Since there is no other precision in Lawler's paper, we give a detailed description of our algorithm along with a proof of the $O(n2^n)$ behavior. The code DYNSA1 has less than 100 FORTRAN statements, including input and output (68 statements perform the optimization) and a copy of it can be obtained from the authors.

It is possible to reduce the execution time of DYNSA1 by about one-half by restricting consideration to subsets with at most $\left\lceil \frac{1}{2} n \right\rceil$ elements, due to the symmetry of the objective function, and by applying the relation

$$f(N) = \min \{ f(S)+f(\bar{S}) \mid S \leqslant N, \ |S| = \left\lceil \frac{n}{2} \right\rceil \}$$

We have modified DYNSA1 into DYNSA2, which uses the same number of memory locations and 143 FORTRAN statements for the optimization. We have run both codes on the CDC Cyber 173 of Université de Montréal, under compile FTN, opt=2 within only 130K of core, on problems with n=10 to 14 (a problem with n=15 requires about 190K). The solutions times are given in Table 1.

The space requirements are the bottleneck for such an algorithm: a run on an 18-room problem would require about 77 seconds but 1110K of memory, i.e. more than one megabyte. We are currently working on an algorithm which would require about $\binom{n+1}{m} + O(n^2)$ memory locations, where $m = \left\lfloor \frac{1}{2}(n-1) \right\rfloor$. Such an algorithm is expected to solve a 16-room problem within 130K and an 18-room problem within 400K. By using auxilliary storage, it might be possible to handle problems with one additional room. However it seems unlikely that a dynamic programming approach would allow exact solution of problems with more than 20 rooms within reasonable time and memory requirements.

Number of rooms	Solution times CDC Cyber 175, sec.	
	DYNSA1	DYNSA2
10	.19	.11
11	.41	.29
12	.87	.51
13	1.85	1.29
14	3.92	2.24

TABLE 1: Solution Times for dynamic programming algorithms.

CONCLUSION

The main results of this paper have shown a remarkable increase in difficulting when generalizing the Optimal Linear Ordering Problem the One-Dimensional Space Allocation Problem. The mere consideration of the lengths of the items to be located has made unapplicable most of the known solution procedure which efficiently solved particular cases of the OLOP. Few exceptions can be found in the paper by Chan and Francis [7]. On the other hand, it has been shown that Held and Karp's dynamic programming method can be extended to the ODSAP, providing a reliable solution procedure for problems with about 15 items. Since most practical applications of the ODSAP model are bound to involve larger problems to be solved, it appears necessary to extend the solution capability beyond the present level. We have identified two ways in that direction. Exact solution of larger ODSAP's could be obtained by branch-and-bound provided that an efficient bounding scheme is devised. We note that the Linear Assignment bound [10], [16], which is basically one of the few available for an exact solution to the Quadratic Assignment Problem cannot even be applied to the ODSAP, since the computation of every entry in the assignment matrix requires the solution of a problem similar to the 2-machines job scheduling problem mentionned in section (a lower bound could be derived from an heuristic solution with performance guarantee, e.g. [27], but we did not investigate this any further). If a satisfactory bounding scheme can be provided it is possible that a branch-and-bound algorithm implementing

it could benefit from using dominance tests such as in [22] and dynamic programming for fast screeming of the lower levels of the enumeration tree. It is still conceivable that a procedure combining branch-and-bound and dynamic programming, such as [19] , performs quite well if efficient storing and addressing techniques are provided for handling the subsets generated by the procedure.

Approximate solutions of rather large ODSAP's can be obtained using either constructive, improvement or hybrid heuristic methods. Several constructive heuristics can be derived from the solution procedures which work for particular cases. An example is an alternating order (see Section 4) using some reasonable estimate (e.g. the geometric means of the rows of the matrix W) for the probabilities used in the case of independent destinations; another example is the Adolphson-Hu procedure applied to a related ODSAP which retains only the maximum weighted tree of the matrix W of the original given ODSAP, with possibly several trial roots. Improvement heuristics in the spirit of Reiter and Sherman [25] , using adjacent pairwise exchanges, insertions, segment insertions or swapping can also be used to improve given solutions. It is expected that hybrid schemes, applying improvement procedures to the solutions generated by constructive methods, produce the best solutions, perhaps at the expense of much computing time. One major issue concerning practical application of heuristics is to determine a trade-off between the value of the solution produced and the computing requirements to obtain it. At the present time this can be resolved by extensive computational comparisons and evaluations. Other

avenues which currently receive more and more attention, concern the performance guarantees and the probabilistic analysis of heuristics, good examples of which being in the studies of a two-dimensional extension of the ODSAP [15] and of a dynamic version of the ODSAP [20].

ACKNOLEDGEMENTS

The authors gratefully acknoledge a motivating correspondence with Professor Britton Harris, University of Pennsylvania and the encouragements from Professor Richard L. Francis, University of Florida, to write this paper. The research of the second author was motivated, five years ago, by Professors Claude Delobel and Michel Sakarovitch, University of Grenoble, and was supported by the National Research Council of Canada, Grant A-4592.

APPENDIX 1: Proof of Theorem 1.

Consider an optimum permutation π which does not verify assumption A3 and let r_j be the first (i.e. leftmost) room in π which is located before (i.e. the left of) some room in the path from r_1 to r_j. Then r_j is located before $r_{p(j)}$. Let r_k be the first room located after r_j, with its predecessor located before r_j; this is well defined since there is at least one room with this property on the path from r_1 to r_j. Then consider the permutation π' deduced from π by inserting r_k just before r_j; that is, if r_j is in position p and r_k in position q ($q > p$) in π, then:

$$\pi'(t) = \begin{cases} \pi(t) & \text{if} \quad t < p \text{ or } t < q \\ \pi(q) = k & \text{if} \quad t = p \\ \pi(t-1) & \text{if} \quad p < t \leqslant q \end{cases}$$

See Figure 1.

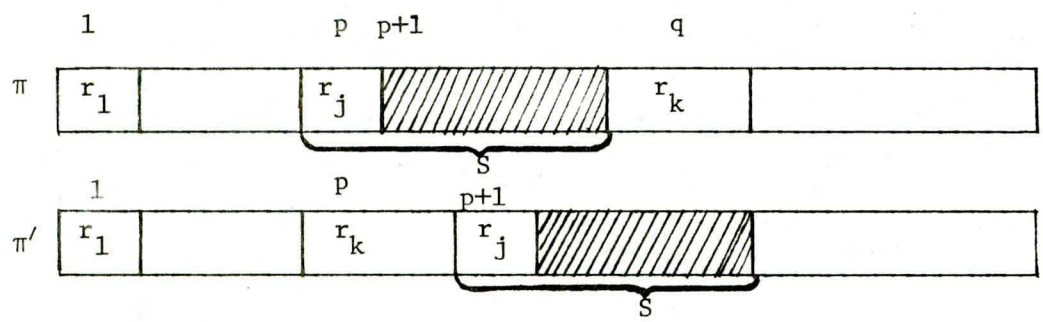

Figure 1: Permutation π and π' used in the proof of Theorem 1.

Denote by S the set of indices of rooms which have been moved to the right, that is $S = \{j, \pi(p+1), \ldots, \pi(q-1)\}$

and L their total length

$$L = \sum_{h \in S} \ell_h = \ell_j + \ell_{\pi(p+1)} + \ldots + \ell_{\pi(q-1)}$$

We evaluate the new center-to-center distances $d^c(p(h), h, \pi')$, as defined by (2-1):

(i) for all fooms r_h in positions $t < p$, $r_{p(h)}$ is before r_h and nothing has been changed, so

$$d^c(p(h),h,\pi') = d^c(p(h),h,\pi)$$

(ii) for the room r_k, $p(k)$ is in a position $t < p$, then

$$d^c(p(k),k,\pi') = d^c(p(k), k,\pi) - L$$

(iii) for the rooms r_h in S with $p(h) \neq k$, $r_{p(h)}$ is located after r_h and has been either moved to the right (if $p(h) \in S$) or not moved at all, so

$$d^c(p(h),h,\pi') \leqslant d^c(p(h),h,\pi)$$

(iv) for the rooms r_h in S with $p(h) = k$

$$d^c(k,h,\pi') \leqslant d^c(k,h,\pi) + L$$

(v) for the rooms r_h in positions $t > q$ with $p(h) \neq k$, $r_{p(h)}$ has been either not moved or moved to the right, so

$$d^c(p(h),h,\pi') \leqslant d^c(p(h),h,\pi)$$

(vi) for the rooms r_h in positions $t > q$ with $p(h) = k$ then

$$d^c(k,h,\pi') = d^c(k,h,\pi) + L$$

Hence $z(\pi') \leqslant z(\pi) - (w_{p(k)k} - \sum_h w_{kh})\, L$

and using (3-1)

$$z(\pi') \leqslant z(\pi)$$

Since π is assumed to be an optimum permutation, we have constructed another optimum permutation π' in which the p first rooms are located after their predecessor. By applying this construction at most n-1 times we obtain an optimum permutation which satisfies assumption A3. #

APPENDIX 2: Proof of Theorem 2.

The proof is similar to the proof of theorem 1, except that we have to take also into account the trips back to r_1. So, consider the permutations π and π' as defined in the proof of theorem 1 and the six cases (i)-(vi):

- in cases (i), (v) and (vi) this distance back to r_1 has not been modified.
- case (ii): r_k and all the rooms to be visited before r_k are in the p first positions, hence the trip back to r_1 has been reduced by L for all patients whose diagnostic is completed in r_k.
- case (iii) and (iv):

  (a) for all rooms r_h in S such that room r_k has not to be visited before r_h (that is $k \neq p(h)$ and $k \neq p(p(h))$ and so on), there is one room, among those to be visited before r_h, which is located in position $t > q$; for all the patients whose diagnostic is completed in r_h, the <u>total</u> distance traveled has not been lengthened.

  (b) for all rooms r_h in S such that room r_k has to be visited before r_h and some room r_g to be visited between r_k and r_h is located in position $t > q$, the total distance traveled by those patients finishing in r_h has not been lengthened.

  (c) for all rooms in S such that room r_k has to be visited before r_h and all rooms to be visited between r_k and r_h are in S, it follows from the definition of r_k that these rooms are located, in π, between r_h and r_k and in reverse order (see Figure 3); let r_f denote the room

which is visited just after r_h and before r_h (it may be r_h itself); it can be verified that the total distance traveled by a patient finishing in r_h has been reduced by the distance between the center of r_f and the rightmost end of S, but also augmented by the half length of r_k. From condition (3-2) it follows that this cannot be an augmentation (the inverse situation in which condition (3-2) is not verified, is pictured in Figure 3).

By summing over all the patients, it follows that π' is not worse than π, and the proof is completed as in theorem 1. #

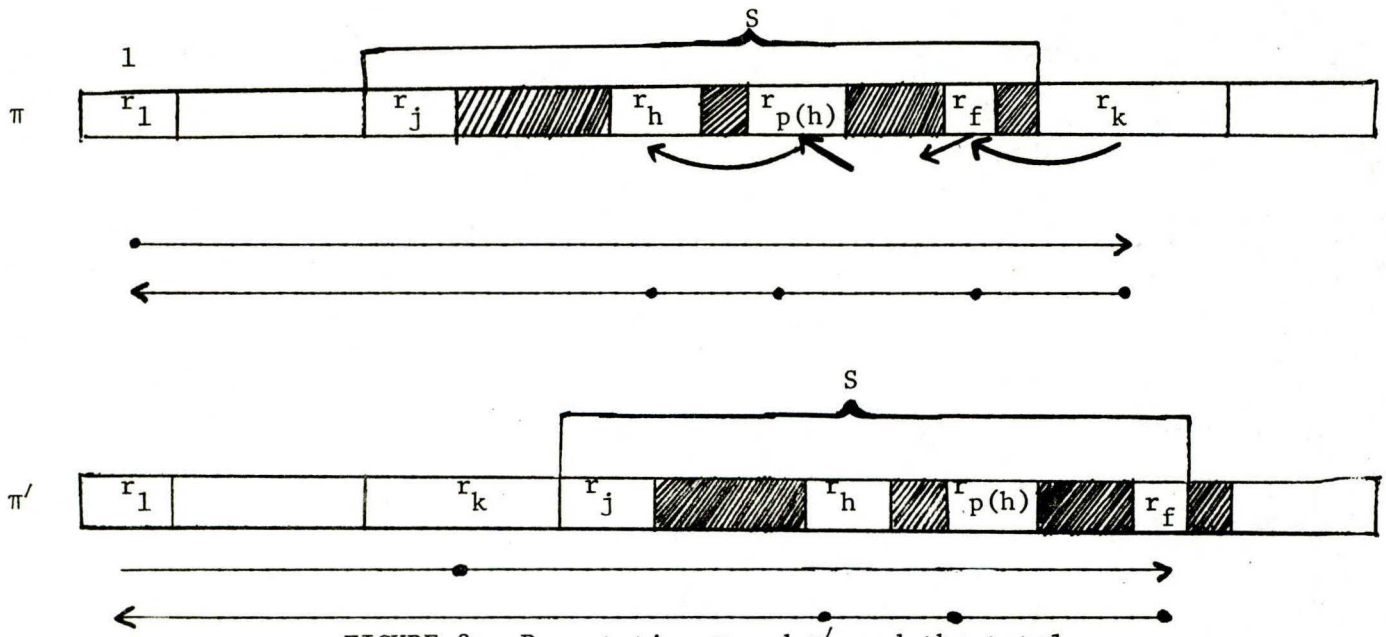

FIGURE 3: Permutation π and π' and the total distance traveled by a patient finishing in r_h, used in the proof of Theorem 2.

APPENDIX 3: Proof of Theorem 5.

Let us first state the ODSAP with independent destinations and arbitrary room lengths (hereafter ODSAPID) in a decision problem format:

Input: positive sequences $(p_1, p_2, \ldots p_n)$ and $(\ell_1, \ell_2, \ldots, \ell_n)$; positive integer W.

Property: there is a permutation π of $\{, 2, \ldots, n\}$ such that

$$\sum_{u=1}^{n-2} \sum_{v=u+2}^{n} p_{\pi^{-1}}(u)\, p_{\pi^{-1}}(v) \sum_{t=u+1}^{v-1} \ell_{\pi^{-1}}(t) \leqslant W$$

In order to prove theorem 5, we note that this problem is in NP and we show that a known NP-complete problem can be reduced to it. Consider the PARTITION problem [13] :

Input: positive sequence $(c_1, c_2, \ldots, c_n)$

Property: there exist a subset $I \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in I} c_i = \sum_{i \notin I} c_i$$

To every instance of this problem, defined by a sequence $(c_1, c_2, \ldots, c_n)$ we associate an instance of ODSAPID with n+1 rooms, defined by ODSAPID1

$p_i = \ell_i = c_i$ for all $i=1, \ldots, n$

Let $N=\{1, 2, \ldots, n\}$

Lemma 1: all permutations π have the same cost for ODSAPID1.

Proof: for a permutation π

$$z(\pi) = \sum_{u=1}^{n-2} \sum_{v=u+2}^{n} c_{\pi-1}(u)\, c_{\pi-1}(v) \sum_{t=u+1}^{v-1} c_{\pi-1}(t)$$

$$= \sum_{\{i,j,k\} \in P_3(N)} c_i\, c_j\, c_k = c_1(N)$$

where $P_k(S)$ is the set of all subsets with k elements of a set S and

$$c_k(S) = \sum_{\{i_1,i_2,\ldots,i_k\}\epsilon P_k(S)} c_{i_1} c_{i_2} \ldots c_{i_k}$$

Then $z(\pi)$ is independent of π #

We now define ODSAPID2 by adding a new room n+1

with $p_{n+1}=2$ and $\ell_{n+1}=1$

Consider a permutation π and let $p=\pi(n+1)$ and

$$I = \{i \mid \pi(i) < p\}$$

$$\bar{I} = \{i \mid \pi(i) > p\}$$

(one of these sets may possibly be empty).

and consider the following decomposition

$$\begin{aligned} z(\pi) = {} & c_3(I) + \sum_{\{i,j\}\epsilon P_2(I)} c_i c_j \left(2 + \sum_{k\epsilon \bar{I}} c_k\right) \\ & + \sum_{i\epsilon I} \sum_{j\epsilon \bar{I}} c_i c_j \\ & + \sum_{(i,j)\epsilon P_2(\bar{I})} c_i c_j \left(2 + \sum_{k\epsilon \bar{I}} c_k\right) \\ & + c_3(\bar{I}) \end{aligned} \qquad \text{(A-1)}$$

The first (last) element represents the interaction within I (resp $\bar{I}$). The other elements represent the remaining interactions, decomposed, as in relation 8, along the length of the rooms in I, of n+1 and of the rooms in $\bar{I}$ respectively.

The use of $c(I)$ and $c(\bar{I})$ in (A-1) is justified by the above lemma 1.

It follows

$$z(\pi) = \sum_{\{i,j,k\}\in P_3(N)} c_i c_j c_k + \sum_{\{i,j\}\in P_2(N)} c_i c_j$$

$$+ \sum_{\{i,j\}\in P_2(I)} c_i c_j + \sum_{\{i,j\}\in P_2(\bar{I})} c_i c_j$$

$$= c_3(N) + 2\, c_2(N) - \sum_{i\in I} \sum_{j\in \bar{I}} c_i c_j$$

Then $z(\pi)$ is minimized when $\sum_{i\in I} \sum_{j\in \bar{I}} c_i c_j$ is maximized, that is for a subset I such that $c_1(I)$ is as close as possible to $\frac{1}{2}\, c_1(N)$. By setting $W = c_3(N) + 2c_2(N) - \frac{1}{4}\,(c_1(N))^2$ it follows that PARTITION has a solution if and only if ODSAPID2 has a solution for this value of W. #

APPENDIX 4: An $O(n\ 2^n)$ version of the dynamic programming algorithm.

A subset $S = \{i_1, i_2, \ldots, i_s\}$ of N will be represented by a simple list $(i_1, i_2, \ldots, i_s)$ and its address is $a = p(i_1) + p(i_2) + \ldots + p(i_s)$ where the numbers $p(i) = 2^{i-1}$ can be precomputed. Hence the address of every subset $S-\{i_k\}$ $(i \in S)$ is simply $a-p(i_k)$.

If we define, for a given subset S

$$\bar{w}(i_k) = \sum_{j \in \bar{S}} w_{ij} \qquad \text{for all } k=1, \ldots, s$$

and $\bar{\bar{w}} = \sum_{k=1}^{s} \bar{w}\ (i_k)$

the recurrence relation could be rewritten as

$$f(a) = \min_{k=1, \ldots, s} \{f(a-p(i_k)) + \ell_{i_k} (\bar{\bar{w}} - \bar{w}(i_k)) : k=1, \ldots, s\}$$

This minimum is computed in $O(s)$ operations, provided the entries $\bar{w}(i_k)$ are available. Hence, in order to prove the $O(n2^n)$ behavior, it suffices to show how the subsets S are generated and the values $\bar{w}(i_k)$ updated in $O(n2^n)$ operations.

The subsets S are generated in DYNSA1 according to the order of their addresses, which is precisely the lexicographic order, by the following algorithm:

<u>Step 0</u> (initialization)

set $f(1) \leftarrow 0$

$a \leftarrow 2$

$k \leftarrow 1$

Step 1 (generate singleton k)

$k \leftarrow k+1$

$i_1 \leftarrow k$

initialize $\bar{w}(k)$

$s \leftarrow 1$

set $f(a) \leftarrow 0$

$k \leftarrow 1$

$a \leftarrow a+1$

Step 2 (add the element k to the current subset)

$s \leftarrow s+1$

$i_s \leftarrow k$

compute $\bar{w}(k)$, update $\bar{w}(j)$ for $j \in S$ $(j \neq k)$ and compute $\bar{\bar{w}}$

compute f(a) using recursion formula

Step 3 (termination test)

if $(a=2^n-1)$ go to step 7

$a \leftarrow a+1$

$k \leftarrow 1$

Step 4 (is k the last element in S?)

if $(i_s \neq k)$ go to step 2

Step 5 (delete the last element in S)

if $(s=1)$ go to step 1

$s \leftarrow s-1$

$k \leftarrow k+1$

go to step 4.

In order to justify this algorithm we first assume that the elements i_1, i_2, ..., i_s of S are ordered such that $i_1 > i_2 > \ldots > i_s$.
If the last element i_s is not 1, then the next subset in the lexicographic order is simply obtained by adding 1 to S. In the other case, we "delete" this last element $i_s=1$ (that is we move backward this pointer s) and we check if the now last element is 2. If it is not, then the next subset is obtained by adding 2, otherwise we keep going backward in S while incrementing k, and still comparing $i_{s'}=1$ (s' is the decremented value of the pointers). At the first occurence of a negative answer we know that the last elements of S were

$$\begin{array}{ll} i_{s'+1} & = k-1 \\ i_{s'+2} & = k-2 \\ \ldots\ldots\ldots\ldots & \\ i_s & = 1 \end{array}$$

and the next subset is obtained by setting $i_{s'+1} = k$ in step 2. This also proves inductively that the elements of the subsets are listed in decreasing order of their indices. On the other end, if S is exhausted before a negative answer occurs, then we had $S = \{s, s-1,\ldots,1\}$ and the next subset is the singleton $\{s+1\}$. For our purposes it is clear that this generation process requires at most $O(s)$ operations per subset, hence is at most $O(n2^n)$. However, it is not difficult to show that it is actually an $O(2^n)$ algorithm.

It remains to show how the computation of $\bar{w}$ (k) is performed in $O(s)$ operations within each step. Before applying the algorithm, we compute

$$w_o(i) = \sum_{j \neq i} w(i,j) \quad \text{for all } i=1, \ldots, n.$$

and

$$w_1(j,i) = \sum_{k=1}^{j-1} w(k,i) - w(j,i) \quad \text{for all } i=2, \ldots, n \text{ and all } j=1, \ldots, i-1$$

In step 1, for $S=\{k\}$, clearly $\bar{w}(k) = w_o(k)$. In step 2, assume the entries $\bar{w}(i)$ have been computed for the set S, and the next set S' is obtained by deleting the last (k-1) elements of S and adding k in position s ($s' \geqslant 2$) then the new values $\bar{w}'(i)$ can be obtained as

$$\bar{w}'(k) = w_o(k) - \sum_{j=1}^{s'-1} w(k, i_j)$$

$$\text{and} \quad \bar{w}'(i_j) = \bar{w}(i_j) + \sum_{u=1}^{k-1} w(u,i_j) - w(k,i_j)$$

$$= \bar{w}(i_j) + w_1(k,i_j).$$

then all the $\bar{w}'(i)$ can be updated in $O(s')$ operations.

In order to preserve space, note that w_1 uses the lower triangular part of an array, and that, since w is assumed to be symmetric, it may use the upper triangular part of the same array. Note also that we do not record which subset defines the minimum in the recurrence relation, in order to save one 2^n array. Once f(N) has been computed, the optimal solution can be retrieved by an $O(n^2)$ backtracking classical in dynamic programming.

The code DYNSA2 differs from DYNSA1 in that additional tests are introduced to avoid generating subsets with more than $\lceil \frac{1}{2} n \rceil$ elements. The relation defining $\bar{w}'$ from $\bar{w}$ are accordingly modified in these cases. The optimum value f(N) is obtained by the relation $f(N)=\min\{f(S)+f(\bar{S}) : |S| = \lceil \frac{1}{2} n \rceil\}$ and two backtracking steps are necessary to construct the whole optimal solution.

Listings of codes DYNSA1 and DYNSA2 are available from the authors upon request .

REFERENCES

1. D.L. ADOLPHSON, "Single Machine Job Sequencing with Precedence Constraints", SIAM J. Comput. 6, 40-54 (1977).

2. D. ADOLPHSON and T.C. HU, "Optimal Linear Ordering", SIAM J. Appl. Math. 25, 403-423 (1973).

3. K.R. BAKER, "Introduction to Sequencing and Scheduling", Wiley, New York, N.Y., 1974.

4. P.P. BERGMANS, "Minimizing Expected Travel Time on Geometrical Patterns by Optimal Probability Rearrangements", Information and Control. 20, 331-350 (1972).

5. P. BRUCKER, J.K. LENSTRA and A.H.G. RINNOOY KAN, "Complexity of Machine Scheduling Problems", Report R75-15, Graduate School of Management, Delft, The Netherlands (1975).

6. J. BRUNO, E.G. COFFMAN Jr. and R. SEHTI, "Scheduling Independent Tasks to Reduce Mean Finishing Time", Comm. ACM 17, 382-387 (1974).

7. A.W. CHAN and R.L. FRANCIS, "Some Layout Problems on the Line with Minimum-Separation Constraints", to appear in Opns. Res.

8. M.R. GAREY, "Optimal Task Sequencing with Precedence Constraints", Discrete Math. 4, 37-56 (1973).

9. M.R. GAREY, D.S. JOHNSON and L. STOCKMEYER, "Some Simplified NP-Complete Graph Problems", Theoretical Comp. Sc. 1, 237-267 (1976).

10. P.C. GILMORE, "Optimal and Sub-optimal Algorithms for the Quadratic Assignment Problem", SIAM J. Appl. Math. 10, 305-313 (1962).

11. D.D. GROSSMAN and H.F. SILVERMAN, "Placement of Records on a Secondary Storage Device to Minimize Access Time," J. ACM. 20, 429-438 (1973).

12. W.A. HORN, "Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties", SIAM J. Appl. Math., 23, 189-202 (1972).

13. R.M. KARP, "Reducibility Among Combinatorial Problems ", Complexity of Computer Computation, (ed. by R. E. Miller and J.W. Thatcher), Plenum Press, New York, N.Y., 85-103, 1972.

14. R.M. KARP and M. HELD, "Finite-State Processes and Dynamic Programming", SIAM J. Appl. Math., 15, 693-718 (1967).

15. R.M. KARP, A.C. MCKELLAR, and C.K. WONG, "Near-Optimal Solutions to a 2-Dimensional Placement Problem", SIAM J. Comp., 4 No. 3, 271-286 (1975).

16. E.L. LAWLER, "The Quadratic Assignment Problem", Management Sc., 9, 586-599 (1963).

17. E.L. LAWLER, "The Quadratic Assignment Problem: A Brief Review" in Combinatorial Programming: Methods and Applications (ed. by B. Roy), D. Reidel Publishing Co., Boston, Mass., 351-360, 1975.

18. R.L. LOVE and J.Y. WONG, "On Solving a One-Dimensional Space Allocation Problem with Integer Programming", INFOR 14, 139-143 (1976).

19. R.E. MARSTEN and T.L. MORIN, "A Hybrid Approach to Discrete Mathematical Programming", Math. Programming 14, 21-40 (1978).

20. A.C. MCKELLAR and C.K. WONG, "Dynamic Placement of Records in Linear Storage", J. ACM 25, 421-434 (1978).

21. P.M. MORSE, "Optimal Linear Ordering of Information Items", Opns Res. 20, 741-751 (1972).

22. J.C. PICARD and M. QUEYRANNE, "The Time-Dependent Traveling Salesman Problem and Application to the Tardiness Problem in One-Machine Scheduling", Operations Research 26, 86-110 (1978).

23. V.R. PRATT, "An NlogN Algorithm to Distribute N Records in a Sequential Access File", Complexity of Computer Computation, (ed. by R.E. Miller, and J.W Thatcher), Plenum Press, New York, N.Y., 111-118, 1972.

24. C.V. RAMAMOORTHY and P.R. BLEVINS, "Arranging Frequency Dependent Data on Sequential Memories", Proc. AFIPS 1971 SJCC 38, 545-556 (1971).

25. S. REITER and G. SHERMAN, "Discrete Optimizing", Journal SIAM 13, 864-869 (1965).

26. D. ROMERO, "Variations sur l'effet Condorcet", Thèse Doct. 3e Cycle, Université de Grenoble, France (1978).

27. S.K. SAHNI, "Algorithms for Scheduling Independent Tasks", J. ACM 23, 116-127 (1976).

28. D.M. SIMMONS, "One-Dimensional Space Allocation: An Ordering Algorithm", Opns. Res. 17, 812-826 (1969).

29. D.M. SIMMONS, "A Further Note on One-Dimensional Space Allocation", Opns. Res. 19, 249-240 (1971).

A CONSULTER
SUR PLACE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00288957 2