Opl Res. Q., Pergamon Press 1975. Vol. 26, 4, i, pp. 717 to 733. Printed in Great Britain

Some Simple Applications of the Travelling Salesman Problem

J. K. LENSTRA
Mathematisch Centrum, Amsterdam
and
A. H. G. RINNOOY KAN
Graduate School of Management, Delft

The travelling salesman problem arises in many different contexts. In this paper we report on typical applications in computer wiring, vehicle routing, clustering and job-shop scheduling. The formulation as a travelling salesman problem is essentially the simplest way to solve these problems. Most applications originated from real world problems and thus seem to be of particular interest. Illustrated examples are provided with each application.

THE TRAVELLING SALESMAN PROBLEM

Introduction

IN THIS paper we discuss four apparently unrelated problems that arise in the context of *computer wiring*, *vehicle routing*, *clustering a data array* and *job-shop scheduling with no intermediate storage*. It turns out that each of these problems can be formulated as a travelling salesman problem (TSP). Three of them originated from real world situations and were not immediately recognized as TSPs; use of TSP algorithms led to better solutions, as will be illustrated below.

Moreover, not only are the four problems special cases of the TSP, but the TSP can conversely be interpreted as a special case of any of these problems. Formulation as a TSP thus is essentially the simplest way to solve them. For the last two problems, their complete equivalence to the TSP is non-trivial.[1,2]

After introducing the TSP and reviewing the methods for its solution, we will deal with the applications in the remaining sections. Each of the four problems is first described verbally, then formulated as a TSP and finally illustrated by some examples.

Formulation of the TSP

A salesman wishes to find the shortest route through a number of cities and back home again. This problem is known as the *travelling salesman problem* and can be stated more formally as follows.

Given a finite set of cities N and a distance matrix (c_{ij}) $(i, j \in N)$, determine

$$\min_\pi \sum_{i \in N} c_{i\pi(i)},$$

where π runs over all *cyclic permutations* of N; $\pi^k(i)$ is the kth city reached by the salesman from city i. If $N = \{1, \ldots, n\}$, then an equivalent formulation is

$$\min_\nu \left(\sum_{i=1}^{i=n-1} c_{\nu(i)\nu(i+1)} + c_{\nu(n)\nu(1)} \right),$$

where ν runs over all *permutations* of N; here $\nu(k)$ is the kth city in a saleman's tour. If G denotes the complete directed graph on the vertex set N with a weight c_{ij} for each arc (i, j), then an optimal tour corresponds to a *hamiltonian circuit* on G (i.e. a circuit passing through each vertex exactly once) of minimum total weight.

If $c_{ij} = c_{ji}$ for all (i, j), the problem is called *symmetric*, otherwise it is called *asymmetric*. If $c_{ik} \leqslant c_{ij} + c_{jk}$ for all (i, j, k), the problem is called *euclidean*.

Solution methods

References 3, 4 and 5 contain recent surveys of known solution methods.

We can distinguish between *optimal* and *suboptimal* algorithms. The first type of algorithm produces solutions that are *guaranteed to be optimal* but may require inordinate running times; of special interest are the branch-and-bound methods developed by Little *et al.*,[6] Held and Karp[7–9] and Bellmore and Malone.[10] Suboptimal algorithms which produce *approximate* solutions in reasonable times include Lin,[11] Christofides and Eilon[12] and Lin and Kernighan.[13]

In fact, we shall be using the following algorithms:

(a) a branch-and-bound procedure based on Little *et al.*[6], incorporating an improved branching strategy that allows early pruning of a branch through sufficiently large penalties;

(b) a branch-and-bound procedure based on Held and Karp[8] for symmetric TSPs;

(c) a heuristic procedure for generating 3-*optimal* tours for symmetric TSPs, following the enumeration scheme given by Lin[11] with deletion of some superfluous checks for improvement.

Descriptions of these algorithms as well as computational experience and ALGOL 60 procedures can be found in Lenstra.[14]

COMPUTER WIRING

Problem description

The following problem arises frequently during the design of computer interfaces at the Institute for Nuclear Physical Research in Amsterdam.

An interface consists of a number of modules, and on each module several pins are located. The position of each module has been determined in advance. A given subset of pins has to be interconnected by wires. In view of possible future changes or corrections and of the small size of the pin, at most two wires are to be attached to any pin. In order to avoid signal cross-talk and to improve ease and neatness of wirability, the total wire length has to be minimized.

TSP formulation

Let P denote the set of pins to be interconnected, c_{ij} the distance between pin i and pin j and H the complete graph on the vertex set P with weights c_{ij} on the arcs.

If any number of wires could be attached to a pin, an optimal wiring would correspond to a minimum *spanning tree* on H, which can be found efficiently by the algorithms of Kruskal[15] or Prim[16] and Dijkstra.[17] However, the degree requirement implies that we have to find a minimum *hamiltonian path* on H (i.e. a path passing through each vertex exactly once). This problem corresponds to finding a minimum *hamiltonian circuit* on G with $N = P \cup \{*\}$ and $c_{i*} = c_{*i} = 0$ for all $i \in N$. The wiring problem can thus be converted into a symmetric euclidean TSP.

A more difficult problem occurs if the positions of the modules have not been fixed in advance but can be chosen so as to minimize the total wire length for all subsets of pins that have to be interconnected. A review of this *placement problem* and the associated *quadratic assignment problem* is given by Hanan and Kurtzberg.[18]

Results

The procedure that was used originally produced clearly non-optimal wiring schemes like the example with two subsets of pins in Figure 1(a). The size and number of the problems was such that Lin's heuristic had to be used. The 3-optimal results on the example are given in Figure 1(b) (see Visschers and Ten Kate[19]).

VEHICLE ROUTING

Problem description

In 28 towns in the Dutch province of North-Holland telephone boxes have been installed by the national postal service (PTT). A technical crew has to visit each telephone box once or twice a week to empty the coin box and, if necessary, to replace directories and perform minor repairs. Each working day of at most 445 min begins and ends in the provincial capital Haarlem. The problem is to minimize the number of days in which all telephone boxes can be visited and the total travelling time.

A similar problem arose in the city of Utrecht. Here about 200 mail boxes have to be emptied each day within a period of one hour by trucks operating from the central railway station. The problem is to find the minimum number of trucks able to do this and the associated minimum travelling time.

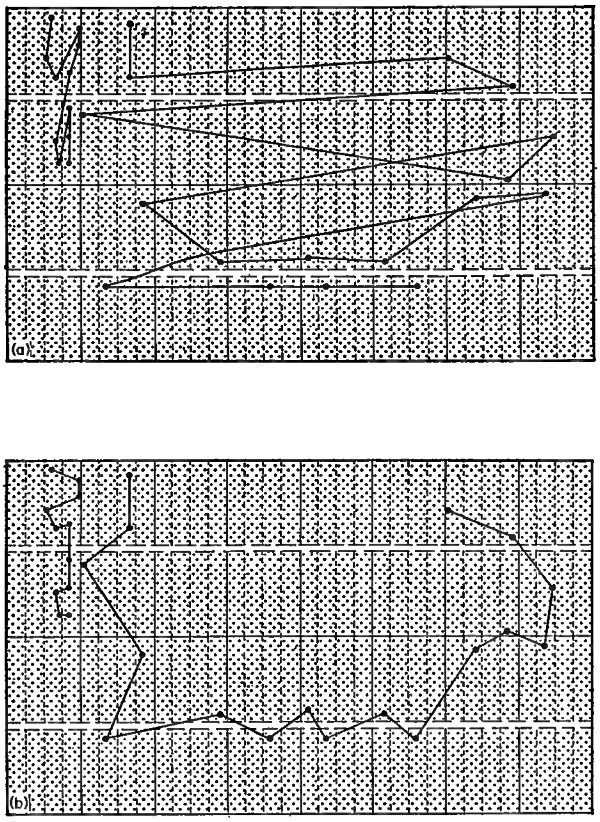


FIG. 1. (a) *Wiring without optimization.* (b) *3-Optimal wiring.*

TSP formulation

Both problems are types of classical *vehicle routing problems* (VRP) (see Eilon *et al.*[5]). They will be denoted by $P1$ and $P2$, respectively, and can be characterized more formally as follows:

n cities i $(1 \leqslant i \leqslant n)$ (the *customers*) are to be visited [$P1$: 28 towns; $P2$: 200 mail boxes] by m *vehicles* [$P1$: m working days; $P2$: m trucks] operating from city $*$ (the *depot*) [$P1$: Haarlem; $P2$: central railway station];

the travelling time between cities i and j is $d_{ij} = d_{ji}$ minutes, for

$$i, j \in \{1, \ldots, n\} \cup \{*\};$$

the time to be spent in city i is e_i minutes, for $i \in \{1, \ldots, n\}$ [$P1$: $8 \times$ number of telephone boxes in town i; $P2$: 1];

the maximum allowable time for any vehicle to complete its route is f minutes [$P1$: 445; $P2$: 60];

there may be additional constraints [$P1$: one town (nr. 28, Den Helder) has to be visited twice on different days];
criteria by which solutions are judged are:
A, the number of vehicles used;
$B(A)$, the total time used for A vehicles.

If a city has to be visited twice, it is duplicated, appropriate travelling and visiting times are added, and n is increased by one.

[$P1$: Den Helder is split up into two cities 28 and 29; $d_{28\,29}:=\infty$; $n:=29$.]

We replace the depot (city $*$) by m artificial depots (cities $n+1,\ldots,n+m$) and extend the definition of (d_{ij}) and (e_i) as follows (cf. Figure 2):

$$\begin{aligned} d_{i,n+l} &= d_{i*} && \text{for } 1 \leqslant l \leqslant m; \\ d_{n+k,j} &= d_{*j} && \text{for } 1 \leqslant k \leqslant m; \\ d_{n+k,n+l} &= \lambda && \text{for } 1 \leqslant k, l \leqslant m; \\ e_{n+k} &= 0 && \text{for } 1 \leqslant k \leqslant m. \end{aligned}$$

$$\begin{array}{c|ccccc|ccc|} & 1 & \ldots & j & \ldots & n & n+1 & \ldots & n+m \\ \hline 1 & d_{11} & \ldots & d_{1j} & \ldots & d_{1n} & d_{1*} & \ldots & d_{1*} \\ \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ i & d_{i1} & \ldots & d_{ij} & \ldots & d_{in} & d_{i*} & \ldots & d_{i*} \\ \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ n & d_{n1} & \ldots & d_{nj} & \ldots & d_{nn} & d_{n*} & \ldots & d_{n*} \\ \hline n+1 & d_{*1} & \ldots & d_{*j} & \ldots & d_{*n} & \lambda & \ldots & \lambda \\ \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ n+m & d_{*1} & \ldots & d_{*j} & \ldots & d_{*n} & \lambda & \ldots & \lambda \\ \hline \end{array}$$

FIG. 2. *The matrix (d_{ij}).*

We obtain a symmetric euclidean TSP by defining $N=\{1,\ldots,n+m\}$ and $c_{ij}=\frac{1}{2}e_i+d_{ij}+\frac{1}{2}e_j$ for all $i,j\in N$. A salesman's tour is feasible for the VRP provided that the time constraint for each vehicle and possible additional constraints are respected. If a TSP solution contains $m-A$ links between artificial depots, then the corresponding VRP solution uses only A vehicles. Adding another vehicle decreases the number of links between artificial depots by one and hence decreases the objective function by λ. Thus, $-\lambda$ may be interpreted as the cost of a vehicle. We may now consider three possible choices of λ:

$\lambda=+\infty$ will lead to $\min_\pi B(m)$, i.e. the minimum total time for m vehicles[5];
$\lambda=0$ will lead to $\min_\pi\{B(A)\,|\,1\leqslant A\leqslant m\}$, i.e. the minimum total time for any number of vehicles[5];
$\lambda=-\infty$ will lead to $\min_\pi B(\min\{A\,|\,1\leqslant A\leqslant m\})$, i.e. the minimum total time for the minimum number of vehicles.

The latter objective is the criterion function for both $P1$ and $P2$.

An appropriate method for obtaining good VRP solutions is the following:

(1) Choose an initial tour which satisfies the VRP constraints.

(2) Apply an iterative procedure for improving the tour and check the constraints whenever a possible decrease in tour length occurs.

An interesting variation on this type of problem arises in the context of money collection at post offices. For security reasons, several good routes have to be available. The problem is then equivalent to the *moonlighting salesman problem*[20] where k disjoint hamiltonian circuits of minimum total weight are sought. No algorithms for this problem have been proposed so far.

Results

Figures 3 and 4 illustrate some results, obtained for $P1$ and $P2$. In both figures, the links with the depot (*) have not been drawn.

For $P1$, Lin's heuristic method was used. All 3-optimal solutions obtained require 4 days, representing a 50 per cent decrease with respect to the schedule that was previously used. An example is given in Figure 3(a). Exchanging three links in this solution resulted in the schedule given in Figure 3(b); it involves only 3 days, including however one of $449\frac{1}{2}$ min. Computational experience revealed that the heuristic procedure converged much faster with $\lambda = -\infty$ than with $\lambda = 0$ (see Kuiper[21]).

For $P2$, a variation on Lin's method was used, whereby only a limited number of promising potential improvements was checked. The number of trucks needed was reduced from ten (Figure 4(a)) to eight (Figures 4(b–d)). In view of the size of the problem, both possibilities $\lambda = 0$ and $\lambda = -\infty$ have been run only once; the convergence with $\lambda = -\infty$ was relatively slow.

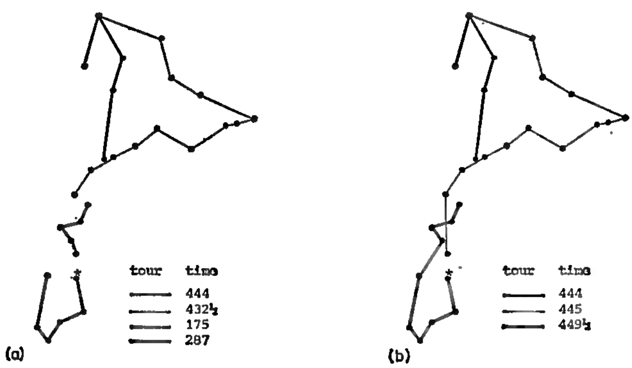


FIG. 3. (a) $P1$: *3-optimal solution*; $\lambda = -\infty$; $B(4) = 1338\frac{1}{2}$. (b) $P1$: *infeasible solution, obtained by hand from Fig.* 3(a); $B(3) = 1338\frac{1}{2}$.

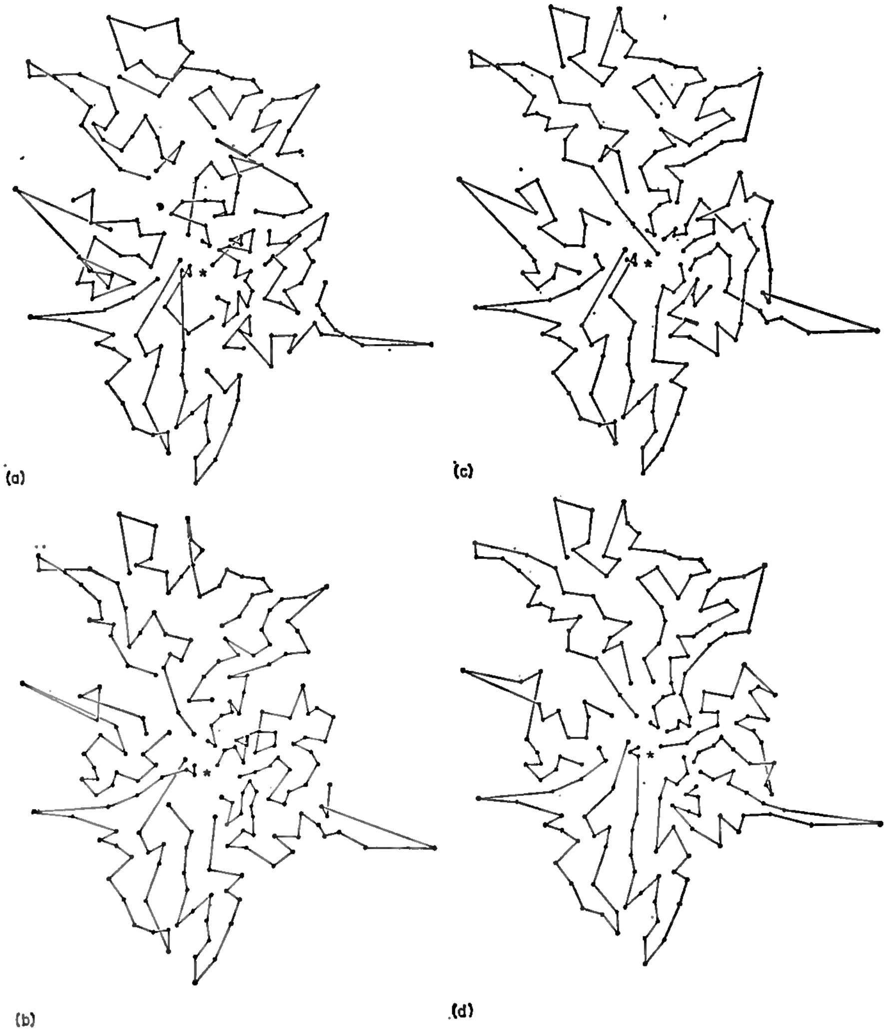


FIG. 4. (a) *P2: previously used solution*; $B(10) = 442$. (b) *P2: locally optimal solution, starting from Fig.* 4(a); $\lambda = 0$; $B(8) = 404$. (c) *P2: locally optimal solution, starting from Fig.* 4(a); $\lambda = -\infty$; $B(8) = 405$. (d) *P2: locally optimal solution, starting from an improvement by hand on Fig.* 4(c); $\lambda = -\infty$; $B(8) = 398$.

CLUSTERING A DATA ARRAY

Problem description

Suppose that a *data array* (a_{ij}) $(i \in R, j \in S)$ is given, where a_{ij} measures the strength of the relationship between elements $i \in R$ and $j \in S$. A *clustering* of the array is obtained by permuting its rows and columns and should identify subsets of R that are strongly related to subsets of S.

This situation occurs in widely different contexts. Here we will apply a clustering technique to three examples. In the first one[22] R is a collection of 24 *marketing* techniques, S is a collection of 17 marketing applications, $a_{ij} = 1$ if technique i has been successfully used for application j, and $a_{ij} = 0$ otherwise. The second example[22] arises in *airport design*; $R(=S)$ is a set of 27 control variables and a_{ij} measures their interdependence. The third example[23] deals with an *import–export matrix*; $R(=S)$ is a set of 50 regions on the Indonesian islands, $a_{ij} = 1$ if in 1971 a quantity of at least 50 tons of rice was transported from region i to region j, and $a_{ij} = 0$ otherwise.

These three examples indicate that the approach is useful for *problem decomposition* and *data reorganization* (see McCormick *et al.*[22]).

To convert this problem into an optimization problem, some criterion has to be defined. The proposed *measure of effectiveness* (ME) is the sum of all products of horizontally or vertically adjacent elements in the array.[22] Figure 5 shows how this criterion relates to various permutations of a 4×4 array. The problem is to find permutations of rows and columns of (a_{ij}) maximizing ME.

$$
\begin{array}{c}
\begin{array}{c|cccc|}
 & 1 & 2 & 3 & 4 \\ \hline
1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 1 \\
3 & 1 & 0 & 1 & 0 \\
4 & 0 & 1 & 0 & 1 \\ \hline
\end{array} \\ \text{ME} = 0
\end{array}
\quad
\begin{array}{c}
\begin{array}{c|cccc|}
 & 1 & 2 & 3 & 4 \\ \hline
1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 \\
3 & 1 & 0 & 1 & 0 \\ \hline
\end{array} \\ \text{ME} = 2
\end{array}
\quad
\begin{array}{c}
\begin{array}{c|cccc|}
 & 1 & 2 & 3 & 4 \\ \hline
1 & 1 & 0 & 1 & 0 \\
3 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 \\ \hline
\end{array} \\ \text{ME} = 4
\end{array}
\quad
\begin{array}{c}
\begin{array}{c|cccc|}
 & 1 & 3 & 2 & 4 \\ \hline
1 & 1 & 1 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 \\
4 & 0 & 0 & 1 & 1 \\
3 & 1 & 1 & 0 & 0 \\ \hline
\end{array} \\ \text{ME} = 6
\end{array}
\quad
\begin{array}{c}
\begin{array}{c|cccc|}
 & 1 & 3 & 2 & 4 \\ \hline
1 & 1 & 1 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 \\
4 & 0 & 0 & 1 & 1 \\ \hline
\end{array} \\ \text{ME} = 8
\end{array}
$$

FIG. 5. ME *for various permutations of a* 4×4 *array.*

TSP formulation

Let $R = \{1, \ldots, r\}$ and $S = \{1, \ldots, s\}$. With the conventions

$$\rho(0) = \rho(r+1) = \sigma(0) = \sigma(s+1) = *,$$
$$a_{i*} = a_{*j} = 0 \quad \text{for } i \in R, \quad j \in S,$$

the ME, corresponding to permutations ρ of R and σ of S, is given by

$$\begin{aligned}
\text{ME}(\rho, \sigma) &= \tfrac{1}{2} \sum_{i \in R} \sum_{j \in S} a_{\rho(i)\sigma(j)}(a_{\rho(i)\sigma(j-1)} + a_{\rho(i)\sigma(j+1)} + a_{\rho(i-1)\sigma(j)} + a_{\rho(i+1)\sigma(j)}) \\
&= \sum_{j=0}^{j=s} \sum_{i \in R} a_{i\sigma(j)} a_{i\sigma(j+1)} + \sum_{i=0}^{i=r} \sum_{j \in S} a_{\rho(i)j} a_{\rho(i+1)j} \\
&= \text{ME}(\sigma) + \text{ME}(\rho),
\end{aligned}$$

so $ME(\rho, \sigma)$ decomposes into two parts, and its maximization reduces to two separate and similar optimizations, one of $ME(\sigma)$ for the columns and the other of $ME(\rho)$ for the rows. McCormick *et al.*[22] state that both subproblems may be rewritten as *quadratic assignment problems*. More precisely, they are symmetric TSPs:

$$\text{TSP}^{\text{col}}: \quad N^{\text{col}} = S \cup \{*\}, \quad c_{jk}^{\text{col}} = -\sum_{i \in R} a_{ij} a_{ik} \quad \text{for } j, k \in N^{\text{col}},$$

$$\text{TSP}^{\text{row}}: \quad N^{\text{row}} = R \cup \{*\}, \quad c_{hi}^{\text{row}} = -\sum_{j \in S} a_{hj} a_{ij} \quad \text{for } h, i \in N^{\text{row}},$$

for $ME(\sigma)$ and $ME(\rho)$, respectively[24]. In general, the clustering problem for a p-dimensional array can be stated as p TSPs. It may be attacked by any algorithm for the TSP; in fact, McCormick's[22] *bond energy algorithm* (BEA) is a simple suboptimal TSP method which constructs a tour by successively inserting the cities[25].

If the data array is symmetric (i.e. $a_{ij} = a_{ji}$ for all i, j), then TSP^{row} and TSP^{col} are identical and only one optimization needs to be performed (see the airport example).

If the data array is square (i.e. $r = s$) but not necessarily symmetric and we want to have equal permutations of rows and columns (i.e. $\rho = \sigma$), then one symmetric TSP results:

$$\text{TSP}^{\text{cow}}: \quad N^{\text{cow}} = N^{\text{col}} = N^{\text{row}}, \quad c_{ij}^{\text{cow}} = c_{ij}^{\text{col}} + c_{ij}^{\text{row}} \quad \text{for } i, j \in N^{\text{cow}}$$

(see the import–export example).

The size of the TSPs might be reduced by assigning identical rows or columns to one single city under the assumption that these rows or columns will be adjacent in at least one optimal solution. This assumption is justified under the conditions expressed by the following theorem.

Theorem

If $a_{ij} \in \{0, 1\}$ for all $i \in R$, $j \in S$, and $c_{kk}^{\text{row}} = c_{kl}^{\text{row}} = c_{ll}^{\text{row}}$ for some $k, l \in N^{\text{row}}$, then row k and row l are identical, and adjacent in at least one optimal solution to TSP^{row}.

Proof: We define $S_i = \{j \mid j \in S, a_{ij} = 1\}$ for all $i \in N^{\text{row}}$. Since $a_{ij} \in \{0, 1\}$ for all $i \in R$, $j \in S$, we have

$$c_{ij}^{\text{row}} = -|S_i \cap S_j| \quad \text{for all } i, j \in N^{\text{row}}, \tag{1}$$

and $c_{kk}^{\text{row}} = c_{kl}^{\text{row}} = c_{ll}^{\text{row}}$ implies that $S_k = S_k \cap S_l = S_l$. Hence row k and row l are identical:

$$a_{kj} = a_{lj} \quad \text{for all } j \in S. \tag{2}$$

Now consider any permutation ρ of R with $\rho(p) = k$, $\rho(q) = l$, $|p - q| > 1$. Insert l between k and $\rho(p+1)$. This will not decrease $ME(\rho)$ if

$$c_{k\rho(p+1)}^{\text{row}} + c_{\rho(q-1)l}^{\text{row}} + c_{l\rho(q+1)}^{\text{row}} \geqslant c_{kl}^{\text{row}} + c_{l\rho(p+1)}^{\text{row}} + c_{\rho(q-1)\rho(q+1)}^{\text{row}}.$$

By (1) and (2), this is equivalent to

$$|S_{\rho(q-1)} \cap S_l| + |S_l \cap S_{\rho(q+1)}| \leqslant |S_l| + |S_{\rho(q-1)} \cap S_{\rho(q+1)}|,$$

which is true, since

$$\begin{aligned} &|S_{\rho(q-1)} \cap S_l| + |S_l \cap S_{\rho(q+1)}| \\ &\qquad = |S_l \cap (S_{\rho(q-1)} \cup S_{\rho(q+1)})| + |S_l \cap S_{\rho(q-1)} \cap S_{\rho(q+1)}| \\ &\qquad \leqslant |S_l| + |S_{\rho(q-1)} \cap S_{\rho(q+1)}|. \end{aligned}$$

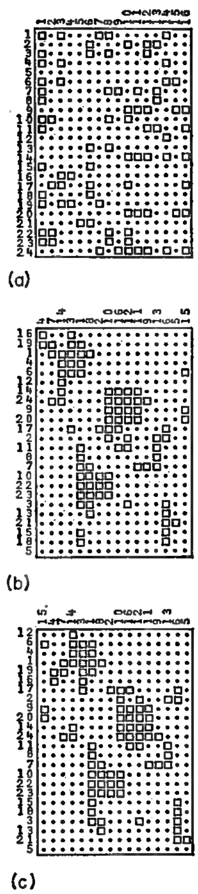


FIG. 6

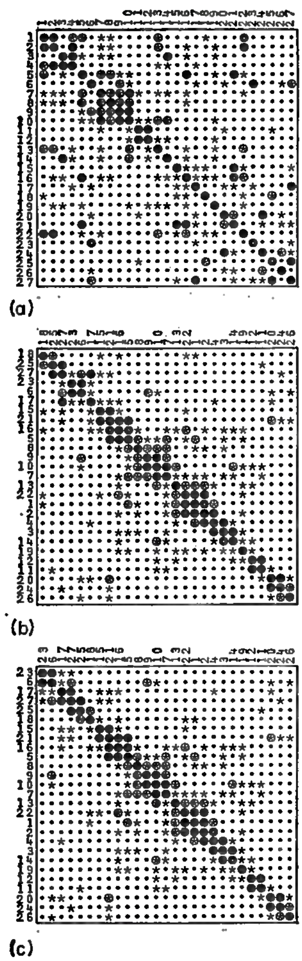


FIG. 7

FIG. 6. *Marketing example*; $\cdot = 0$, $\square = 1$. (a) *Initial array*; ME = 39. (b) BEA *clustering*; ME = 97. (c) *Optimal clustering*; ME = 97.

FIG. 7. *Airport example*; $\cdot = 0$, $* = 1$, $\otimes = 2$, $\bullet = 3$. (a) *Initial array*; ME = 592. (b) BEA *clustering*; ME = 1154. (c) *Optimal clustering*; ME = 1160.

Analogous theorems hold for TSP$^{\text{col}}$ and TSP$^{\text{cow}}$. Definining $R_j = \{i \mid i \in R, a_{ij} = 1\}$ for all $j \in N^{\text{col}}$, we have in the latter case

$$c_{ij}^{\text{cow}} = -|S_i \cap S_j| - |R_i \cap R_j| \quad \text{for all } i, j \in N^{\text{cow}}, \tag{3}$$

and we have to show that

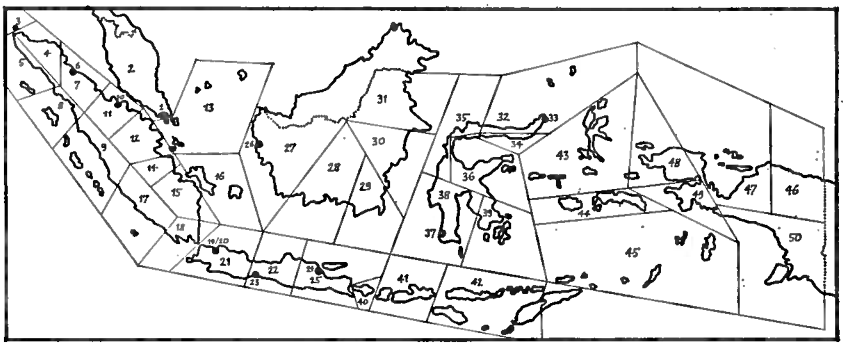
$$\begin{aligned} a_{kj} &= a_{lj} \quad \text{for all } j \in S, \\ a_{ik} &= a_{il} \quad \text{for all } i \in R. \end{aligned} \tag{4}$$

It follows from (3) and $c_{kk}^{\text{cow}} = c_{kl}^{\text{cow}} = c_{ll}^{\text{cow}}$ that $|S_k| + |R_k| = |S_k \cap S_l| + |R_k \cap R_l| = |S_l| + |R_l|$. If $|S_k| > |S_k \cap S_l|$, then $|R_k| < |R_k \cap R_l|$, which is impossible; hence $|S_k| = |S_k \cap S_l| = |S_l|$ and $|R_k| = |R_k \cap R_l| = |R_l|$, which trivially leads to (4).

These results cannot be generalized to cover the case where a_{ij} can take on other values than 0 or 1. For example, if $R = \{1, 2, 3\}$ and $a_{1j} = a_{2j} = 1$, $a_{3j} = 2$ for $j \in S$, then the identical rows 1 and 2 are separated by row 3 in the optimal solution.

Results

The techniques and applications pertaining to the marketing example can be found in McCormick *et al.*[22] Figure 6 shows the initial data array, the clustering produced by the BEA[22] and a clustering corresponding to optimal solutions of TSP$^{\text{col}}$ and TSP$^{\text{row}}$, found by Little's algorithm after application of the theorem on row identification. It turns out that the BEA clustering is optimal.

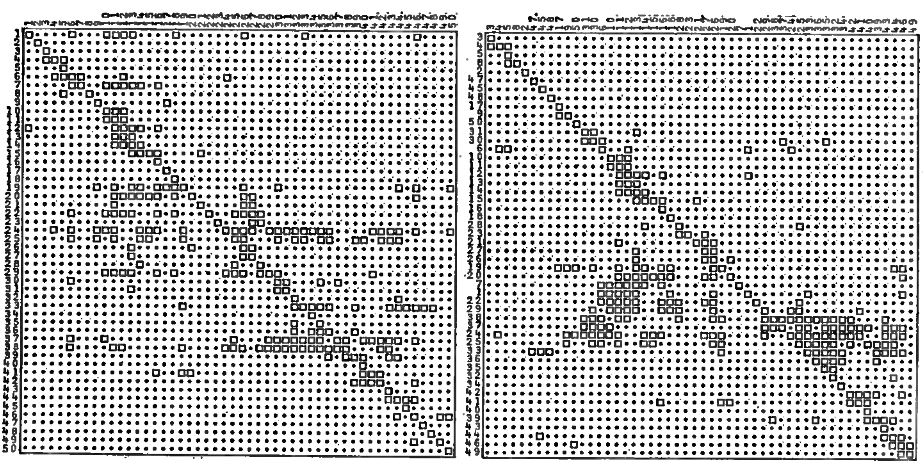


1. Singapore
2. Malay
3. Sabang
4. Aceh I
5. Aceh II
6. Belawan
7. Sumut I
8. Sumut II
9. Sumbar
10. Dumai
11. Ridar I
12. Ridar II
13. Rikep
14. Jambi
15. Sumsel I
16. Sumsel II
17. Bengkulu
18. Lampung
19. Jaya I
20. Jaya II
21. Jabar
22. Jateng I
23. Jateng II
24. Surabaya
25. Jatim
26. Pontianak
27. Kalbar
28. Kalteng
29. Kalsel
30. Kaltim I
31. Kaltim II
32. Sulut I
33. Bitung
34. Sulut II
35. Suteng I
36. Suteng II
37. Makasar
38. Sulsel
39. Sulteng
40. Bali
41. Nusa Tenguara Barat
42. Nusa Tenguara Timur
43. Malut
44. Malteng
45. Malsel
46. Irbaut I
47. Irbaut II
48. Irbaut III
49. Irbasel I
50. Irbasel II

FIG. 8. *Import–export example: regions on the Indonesian islands.*

The control variables in the airport example can be found in McCormick *et al.*[22] Figure 7 shows the symmetric initial data array, the BEA clustering[22] and a clustering corresponding to an optimal solution of $TSP^{col}(= TSP^{row})$, found by Held and Karp's method. The BEA clustering is not optimal and, in fact, not even 3-optimal, since it can be improved by exchanging three links.

The geographical distribution of the regions on the Indonesian islands in the import–export example is given in Figure 8. Figure 9 shows the square but asymmetric initial data array and a clustering corresponding to a 3-optimal solution of TSP^{cow}, found by Lin's heuristic.



(a) (b)

FIG. 9. *Import–export example*; $\cdot$ = 0, $\square$ = 1. (a) *Initial array*; ME = 223. (b) 3-*Optimal clustering subject to* $\rho = \sigma$; ME = 290.

JOB-SHOP SCHEDULING WITH NO INTERMEDIATE STORAGE

Problem description

One of the basic assumptions in most existing theory on machine scheduling is that a job is allowed to wait arbitrarily long before being processed on its next machine.[26] This assumption is highly unrealistic in some real world situations where intermediate storage space is finite or may even be non-existent. The former situation exists for instance in a computer system where buffer space is limited and costly; the latter situation is met in steel or aluminium rolling where the very high temperature of the metal has to be maintained throughout the production process.

Several researchers[27–34] have studied the problem of minimizing the total processing time under the restriction of no intermediate storage in a flow-shop, where the machine order of each job is identical, which implies that the processing order on each machine will be identical, and simplifies the analysis. Van Deman and Baker[35] used a different criterion.

Extensions both to non-zero but finite intermediate storage and to different processing orders per machine seem to complicate the situation considerably. We shall restrict our attention to a job-shop where:

(a) the machine order may vary per job;
(b) each job visits each machine at least once;
(c) no passing is permitted, i.e. the processing order is identical on all machines;
(d) no intermediate storage is allowed.

Reddi and Ramamoorthy[36] consider a more general production process, involving (a), (d) and a "non-subsumption" condition under which lower bounds can be developed. The computation of this lower bound is equivalent to solving a TSP, and the algorithm thus appears to be time-consuming.

TSP formulation

The job-shop scheduling problem can be described as follows:
n jobs J_i $(1 \leqslant i \leqslant n)$ have to be processed on *m machines* M_l $(1 \leqslant l \leqslant m)$;
job J_i $(1 \leqslant i \leqslant n)$ consists of m_i *operations* O_{ik} $(1 \leqslant k \leqslant m_i)$;
the machine order of J_i $(1 \leqslant i \leqslant n)$ is given by $\mu_i = (\mu_i(1), \ldots, \mu_i(m_i))$, i.e. the kth operation O_{ik} of J_i has to be performed on $M_{\mu_i(k)}$;
the processing time of O_{ik} $(1 \leqslant i \leqslant n, 1 \leqslant k \leqslant m_i)$ is given by p_{ik};
the total processing time has to be minimized under the conditions (a)–(d).
We define

$$k'_{il} = \min\{k \mid \mu_i(k) = l, 1 \leqslant k \leqslant m_i\},$$

$$k''_{il} = \max\{k \mid \mu_i(k) = l, 1 \leqslant k \leqslant m_i\},$$

$$P_i = \sum_{k=1}^{k=m_i} p_{ik},$$

$$P'_{il} = \sum_{k=k_{il}'}^{k=m_i} p_{ik},$$

$$P''_{il} = \sum_{k=1}^{k=k_{il}''} p_{ik}.$$

$O_{ik_{il}'}$ and $O_{ik_{il}''}$ are the first and last operations of J_i on M_l; their existence is ensured by condition (b).

For each pair of jobs (J_i, J_j), we will calculate a coefficient c_{ij}, representing the minimum difference between the starting times of O_{i1} and O_{j1} if J_j is scheduled directly after J_i. By condition (c), $O_{ik_{il}''}$ has to precede $O_{jk_{jl}'}$ on M_l, for $1 \leqslant l \leqslant m$. We introduce a directed graph G_{ij} with vertex set N_{ij} and arc set

A_{ij}, defined by

$$N_{ij} = \{O_{hk} \mid h = i, j, 1 \leqslant k \leqslant m_h\};$$

$$A_{ij} = \{(O_{hk}, O_{h,k+1}) \mid h = i, j, 1 \leqslant k \leqslant m_h - 1\} \cup \{(O_{ik_{il}''}, O_{jk_{jl}'}) \mid 1 \leqslant l \leqslant m\};$$

a weight p_{hk} is attached to each vertex $O_{hk} \in N_{ij}$. For an example with $m = 3$, $\mu_i = (2, 1, 2, 3, 2)$ and $\mu_j = (1, 2, 3, 1)$, the graph G_{ij} is given in Figure 10.

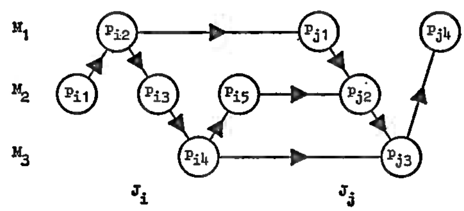


FIG. 10. *An example of the graph G_{ij}.*

As to the *path of maximum weight* (also called *longest* or *critical path*) in G_{ij}, it is clear that

$$\text{it starts from } O_{i1} \text{ and ends in } O_{jm_j}; \tag{5}$$

$$\text{it contains exactly one arc } (O_{ik_{il}''}, O_{jk_{jl}'}). \tag{6}$$

Condition (d) implies that c_{ij} is equal to the latest possible starting time of O_{j1} in G_{ij} if O_{i1} starts at time zero and O_{jm_j} finishes as early as possible. It follows from (5) and (6) that

$$c_{ij} = \max_l (P_{il}'' + P_{jl}') - P_j. \tag{7}$$

The minimum total processing time is now given by

$$\min_\nu \left(\sum_{i=1}^{i=n-1} c_{\nu(i)\nu(i+1)} + P_{\nu(n)} \right), \tag{8}$$

where ν runs over all permutations of $\{1, \ldots, n\}$; $\nu(i)$ is the ith job in a processing schedule.

We add a job J_* with $m_* = m$, $\mu_*(k) = k$ and $p_{*k} = 0$ for $1 \leqslant k \leqslant m$, representing beginning and end of a schedule. According to (7), its coefficients are given by $c_{*i} = 0$, $c_{i*} = P_i$ for $1 \leqslant i \leqslant n$. Determination of (8) now corresponds to solving a TSP with $N = \{*\} \cup \{1, \ldots, n\}$ and (c_{ij}) defined by (7).

This TSP is asymmetric and euclidean. To prove the latter assertion we have to show that $c_{ij} + c_{jk} \geqslant c_{ik}$ for any $i, j, k \in N$, or, equivalently, that

$$\max_l (P_{jl}'' + P_{jl}') + \max_l (P_{jl}'' + P_{kl}') \geqslant \max_l (P_{il}'' + P_{kl}') + P_j.$$

This is true, since for any $l \in \{1, \ldots, m\}$

$$(P''_{il}+P'_{jl})+(P''_{jl}+P'_{kl}) \geqslant (P'_{il}+P'_{kl})+P_j.$$

We make two final remarks on this TSP formulation.

Remark 1. In a flow-shop we know that $\mu_i = (1, 2, \ldots, m)$ for $1 \leqslant i \leqslant m$, and (7) simplifies to $c_{ij} = \max_l(P''_{il} - P''_{j,l-1})$ (note that $P''_{j0} = 0$).[27, 28, 33]

Remark 2. So far, distances have been defined as differences between the starting times of the first operations of jobs. More generally, one might arbitrarily select any two operations O_{ik_i*} and O_{ik_i**} for each job J_i and define c_{ij} as the minimum difference between the starting times of O_{ik_i*} and O_{jk_j**} if J_i precedes J_j directly. This will lead to modifications in (7) and (8), but to an equivalent TSP.[31, 32]

Results

To illustrate the consequences of the no intermediate storage condition, we solved three job-shop scheduling problems[37] under this restriction, using Little's TSP algorithm. In Table 1 the solution values are compared with the lengths of the schedules when infinite intermediate storage is allowed. Figure 11 illustrates the optimal schedules for one of these problems; the unrestricted schedule was found by a method of Florian *et al.*[38] In general, the conditions of no intermediate storage and no passing can be expected to lead to large amounts of idle time on the machines.

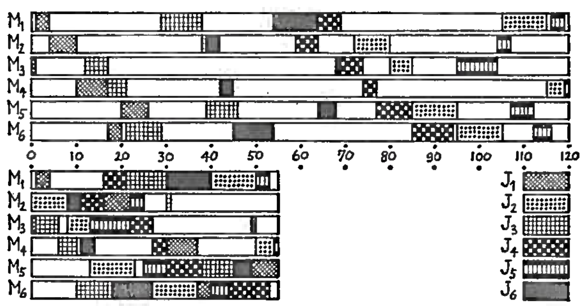


FIG. 11. *Optimal schedules for a* 6 × 6 *problem without and with intermediate storage.*

TABLE 1. EFFECT OF NO INTERMEDIATE STORAGE

Number of jobs	Number of machines	Value without intermediate storage	Value with intermediate storage
6	6	120	55
10	10	2433	972*
20	5	2132	1165

* Indicates that the optimality has not been proved.

ACKNOWLEDGEMENTS

We would like to thank B. J. Lageweg (Mathematisch Centrum) for useful comments and programming advice, J. Visschers and P. ten Kate (Instituut voor Kernphysisch Onderzoek) and J. Berendse and J. H. Kuiper (PTT) for implementing the applications to computer wiring and vehicle routing respectively, and A. W. Roes and T. J. Wansbeek (Instituut voor Verkeers- en Vervoerseconomie) for making available the data for the import–export example.

REFERENCES

[1] J. K. Lenstra and A. H. G. Rinnooy Kan (1974) Some simple applications of the travelling salesman problem. Report BW 38, Mathematisch Centrum, Amsterdam; Working Paper WP/74/12, Graduate School of Management, Delft.

[2] P. Brucker, J. K. Lenstra and A. H. G. Rinnooy Kan (1975) Complexity of machine scheduling problems. Report BW 43, Mathematisch Centrum, Amsterdam.

[3] M. Bellmore and G. L. Nemhauser (1968) The traveling salesman problem: a survey. *Ops Res.* **16**, 538.

[4] A. M. Isaac and E. Turban (1969) Some comments on the traveling salesman problem. *Ops Res.* **17**, 543.

[5] S. Eilon, C. D. T. Watson-Gandy and N. Christofides (1971) *Distribution Management: Mathematical Modelling and Practical Analysis*. Griffin, London.

[6] J. D. C. Little, K. G. Murty, D. W. Sweeney and C. Karel (1963) An algorithm for the traveling salesman problem. *Ops Res.* **11**, 972.

[7] M. Held and R. M. Karp (1970) The traveling-salesman problem and minimum spanning trees. *Ops Res.* **18**, 1138.

[8] M. Held and R. M. Karp (1971) The traveling-salesman problem and minimum spanning trees: part II. *Math. Progr.* **1**, 6.

[9] K. Helbig Hansen and J. Krarup (1974) Improvements of the Held–Karp algorithm for the symmetric traveling-salesman problem. *Math. Progr.* **7**, 87.

[10] M. Bellmore and J. C. Malone (1971) Pathology of traveling-salesman subtour elimination algorithms. *Ops Res.* **19**, 278, 1766.

[11] S. Lin (1965) Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **44**, 2245.

[12] N. Christofides and S. Eilon (1972) Algorithms for large-scale travelling salesman problems. *Opl Res. Q.* **23**, 511.

[13] S. Lin and B. W. Kernighan (1973) An effective heuristic algorithm for the traveling-salesman problem. *Ops Res.* **21**, 498.

[14] J. K. Lenstra (1972) Branch-and-bound algorithmen voor het handelsreizigersprobleem. Report BN 16, Mathematisch Centrum, Amsterdam.

[15] J. B. Kruskal (1956) On the shortest spanning subtree of a graph and the traveling-salesman problem. *Proc. Am. Math. Soc.* **2**, 48.

[16] R. C. Prim (1957) Shortest connection networks and some generalizations. *Bell Syst. Tech. J.* **36**, 1389.

[17] E. W. Dijkstra (1959) A note on two problems in connexion with graphs. *Numer. Math.* **1**, 269.

[18] M. Hanan and J. M. Kurtzberg (1972) A review of the placement and quadratic assignment problems. *SIAM Rev.* **14**, 324.

[19] J. Visschers and P. ten Kate (1973) BAKALG, een bedradingsprogramma met optimalisatie van draadlengte. Report SO-1, Instituut voor Kernphysisch Onderzoek, Amsterdam.

[20] J. Krarup (1974) Private communication.

[21] J. H. Kuiper (1973) "Hoe een PTT-er handelsreiziger werd"—een routing probleem. Unpublished report.

[22] W. T. McCormick, Jr., P. J. Schweitzer and T. W. White (1972) Problem decomposition and data reorganization by a clustering technique. *Ops Res.* **20**, 993.

[23] A. W. Roes (1973) Enige methoden ter verkrijging van een routeschema voor de interinsulaire scheepvaart in Indonesië. Unpublished report.

[24] J. K. Lenstra (1974) Clustering a data array and the traveling-salesman problem. *Ops Res.* **22**, 413.
[25] H. Müller-Merbach (1970) *Optimale Reihenfolgen.* Springer, Berlin.
[26] A. H. G. Rinnooy Kan (1973) The machine scheduling problem. Report BW 27, Mathematisch Centrum, Amsterdam; Report R/73/4, Graduate School of Management, Delft.
[27] J. Piehler (1960) Ein Beitrag zum Reihenfolgeproblem. *Unternehmensforschung* **4**, 138.
[28] S. S. Reddi and C. V. Ramamoorthy (1972) On the flow-shop sequencing problem with no wait in process. *Opl Res. Q.* **23**, 323.
[29] D. A. Wismer (1972) Solution of the flowshop-scheduling problem with no intermediate queues. *Ops Res.* **20**, 689.
[30] G. Liesegang and M. Rüger (1972) Letter. *Opl Res. Q.* **23**, 591.
[31] S. K. Goyal (1973) A note on the paper: On the flow-shop sequencing problem with no wait in process. *Opl Res. Q.* **24**, 130.
[32] S. S. Reddi and C. V. Ramamoorthy (1973) Reply to Dr. Goyal's comments. *Opl Res. Q.* **24**, 133.
[33] J. Grabowski and M. M. Syslo (1973) On some machine sequencing problems (I). *Zastos. Mat.* **13**, 339.
[34] M. M. Syslo (1974) On some machine sequencing problems (II). *Zastos. Mat.* **14**, 93.
[35] J. M. Van Deman and K. R. Baker (1974) Minimizing mean flowtime in the flow shop with no intermediate queues. *AIIE Trans.* **6**, 28.
[36] S. S. Reddi and C. V. Ramamoorthy (1973) A scheduling problem. *Opl Res. Q.* **24**, 441.
[37] J. F. Muth and G. L. Thompson (Eds.) (1963) *Industrial Scheduling.* Prentice-Hall, Englewood Cliffs, N.J.
[38] M. Florian, P. Trepant and G. McMahon (1971) An implicit enumeration algorithm for the machine sequencing problem. *Mgmt Sci.* **17**, B782.