Well-Solved Special Cases of the Traveling Salesman Problem

P. C. Gilmore

Computer Science Department
University of British Columbia

E. L. Lawler
D. B. Shmoys

Computer Science Division
University of California at Berkeley
Berkeley, CA 94720

1. Introduction

Despite the general pessimism contributed by both theory and practice, there is a bright side to the TSP. Many special cases can be easily and efficiently solved. We survey these special cases in this chapter, with the expectation that the reader will find them interesting, instructive and possibly even useful.

There are two broad categories of special cases of the TSP. In one category are problems that are special because of restrictions on the matrix C of arc lengths. For example, C may be upper triangular or a circulant matrix. In a second category are problems in which the TSP is to be solved over a network with a particular structure but with no restriction on the lengths of the arcs. For example, the network may have limited bandwidth.

The majority of the results presented here involve special cases of the first type, with the best known example being the single state-variable sequencing problem [Gilmore & Gomory 1964]. The solution to this problem involves first solving an assignment problem for the distance matrix C and then patching together subtours so as to obtain an optimal solution to the TSP. Since several other special cases involve a similar approach, we have introduced a general theory of subtour patching in an effort to unify results.

Several of the results in this chapter are either new or are presented in an original manner. For example, we believe the formulation of the wallpapering problem presented here is original, and we have obtained more general results than were known previously. In addition, we believe the results on bandwidth-limited graphs and the generalizations of the Gilmore-Gomory model are original. Some of the results contained in this chapter were drawn from the Soviet literature, and were largely unknown in the West. Unpublished notes by E. Ya. Gabovich were extremely valuable in pointing out the most important (and the most obscure) Soviet results.

Throughout this chapter we will make extensive use of concepts related to the symmetric group of permutations on n elements. It is expected that the reader has some familiarity with the concepts, so we briefly note only some of the more important points. Consider an arbitrary permutation φ; $\varphi(i)=j$ denotes that i is mapped to j by φ. Since the set of all permutations on n elements forms a group, for any two permutations τ and φ there exists a unique permutation ψ such that $\tau=\varphi\psi$. As is customary, $\rho\varphi(i)$ denotes $\rho(\varphi(i))$ and also $\tau^{-1}(j)=i$ is equivalent to $\tau(i)=j$. It is well known that every permutation φ has a unique set of disjoint factors. We write permutations in their factored form, e.g. if $\tau(1)=2$, $\tau(2)=1$, $\tau(3)=4$, and $\tau(4)=3$ we write that $\tau=(1,2)(3,4)$.

There is a one-to-one correspondence between permutations and feasible solutions to the assignment problem; $\varphi(i)=j$ has the interpretation that element c_{ij} is used in the assignment. Therefore, tours correspond to permutations where all of the elements (cities) are contained within one factor, that is, cyclic permutations or cycles. Also note that cycles on disjoint sets of elements commute, in contrast to the case of two arbitrary permutations. For an assignment that is not a tour, factors correspond to subtours. The cost of a permutation φ is

$$c(\varphi)=\sum_{i=1}^{n} c_{i\varphi(i)}.$$

Often we will modify one assignment φ by multiplying it by another permutation ψ; we will be interested in the additional cost of this permutation

$$c_{\varphi}(\psi) = c(\varphi\psi) - c(\varphi)$$

above the original cost of φ. When the bottleneck criterion is used we will denote the bottleneck cost

$$\bar{c}(\varphi) = \max_i c_{i\varphi(i)}$$

and, parallel to the case above, we let

$$\bar{c}_{\varphi}(\psi) = \bar{c}(\varphi\psi) - \bar{c}(\varphi).$$

We have indicated only the basic concepts about the symmetric group of permutations that will be used throughout the chapter; the interested reader is referred to [Herstein 1975] for a more comprehensive treatment.

2. The Constant TSP

In this section we consider conditions on the distance matrix C under which all Hamiltonian cycles have the same length. For such a *constant* TSP, one simply needs to find *any* Hamiltonian cycle in the underlying network.

Let $\mathbf{C}$ be the collection of all $n \times n$ matrices C such that $c(\tau)$ is constant for all cyclic permutations τ of the n cities. $\mathbf{C}$ is evidently a linear subspace of the space of all $n \times n$ matrices. That is, if $C_1, C_2 \in \mathbf{C}$, with $c_1(\tau) = \alpha_1$, $c_2(\tau) = \alpha_2$, then $C' = \lambda_1 C_1 + \lambda_2 C_2 \in \mathbf{C}$, because $c'(\tau) = \lambda_1 \alpha_1 + \lambda_2 \alpha_2$ for all τ.

Lemma 1 The dimension of $\mathbf{C}$ is $2n - 1$.

Proof: Let $T = \{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(\omega)}\}$, where $\omega = (n-1)!$, denote the set of all cyclic permutations on n cities. Let

$$t_{ij}^{(k)} = \begin{cases} 1 & \text{if } \tau^{(k)}(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

It follows that

$$-1 = -\frac{1}{n} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} t_{ij}^{(k)}, \quad k = 1,2,\ldots,\omega. \tag{1}$$

In order for a matrix C to belong to $\mathbf{C}$, there must exist a number α such that

$$\sum_{\substack{i,j=1 \\ i \neq j}}^{n} t_{ij}^{(k)} c_{ij} - \alpha = 0, \quad k = 1,2,\ldots,\omega. \tag{2}$$

Equations (2) give us a homogeneous system in $n(n-1) + 1$ variables ($c_{12}, c_{13}, \ldots, c_{n,n-1}$, and α). The coefficient matrix for this system has ω rows and $n(n-1) + 1$ columns. The column vectors will be denoted by $t_{ij} = (t_{ij}^{(1)}, \ldots, t_{ij}^{(\omega)})^T$

$(i,j = 1,2,\ldots,n;\ i \neq j)$ and $t_a = (-1,\ldots,-1)^T$. By (1), we see that t_a is a linear combination of the t_{ij}. Further,

$$t_{1h}^{(k)} = \frac{1}{n-2} \sum_{\substack{i,j=2 \\ i \neq j}}^{n} t_{ij}^{(k)} - \sum_{i=2}^{n} t_{ih}^{(k)}, \quad k = 1,\ldots,\omega,$$

$$t_{h1}^{(k)} = \frac{1}{n-2} \sum_{\substack{i,j=2 \\ i \neq j}}^{n} t_{ij}^{(k)} - \sum_{j=2}^{n} t_{hj}^{(k)}, \quad k = 1,\ldots,\omega,$$

so that t_{1h} and t_{h1}, $h = 2,\ldots,n$, are linear combinations of the t_{ij} with $i,j \neq 1$.

We claim the remaining $(n-1)(n-2)$ vectors t_{ij} $(i,j = 2,\ldots,n;\ i \neq j)$ are linearly independent. For suppose that

$$\sum_{\substack{i,j=2 \\ i \neq j}}^{n} \lambda_{ij}\, t_{ij}^{(k)} = 0, \quad k = 1,\ldots,\omega.$$

Consider the cyclic permutation $\tau^{(k)} = (1,2,\ldots,n)$ with $t_{23}^{(k)} = t_{34}^{(k)} = \ldots = t_{n-1,n}^{(k)} = 1$, $t_{ij}^{(k)} = 0$ otherwise, and also the permutations $(1,3,4,\ldots,n-1,n,2)$, $(1,4,5,\ldots,n,2,3),\ldots,(1,n,2,\ldots,n-3,n-2,n-1)$. (Note that we have eliminated t_{12}, t_{n1} from the problem.) From these permutations we obtain

$$\begin{aligned}
\lambda_{23} + \lambda_{34} + \cdots + \lambda_{n-2,n-1} + \lambda_{n-1,n} &= 0 \\
\lambda_{34} + \lambda_{45} + \cdots + \lambda_{n-1,n} + \lambda_{n,2} &= 0 \\
&\vdots \\
\lambda_{n,2} + \lambda_{23} + \cdots + \lambda_{n-3,n-2} + \lambda_{n-2,n-1} &= 0
\end{aligned}$$

Adding all of these equations yields

$$\lambda_{23} + \lambda_{34} + \cdots + \lambda_{n-1,n} + \lambda_{n,2} = 0.$$

Subtracting this equation from each of the former ones gives

$$\lambda_{23} = \lambda_{34} = \cdots = \lambda_{n-1,n} = \lambda_{n,2} = 0.$$

In this way we can prove that all $\lambda_{ij} = 0$.

Thus, the rank of the coefficient matrix is $(n-1)(n-2)$. It follows that the dimension of $\mathbf{C}$ is $n(n-1) + 1 - (n-1)(n-2) = 2n - 1$. □

Let R_i (C_j) be the $n \times n$ matrix whose ith row (jth column) contains ones and all other elements are zeros.

Lemma 2 Any subset of $2n - 1$ matrices from the set $\{ R_1,\ldots,R_n,\ C_1,\ldots,C_n \}$ is a basis of **C**.

Proof: The $2n$ matrices $R_1,\ldots,R_n,\ C_1,\ldots,C_n$ all belong to **C**, but they are not linearly independent, since

$$\sum_{i=1}^{n} R_i = \sum_{j=1}^{n} C_j .$$

It is easily seen, however, that any $2n - 1$ of them are linearly independent and thus, by Lemma 1, form a basis of **C**. □

Theorem 1 The only matrices C for which all cyclic permutations on the n cities have the same length are those of the form $c_{ij} = a_i + b_j$.

Proof: Follows immediately from Lemma 2. □

Consider a transformation $C' = t(C)$ on distance matrices for which there exist constants α and β such that

$$c'(\tau) = \alpha + \beta\, c(\tau)$$

for all tours τ. Such a transformation is called a *linear admissible transformation;* depending upon the sign of β, it preserves or reverses the total ordering of tours according to length.

Theorem 2 The only linear admissible transformations are those obtained by adding constants a_i to the ith row and b_j to the jth column of a scalar multiple of **C**.

Proof: Let $C' = t(C)$ be such that $c'(\tau) = \alpha + \beta\, c(\tau)$ for all tours τ. Defining $C'' = C' - \beta\, C$, we have $c''(\tau) = \alpha$ for all τ. By Lemma 2, C'' is a linear combination of $R_1,\ldots,R_n, C_1,\ldots,C_n$. Hence $C' = C' + \beta\, C$ can be obtained in the way stated in the theorem. □

Theorems 1 and 2 follow from results of [Berenguer 1979] which were originally stated for the multisalesmen problem (see also [Gabovich 1976]). The proof of Lemma 1 is adapted from [Lenstra & Rinnooy Kan 1979]. Note that Theorem 1 remains true if the adjective "cyclic" is deleted. That is, matrices of the form $c_{ij} = a_i + b_j$ are also the only ones for which all permutations, i. e. assignments, have the same length.

Let $G = (V, A)$ be an arbitrary digraph and C be a distance matrix such that

$$c_{ij} = \begin{cases} a_i + b_j & \text{if } (i,j) \in A, \\ +\infty & \text{otherwise.} \end{cases}$$

Then the TSP is simply the problem of finding one of the Hamiltonian cycles, if any, in G. If G has some special structure, this may be easy. For example, if G

is the line digraph of an Eulerian digraph then G is necessarily Hamiltonian and the problem is easily solved [Syslo 1973]. Other cases in which it is easy to find a Hamiltonian cycle are discussed in Chapter 11.

It does not seem to be possible to obtain such simple conditions on the matrix C for all Hamiltonian cycles to have the same bottleneck length. However, the reader should keep some simple facts in mind, as indicated by the following propositions.

Proposition 1 Adding a constant to all elements of C adds the same constant to the bottleneck length of each tour.

Proposition 2 If C and C' are two matrices whose elements are similarly ordered, i. e.

$$c_{ij} \leq c_{kl} \quad \text{if and only if } c'_{ij} \leq c'_{kl},$$

then the ordering of tours according to bottleneck length is the same for both C and C'.

Proposition 3 Let $k \leq \bar{c}(\tau^*)$, where τ^* is a bottleneck optimal tour. Then replacing C by C', where

$$c'_{ij} = max\{c_{ij}, k\}$$

leaves the bottleneck length of all tours unchanged.

Suppose φ is a bottleneck optimal assignment for C. Then by Propositions 1 and 3, the matrix C', where

$$c'_{ij} = max\{c_{ij} - \bar{c}(\varphi), 0\},$$

is nonnegative and preserves the order of tours with respect to the bottleneck criterion.

Exercise

1. Devise an algorithm to test whether or not a given matrix C is of the form $c_{ij} = a_i + b_j$.

3. The Small TSP

In this section we examine the TSP for small matrices. The results presented are from [Gabovich 1970]. Let us call an $n \times n$ matrix C *small* if there exist n-dimensional vectors a and b such that $c_{ij} = \min\{a_i, b_j\}$. For simplicity of notation, assume that $a_1 \leq a_2 \leq \ldots \leq a_n$. A small matrix where all of the a_i and b_j are distinct is said to have *distinct values*. Let d_i be the ith smallest of the $2n$ distinct values a_i and b_j and then let $D = \{d_1, d_2, \ldots, d_n\}$. In addition, let

$d = \sum_{i=1}^{n} d_i$. Consider solving the TSP for the distance matrix C. The length of an optimal tour can be found easily, and is limited to only a handful of different values.

Theorem 3. Let C be a small matrix with distinct values. The length of an optimal tour for C is d if and only if one of the following three conditions holds:

(S1) For some city i, both a_i and b_i are in D.
(S2) $D = \{a_1, a_2, \ldots, a_n\}$.
(S3) $D = \{b_1, b_2, \ldots, b_n\}$.

Proof: Suppose that (S1) holds. The cities can be partitioned into four sets: those with neither a_i nor b_i in D; those with only a_i in D; those with only b_i in D; those with both a_i and b_i in D. Call these sets D_0, D_a, D_b and D_2, respectively. By a very simple counting argument, $|D_0| = |D_2|$. Construct a tour as follows. Start at a city in D_2. Then, in any order, visit the cities of D_a. Next, go to a city in D_0. This is followed by visiting the cities of D_b in any order. The tour is completed by visiting the remaining cities in D_2 and D_0 alternately; that is, one from D_2, then one from D_0, and so on. It is not hard to see that the cost of this tour is d. Since the values are distinct, each arc of a tour must have a different value, and thus the cost of any tour must be at least d. Next suppose that (S2) holds. This is a constant TSP and the cost of any tour is d. The same is true for (S3).

Finally, assume that (S1), (S2), and (S3) all do not hold. It follows that $D = \{a_1, \ldots, a_k, b_{k+1}, \ldots, b_n\}$ for some $1 \le k \le n-1$. Suppose that a tour of length d existed. The costs of the arcs in this tour must correspond precisely to those costs in D. Therefore, in this tour some arcs have costs that correspond to a values and some that correspond to b values. Therefore, at some point in the tour an arc with cost b_i must be followed by one with cost a_j. But for this to happen, i must equal j, which is impossible. □

An almost identical proof will give a slightly stronger version of this theorem.

Theorem 3'. Let C be a small matrix with distinct values. Let $D' = \{d'_1, d'_2, \ldots, d'_n\}$ be some set of a_i and b_i values. Then there exists a tour that uses precisely those costs if and only if one of the following three conditions holds:

(S1') For some city i, both of the values a_i and b_i are in D'.
(S2') $D' = \{a_1, a_2, \ldots, a_n\}$.
(S3') $D' = \{b_1, b_2, \ldots, b_n\}$.

In Theorem 3, it was shown that d cannot be attained under certain conditions. In this case, what is the optimal value? This question is answered by the next result.

Theorem 4. Let C be a small matrix with distinct values The length of the shortest tour for the TSP given by C is either d, $d - d_n + d_{n+1}$, or $\min\{d - d_n + d_{n+2}, d - d_{n-1} + d_{n+1}\}$. Furthermore, suppose that the optimal cost is not d; then the optimal cost is greater than $d - d_n + d_{n+1}$ if and only if one of the following three conditions holds:

(S4) $k=1$; $d_n=b_2$; $d_{n+1}=a_2$.

(S5) $k=n-1$; $d_n=a_{n-1}$; $d_{n+1}=b_{n-1}$.

(S6) $2\le k\le n-2$; either ($d_n=a_k$ and $d_{n+1}=b_k$) or ($d_n=b_{k+1}$ and $d_{n+1}=a_{k+1}$).

Here k is the largest indexed city that has its a value in D, the set of the n smallest values.

Proof: Since the case of an optimal tour of length d was completely characterized by Theorem 3, assume that the optimal tour costs more than d. The next best possible value is $d'=d-d_n+d_{n-1}$, and to see if this can be attained, consider $D'=D\cup\{d_{n+1}\}-\{d_n\}$. By Theorem 3', d' is attainable if and only if D' satisfies (S1'), (S2'), or (S3').

It is possible to list precisely those cases when all three of these conditions fail. As indicated in the proof of Theorem 3, if the optimal value is not d, $D=\{a_1,\ldots,a_k,b_{k+1},\ldots,b_n\}$. First, note that if $d_n=b_i$ and $d_{n+1}=b_j$, (S1') will be satisfied. Therefore, for all three to fail, either $d_n=a_k$, or $d_{n+1}=a_{k+1}$. Suppose that $d_n=a_k$; it is easy to see that condition (S1') will be satisfied unless $d_{n+1}=b_k$. If d_{n+1} does equal b_k, then $D'=\{a_1,\ldots,a_{k-1},b_k,\ldots,b_n\}$. If $k=1$, this implies that (S3') holds; otherwise (S1'), (S2'), and (S3') all fail to hold. Finally, suppose that $d_{n+1}=a_{k+1}$. Again, condition (S1') will hold unless $d_n=b_{k+1}$. In this case, $D'=\{a_1,\ldots,a_{k+1},b_{k+2},\ldots,b_n\}$. If $k=n-1$, it follows that (S2') is satisfied, but otherwise all three conditions fail. These cases are precisely those stipulated by (S4), (S5), and (S6).

To complete the proof of the theorem, suppose that d' is not attainable. It is straightforward to show that the restrictions placed on d_n and d_{n+1} insure that both $D-\{d_n\}\cup\{d_{n+2}\}$ and $D-\{d_{n-1}\}\cup\{d_{n+1}\}$ must satisfy one of (S1'), (S2'), and (S3'). Since both $d-d_n+d_{n+2}$ and $d-d_{n-1}+d_{n+1}$ are attainable, the optimum is simply the minimum of the two values. □

It is interesting to consider the case where the values a_i and b_i are not necessarily distinct.

Theorem 5. Consider an instance of the TSP given by a small distance matrix. The length of an optimal tour is equal to d if and only if one of the following four conditions holds:

(S7) (S1),(S2), or (S3) holds.

(S8) $d_n=d_{n+1}$; none of (S4), (S5) and (S6) holds.

(S9) $d_n=d_{n+1}=d_{n+2}$.

(S10) $d_{n-1}=d_n=d_{n+1}$.

Proof: Left as an exercise. □

Exercises

2. Prove Theorem 5.

3. Formulate and prove the theorem corresponding to Theorem 4 when a_i and b_j are not assumed to be distinct.

4. Construct an $O(n)$ time algorithm to find an optimal tour for small matrices. (The input must be a and b.) *Hint:* It is possible to find the median of n numbers in $O(n)$ time.

5. Prove that if C is a symmetric small matrix then the length of the optimal tour is d.

4. Circulant Matrices

In this section we show how to find a shortest Hamiltonian path in the case that C is a circulant matrix. Although we do not know of a polynomial algorithm for the TSP for circulants, the Hamiltonian path result does enable us to obtain an approximate solution that is quite close to the optimum in many cases.

A *circulant* is an $n \times n$ matrix of the form

$$\begin{pmatrix} c_0 & c_1 & c_2 & & c_{n-2} & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \cdots & c_{n-3} & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & \cdots & c_{n-4} & c_{n-3} \\ \vdots & & & \cdots & \vdots & \\ c_1 & c_2 & c_3 & & c_{n-1} & c_0 \end{pmatrix}.$$

The cells (i,j) such that $(j - i) = k \pmod n$ have the same value c_k; these cells comprise the kth *stripe* of C. Note that each stripe yields a feasible solution to the assignment problem defined by C.

Theorem 6 [Garfinkel 1977] The number of subtours in the assignment given by the kth stripe is gcd (k,n).

Proof: Cities i and j are in the same subtour if and only if there exist integers m_1, m_2 such that $j - i = m_1 k + m_2 n$. It follows from elementary number theory that i and j are in the same subtour if and only if $i = j \pmod g$, where $g = \gcd(k, n)$. Hence there are $\gcd(k, n)$ subtours, each of which contains $n / \gcd(k, n)$ cities. □

Corollary 1 If gcd $(k,n) = 1$, then the kth stripe yields a Hamiltonian cycle.

Corollary 2 If n is prime then each stripe, other than the 0th, yields a Hamiltonian cycle.

Let $c_{k(0)} \le c_{k(1)} \le c_{k(2)} \le \ldots \le c_{k(n-1)}$ and let

$$g_0 = \gcd(k(0), n),$$
$$g_{i+1} = \gcd(k(i+1), g_i).$$

Lemma 3 A lower bound on the length of a shortest Hamiltonian path is given by

$$(n - g_0)\, c_{k(0)} + (g_0 - g_1)\, c_{k(1)} + \ldots + (g_{n-2} - g_{n-1})\, c_{k(n-1)}.$$

Proof sketch: Ignore directions of arcs and consider the undirected multigraph that results. Edges from stripe $k(0)$, with cost $c_{k(0)}$, yield a subgraph with g_0 connected components. Edges from stripes $k(0)$ and $k(1)$ yield a subgraph with g_1 connected components, and so forth. It is now seen that a shortest spanning tree, such as one obtained by Kruskal's algorithm, has length equal to the asserted lower bound. Since every Hamiltonian path is a spanning tree, the length of a shortest spanning tree is a lower bound on the length of a shortest Hamiltonian path. □

Theorem 7 [Bach, Luby & Goldwasser 1982] The nearest neighbor rule, starting from any city, yields a shortest Hamiltonian path.

Proof sketch: Show that the nearest neighbor rule yields a Hamiltonian path whose length is equal to the lower bound given in Lemma 3. □

Corollary 3 The nearest neighbor rule, starting from any city, yields a tour that differs in length from the optimum by no more than $c_{k(n-1)} - c_{k(0)}$.

Proof: No tour can be shorter than the length of a shortest Hamiltonian path plus $c_{k(0)}$. The nearest neighbor rule chooses one arc in addition to a shortest Hamiltonian path and its length cannot exceed $c_{k(n-1)}$. □

We comment that the Hamiltonian path produced by the nearest neighbor rule is optimal in a very strong sense--its kth shortest arc is as short as the kth shortest arc in any other Hamiltonian path. In particular the path is bottleneck optimal.

Exercises

6. Let C be an $n \times n$ circulant, where

$$c_k = ak + b \pmod{n}$$

with gcd $(a,n) = 1$. The elements of C are thus $0,1,2,\ldots,n-1$.

(a) Show that $g_1 = 1$ for C (where $c_{k(0)} = 0$, $c_{k(1)} = 1$). (*Hint*: It may be easiest to show gcd $(k(1) - k(0), n) = 1$, which implies the desired result.)

(b) Show that the length of any Hamiltonian cycle is a multiple of n.

(c) Use the above results to show that the nearest neighbor rule yields an optimal Hamiltonian *cycle*.

7. (Open question) Is there a polynomial-time algorithm for solving the TSP for circulants, or is this problem NP-hard?

5. Upper Triangular Matrices

We say that C is *upper triangular* if $i \geq j$ implies $c_{ij} = 0$. In this section we shall show that the TSP for upper triangular matrices is essentially as easy as

the assignment problem [Lawler 1972].

Lemma 4 Let C be upper triangular and φ be an assignment that is optimal subject to the constraint that $\varphi(n)=1$. Then $c(\varphi)$ is a lower bound on the length of an optimal tour.

Proof: Call an arc (i,j) *backward* if $i \geq j$; for such an arc $c_{ij} = 0$. Let τ be an optimal tour. Remove each backward arc from the part of τ that extends from city n to city 1. The result is a set of paths, each extending from a city j to a city i, with $j \leq i$. Now turn each of these paths into a cycle by adding a backward arc from i to j. The result is an assignment φ with $\varphi(n)=1$ and $c(\varphi)=c(\tau)$. □

Theorem 8 Let C be upper triangular and φ be an assignment that is optimal subject to the constraint that $\varphi(n)=1$. Then $c(\varphi)$ is equal to the length of an optimal tour τ that can be easily constructed from φ.

Proof: Let φ be an assignment that is optimal subject to the constraint that $\varphi(n)=1$. If φ is not a tour then it consists of s subtours, where $s \geq 2$, each of each contains at least one backward arc (as defined in the proof of Lemma 4). Remove arc $(n,1)$ from the subtour containing cities 1 and n and any one backward arc from each of the other subtours. The result is a set of paths, one extending from city 1 to city n, and the others from j_1 to i_1, j_2 to i_2, ..., j_{s-1} to i_{s-1}, where $j_1 > j_2 > \cdots > j_{s-1}$ and $i_1 \geq j_1 > j_2$, $i_2 \geq j_2 > j_3$, ..., $i_{s-1} \geq j_{s-1} > 1$. Now add backward arcs (n,j_1), (i_1,j_2),..., (i_{s-2},j_{s-1}), $(i_{s-1},1)$ to obtain a tour τ with $c(\tau)=c(\varphi)$. Since, by Lemma 4, $c(\varphi)$ is a lower bound on the length of an optimal tour, τ is optimal. □

Note that an assignment φ that is optimal subject to the constraint that $\varphi(n)=1$ is easily obtained by applying any algorithm for the assignment problem to the $(n-1)\times(n-1)$ matrix C' that results from the deletion of column 1 and row n from C. Standard assignment algorithms require no more than $O(n^3)$ time. The construction of an optimal tour, as indicated in the proof of Theorem 8 requires considerably less time. The reader may be interested in verifying that the construction requires no more than $O(n)$ time.

As a simple example, let

$$C = \begin{bmatrix} 0 & -1 & 7 & -20 & 3 & -2 & 5 \\ 0 & 0 & 12 & 8 & 16 & 9 & 8 \\ 0 & 0 & 0 & 3 & 7 & 6 & 2 \\ 0 & 0 & 0 & 0 & 4 & 4 & 9 \\ 0 & 0 & 0 & 0 & 0 & -18 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then

$$C' = \begin{bmatrix} -1 & 7 & \mathbf{-20} & 3 & -2 & 5 \\ \mathbf{0} & 12 & 8 & 16 & 9 & 8 \\ 0 & 0 & 3 & 7 & 6 & \mathbf{2} \\ 0 & \mathbf{0} & 0 & 4 & 4 & 9 \\ 0 & 0 & 0 & 0 & \mathbf{-18} & -1 \\ 0 & 0 & 0 & \mathbf{0} & 0 & 3 \end{bmatrix}$$

and an optimal solution to the assignment problem is indicated in bold face. This is converted to an optimal solution to the TSP as shown in Figure 1.

– Insert Figure 1 about here --

Exercises

8. Devise a procedure to determine whether or not a matrix C can be made upper triangular by adding constants to its rows and columns and by renumbering the cities (effecting a symmetric permutation of rows and columns). (*Hint*: Guess at the identity of the cities to be numbered 1 and n. This determines the constants to be added to rows and columns, almost uniquely. Then see if it is possible to renumber the remaining $n-2$ cities to achieve upper triangularity. This question is essentially equivalent to determining whether or not a given matrix is the adjacency matrix of an acyclic digraph.)

9. Suppose C is upper triangular and nonnegative. Show that the length of a shortest path from city 1 to city n is equal to the length of an optimal tour. (This means that an $O(n^2)$ shortest path computation suffices, instead of an $O(n^3)$ assignment computation.)

10. What adaptations of the algorithm of this section are required to solve the bottleneck TSP? (The bottleneck assignment problem can be solved in $O(n^{2.5} \log n)$ time [Hopcroft & Karp 1973].)

11. An author is writing a book with n sections. She would prefer that certain sections precede others, because of the relationships in their contents. She is able to specify a partial order, $\leq$, describing the desired precedence relations. How should she order the sections so as to minimize the number of times that there is a section x in the book immediately preceding y, with $x \not\leq y$? Formulate as a TSP with an upper triangular distance matrix. (Note: A feasible solution to the TSP may not be consistent with the partial order, in the sense that there may be a section x preceding a section y, with $y \leq x$. If consistency with the partial order is demanded, the resulting "optimal linear extension" problem is known to be NP-hard.)

6. Graded Matrices

We say that a matrix C is *graded across its rows* if $c_{ij} \leq c_{i,j+1}$ for all i,j and *graded up its columns* if $c_{ij} \geq c_{i+1,j}$, for all i,j. A matrix is *doubly graded* if it is graded both across its rows and up its columns.

The TSP is NP-hard for graded, even doubly graded, matrices since any matrix can be made doubly graded by a linear admissible transformation, that is, by adding constants to its rows and columns. However, it is possible to obtain a useful approximation result for graded matrices, as we show below. In Section 10 we show that there is a polynomial algorithm for obtaining an optimal solution to the bottleneck TSP for graded matrices.

Theorem 9. Let C be nonnegative and graded up its columns. Given an optimal assignment φ it is easy to construct a tour τ such that

$$c(\tau) \leq c(\varphi) + \max_j \{ c_{1j} \}.$$

Proof: If φ is a tour, let $\tau = \varphi$. Else choose one city from each of the $m \geq 2$ subtours of φ, and let these cities be $i_1, i_2, \ldots, i_m$, with $i_1 < i_2 < \ldots < i_m$. Remove arcs $(i_1, \varphi(i_1)), \ldots (i_m, \varphi(i_m))$ from φ and substitute arcs $(i_2, \varphi(i_1))$, $(i_3, \varphi(i_2)), \ldots, (i_{m-1}, \varphi(i_{m-2}))$, $(i_m, \varphi(i_{m-1}))$ and $(i_1, \varphi(i_m))$. From

$$\begin{aligned} c_{i_1 \varphi(i_1)} &\geq c_{i_2 \varphi(i_1)}, \\ c_{i_2 \varphi(i_2)} &\geq c_{i_3 \varphi(i_2)}, \\ &\vdots \\ c_{i_{m-1} \varphi(i_{m-1})} &\geq c_{i_m \varphi(i_{m-1})}, \end{aligned}$$

it follows that

$$\begin{aligned} c(\tau) &\leq c(\varphi) + c_{i_1 \varphi(i_m)} - c_{i_m \varphi(i_m)} \\ &\leq c(\varphi) + \max_j \{ c_{ij} \}. \end{aligned}$$

□

As a very simple example, let

$$C = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 & \mathbf{0} \\ 4 & 3 & 2 & 1 & \mathbf{0} & 0 \\ 3 & 2 & 1 & \mathbf{0} & 0 & 0 \\ 2 & 1 & \mathbf{0} & 0 & 0 & 0 \\ 1 & \mathbf{0} & 0 & 0 & 0 & 0 \\ \mathbf{0} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3}$$

with an optimal assignment φ indicated by the bold entries. Note that φ has three subtours: (1, 6), (2, 5), and (3, 4). Letting $i_1 = 1$, $i_2 = 2$, $i_3 = 3$, we convert φ to a tour τ as shown in Figure 2. This gives us $c(\tau) = c(\varphi) + c_{31} - c_{61} = c(\varphi) + 3$.

-- Insert Figure 2 about here --

7. Pyramidal Tours

Let us say that a tour τ on cities $1,2,\ldots,n$ is *pyramidal* if, for each city j, $1 < j < n$, either $\tau^{-1}(j) < j < \tau(j)$ or $\tau^{-1}(j) > j > \tau(j)$. In other words τ is pyramidal if it is of the form $(1, i_1, \ldots, i_r, n, j_1, \ldots, j_{n-r-2})$, where $i_1 < i_2 < \ldots < i_r$ and $j_1 > j_2 > \ldots > j_{n-r-2}$.

An equivalent characterization of pyramidal tours is as follows. Let Δ_n denote the set of all pyramidal tours on cities $1,2,\ldots,n$. Then $\Delta_2 = \{(1, 2)\}$ and Δ_{n+1}, for $n \geq 2$, contains all permutations of the form $(n, n+1)\tau$ or $\tau(n, n+1)$, where $\tau \in \Delta_n$, and only such permutations.

For any distance matrix C it is possible to compute a shortest pyramidal tour by the following dynamic programming scheme. Let $C(i, j)$ denote the length of a shortest Hamiltonian path from i to j on cities $1,2,\ldots, max\{i, j\}$, subject to the condition that the path passes through cities in descending order of

index from i to 1 and then through the complementary subset in ascending order of index from 1 to j. By the usual sort of argument involving the principle of optimality, we have

$$C(i,j) = \begin{cases} C(i, j-1) + c_{j-1,j} & \text{for } i < j-1, \\ \min_{k<i} \{ C(i,k) + c_{kj} \} & \text{for } i = j-1, \\ C(i-1, j) + c_{i,i-1} & \text{for } i > j+1, \\ \min_{k<j} \{ C(k,j) + c_{ik} \} & \text{for } i = j+1. \end{cases} \tag{4}$$

It is possible to compute $C(i,n)$ and $C(n,i)$, for all $i < n$ in $O(n^2)$ time, starting from the initial conditions $C(1,2) = c_{12}$ and $C(2,1) = c_{21}$. The length of a shortest pyramidal tour is then given by

$$min \{ C(n-1, n) + c_{n,n-1}, C(n, n-1) + c_{n-1,n} \}.$$

Let us now consider conditions under which we can be assured that there exists an optimal tour that is pyramidal. For a given matrix C, let

$$d_{ij} = c_{ij} + c_{i+1,j-1} - c_{i,j-1} - c_{i+1,j}, \tag{5}$$

where by definition $c_{ij} = 0$ if $i = n+1$ or $j = 0$. It is easy to establish that

$$c_{ij} = \sum_{k=i}^{n} \sum_{l=1}^{j} d_{kl}.$$

If the matrix $D = (d_{ij})$, as defined by (5), is nonnegative, we say that C is a *(cumulative) distribution* matrix generated by the *density* matrix D.

Lemma 5 If C is a distribution matrix then

$$c_{ij'} + c_{i'j} \geq c_{ij} + c_{i'j'},$$

for all $i < i'$, $j < j'$.

Proof: Left as an exercise. □

Corollary 4 If C is a distribution matrix then the identity permutation $\iota(i) = i$ is an optimal assignment.

Proof: Let φ be an optimal assignment. If φ is not the identity permutation, then there are cities i, i', with $i < i'$ such that $\varphi(i) > \varphi(i')$. Apply Lemma 5, with $j' = \varphi(i)$, $j = \varphi(i')$, to obtain an assignment φ', where $\varphi'(i) = j$, $\varphi'(i') = j'$, $\varphi'(k) = \varphi(k)$ for $k \neq i, i'$, and $c(\varphi') \leq c(\varphi)$. A finite sequence of such rearrangements yields the identity permutation, which must therefore be optimal. □

Theorem 10 If C is a distribution matrix then there exists an optimal tour that is pyramidal.

Proof: By induction on the number of cities. The theorem is trivially true for two cities. So let us assume it is true for $n-1$ cities and consider a problem with n cities. Let τ be an optimal tour. If τ is not pyramidal then there is a "peak" $j \neq n$ such that $\tau^{-1}(j) < j > \tau(j)$. (The reader may care to show that the number of "peaks" is equal to the number of "valleys," where a valley j is such that $\tau^{-1}(j) > j < \tau(j)$.) Let $i = j$, $i' = \tau^{-1}(j)$, $j' = \tau(j)$ and apply Lemma 5. The result is a permutation τ', with $\tau'(j) = j$, $\tau'(i') = j'$, $\tau'(k) = \tau(k)$ for $k \neq i', j$ and with $c(\tau') \leq c(\tau)$. The permutation τ' has two subtours, one containing j only and the other containing the other $n-1$ cities. By inductive assumption, if the latter subtour is not pyramidal, it can be replaced by a pyramidal tour of no greater length. (A submatrix of a distribution matrix is a distribution matrix.) We now "patch" j back into τ' Let i be such that $i < j < \tau'(i)$. Now apply Lemma 5 with $i' = j$ and $j' = \tau(i)$. The result is a pyramidal tour τ'' on the n cities with $c(\tau'') \leq c(\tau') \leq c(\tau)$ and the theorem is proved. □

As a simple application of Theorem 10, consider the distribution matrix C generated by the density matrix D, where

$$C = \begin{bmatrix} 4 & 16 & 20 & 23 & 40 \\ 3 & 12 & 15 & 17 & 33 \\ 3 & 10 & 13 & 15 & 31 \\ 1 & 4 & 5 & 6 & 18 \\ 0 & 3 & 3 & 4 & 14 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 3 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 2 & 4 & 2 & 1 & 4 \\ 1 & 0 & 1 & 0 & 2 \\ 0 & 3 & 0 & 1 & 10 \end{bmatrix}.$$

Solving for a shortest pyramidal tour by the recurrence relations (4), we obtain $\tau = (1, 4, 5, 3, 2)$ with $c(\tau) = 45$.

As we have seen, if the matrix C is a distribution matrix then the TSP can be solved in $O(n^2)$ time by applying the dynamic programming technique for finding a shortest pyramidal tour. As we show in the exercises below and in the next section, there are other conditions under which there exists an optimal tour that is pyramidal so that the same dynamic programming technique can be applied. In Section 13 we shall show that there are also special cases of distribution matrices for which it is possible to solve the TSP in less than $O(n^2)$ time.

Exercises

12. Prove that the two characterizations of pyramidal tours given in the beginning of this section are equivalent.
13. Prove that C is equivalent to a distribution matrix by a linear admissible transformation if and only if $d_{ij} \geq 0$ for $i = 1,2,\ldots,n-1$, $j = 2,3,\ldots,n$, where d_{ij} is as defined by (5).
14. Prove Lemma 5.
15. Suppose that

$$c_{ik} \geq max\{c_{ij}, c_{jk}\},$$

$$c_{ik} \geq max\{c_{ji}, c_{kj}\},$$

for all $i < j < k$. Prove that there exists a *bottleneck* optimal tour that is pyramidal.

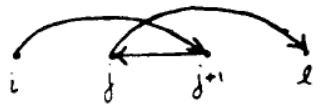
16. Let $a_1 \geq a_2 \geq \ldots \geq a_n \geq 0$, $0 \leq b_1 \leq b_2 \leq \ldots \leq b_n$. Show that each of the following matrices is a distribution matrix:

(a) $c_{ij} = a_i + b_j$.
(b) $c_{ij} = a_i b_j$.
(c) $c_{ij} = |a_i - a_j|$.
(d) $c_{ij} = min\{a_i, b_j\}$.
(e) $c_{ij} = max\{a_{n-i+1}, b_j\}$.

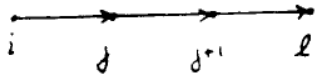
8. The Demidenko Conditions

In the previous section it was shown that for certain classes of matrices, there must be an optimal tour that is pyramidal. It was further shown that an optimal pyramidal tour can be found in $O(n^2)$ time. In this section, we show that a broad class of matrices has the property that there exists an optimal tour that is pyramidal. This result subsumes several results from the Soviet literature, which show that more restricted classes of matrices have optimal solutions that are pyramidal (for example, see Exercise 17).

The class of TSP instances **C** is defined by a set of conditions, each of which depends only on four cities. For any four cities, $(i,j,j+1,l)$ where $i<j<j+1<l$, the following conditions must hold: the path

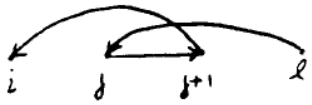


is at least as long as

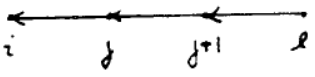


(6)

and the symmetric condition that the path

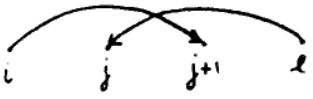


is at least as long as

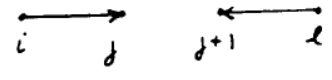


: (7)

the pair of arcs

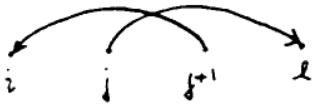


is at least as long as

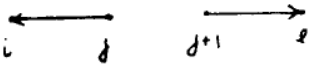


(8)

and its symmetric counterpart that the pair



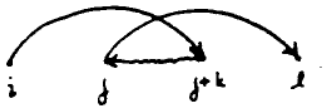
is at least as long as



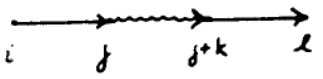
. (9)

First, two preliminary lemmas are given.

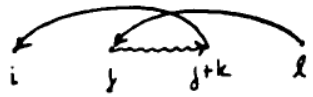
Lemma 6 Let $C \in \mathbf{C}$, and $i,j,j+1,\ldots,j+k,l \in \{1,\ldots,n\}$ where $i<j$, $j+k<l$, and $1 \le k \le n-j-1$. It follows that



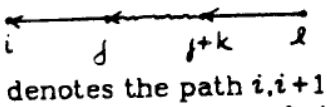
is at least as long as



and that, symmetrically,

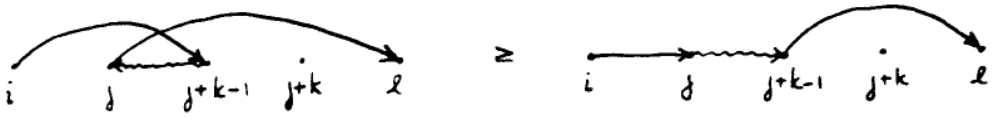


is at least as long as

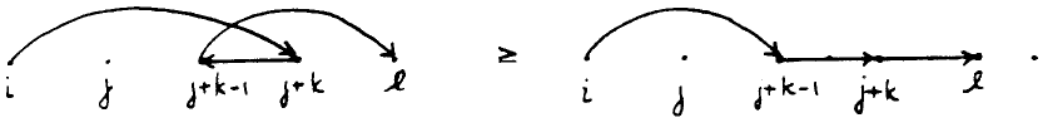


(Throughout this section, a "wiggly" arc from i to j denotes the path $i,i+1,\ldots,j$. In addition, from here on, the obvious $\ge$ will be used to denote the words "is at least as long as".)

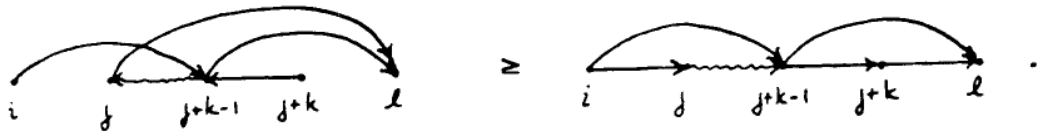
Proof: By induction on k. For $k=1$, these inequalities reduce to the conditions (6) and (7) which must hold (since $C \in \mathbf{C}$). Suppose that the lemma were true for $k-1$. Then,



by the inductive hypothesis. But, by (6),

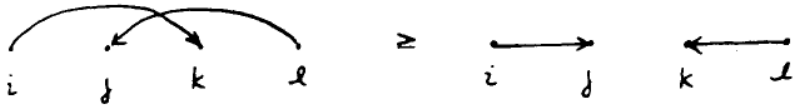


Add the two inequalities to get



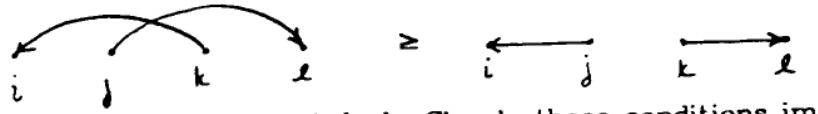
The arcs $(i,j+k-1)$ and $(j+k-1,l)$ on both sides cancel, giving the desired result. The symmetric case has an exactly parallel proof. □

Consider the conditions,



(10)

and



(11)

for arbitrary (i,j,k,l) where $i<j<k<l$. Clearly these conditions imply (8) and

(9); however, the converse is also true.

Lemma 7 The conditions (8) and (9) hold for arbitrary $(i,j,j+1,l)$, $i<j<j+1<l$, if and only if conditions (10) and (11) hold for arbitrary (i,j,k,l) where $i<j<k<l$.

Proof: Left as an exercise to the reader. □

Consider an arbitrary tour τ. As in Section 7, call i a *peak* of τ if $\tau^{-1}(i)<i>\tau(i)$; call i a *valley* of τ if $\tau^{-1}(i)>i<\tau(i)$; otherwise i is called an *intermediate term*. $P(\tau)$, $V(\tau)$, and $I(\tau)$ denote the sets of peaks, valleys, and intermediate terms, respectively.

Consider the cycle on eight cities, $\tau=(1,3,5,4,7,8,6,2)$. We can "graph" this cycle as is depicted in Figure 3.

– Insert Figure 3 about here –

In Figure 3 it is easy to see that 5 and 8 are the only peaks, whereas 1 and 4 are the only valleys. For every peak $i \in P(\tau)$, define the set $S(\tau,i)$ to be the *slopes* of the peak; that is, include all points that form a decreasing sequence by repeatedly applying τ to i, or repeatedly applying τ^{-1} to i, up to, but *not* including the next valleys. Therefore, in the above example, $S(\tau,5)=\{3,5\}$ and $S(\tau,8)=\{7,8,6,2\}$. Furthermore, if τ is pyramidal there is exactly one peak, n, and $S(\tau,n) = \{2,3,\ldots,n\}$.

The following lemma is an equivalent characterization of pyramidal tours.

Lemma 8 A tour τ is pyramidal if and only if it satisfies the following condition: for any peak $k \in P(\tau)$, $3 \le k \le n$, and any i, $0 \le i \le k-3$,

$$\{k,\ldots,k-i\} \subset S(\tau,k) \Rightarrow (k-i-1 \in P(\tau) \text{ or } k-i-1 \in S(\tau,k)).$$

Proof: Let k be a peak in $P(\tau)$. Suppose that the property above does not hold for some k and i within the specified range, but does hold for all j, $k > j > i$. In this case, k and i will be called a *bad* (k,i) pair. The claim is that τ is pyramidal if and only if there are no bad pairs. For a pyramidal tour τ, $S(\tau,n)=\{2,\ldots,n\}$, so by checking the boundary conditions, it follows that no bad pairs exist. To prove the other direction, assume that a tour has no bad pairs. Find the peak k of smallest size. Consider $S(\tau,k)$; for some i, $k-i \in S(\tau,k)$ and $k-i-1 \notin S(\tau,k)$. But τ has no bad pairs, and by the choice of k, $k-i-1$ is not a peak. Therefore, $k-i \le 2$, and since the valleys on either side of k must be less than $k-i$, both of these valleys must be 1. But then k is the only peak, and hence τ is pyramidal. □

Theorem 11 [Demidenko 1980] Let $C \in \mathbf{C}$. For any tour τ there exists a pyramidal tour τ_0 of no greater cost.

Proof: We shall make critical use of Lemma 8. Suppose that τ is a non-pyramidal tour. We give a construction for finding a pyramidal tour of no greater cost by eliminating all of the bad pairs in τ. For any bad pair (k,i), the structure of the peak k and its relationship to $k-i-1$ is limited to a small number of cases. For each of these cases we define an *elementary* transformation that removes the bad pair without increasing the length the tour. These

elementary transformations are used to construct a set of *composite* transformations. These composite transformations will have the following property: if (k,i) is the bad pair that a particular transformation is designed to remove, not only does it fix (k,i), but no new bad pair (k',i') with $k'-i' \geq k-i$ is created. Thus, by finding the bad (k,i) pair with the largest $k-i$ and fixing that pair, these composite transformations can be used to remove all bad (k,i) pairs where $k-i \geq 3$.

To complete the proof of the theorem, all that is left is to show is that these composite transformations can indeed be constructed. The proof given by Demidenko is very technical and only a sketch of it is given here.

Consider an arbitrary bad pair (k,i) for some permutation τ. There are two basic cases: either (I) $k-i-1 \in V(\tau)$, or (II) $k-i-1 \in S(\tau,l)$, $l>k$. (Since none of $k-1,\ldots k-i$ are peaks, note that $k-i-1$ cannot be on a slope of a peak lower than k.) Within both of these cases there are several subcases. For case I, $k-i-1$ can either be a valley of the peak k, or it can be a valley of another peak. From this breakdown we get the three cases that are depicted in Figure 4, and their mirror images.

-- Insert Figure 4 about here --

In case I(b), as well as in the cases that follow, the point $k-i+q$ denotes the smallest point on the right slope that is larger than $k-i$. The points $k-i+p$ and $k-i+r$ are, respectively, the consecutive points on the left slope that are just lower and just higher than $k-i+q$. (Note that r need not be $p+2$.)

Consider case I(a). The elementary transformation for this case alters the tour by visiting city $k-i-1$ in between cities $\tau^{-1}(k-i)$ and $k-i$. This is depicted in Figure 5. The dashed line superimposed on the original tour indicates the tour after the transformation has been made.

-- Insert Figure 5 about here --

To prove that the new tour is no longer than τ two cases must be considered. Suppose that $\tau^{-1}(k-i-1) > \tau(k-i-1)$. In this case, the relevant arcs of τ are as depicted in Figure 6.

-- Insert Figure 6 about here --

The dashed arc from $k-i$ to $k-i-1$ is a dummy arc that will cancel out at the end. This technique of adding dummy arcs is a very powerful tool, and it requires a bit of cleverness to determine which arcs are the correct ones to add. Condition (6) can be used to show that the collection of arcs in Figure 7 is no longer than the collection in Figure 6.

-- Insert Figure 7 about here --

Applying condition (11) to the collection of arcs in Figure 7 we get the desired result; that is, the solid arcs in Figure 8 are precisely the arcs that replace the solid arcs in Figure 6 depicting τ.

-- Insert Figure 8 about here --

The case $\tau(k-i-1) < \tau^{-1}(k-i-1)$ is left as an exercise to the reader. It is important to note that this transformation already has the property that no new bad pairs (k',i') are created where $k'-i' \geq k-i$.

Next consider case I(b). In this case, the elementary transformation is more complicated; the resulting tour is depicted Figure 9.

– Insert Figure 9 about here –

As above, it can be shown that the new tour is no longer than τ. However, it is not true that no new bad pairs (k',i') are created that have $k'-i' \geq k-i$. In fact, it is easy to see that $(k,i-\tau)$ has become a bad pair. However, as shown by the graph of the new tour, the essential structure of the tour is exactly as it was before that transformation, only the left slope is shorter. The result of repeated applications of similar transformations to τ is given in Figure 10,

– Insert Figure 10 about here –

and this new tour has length no greater than τ. This is the composite transformation to fix bad pairs in case I(b).

Next consider case I(c). In this case, the elementary transformation changes τ into the tour given in Figure 11.

– Insert Figure 11 about here –

Again, we must form a composite transformation. This is done is a manner very similar to the case above.

Finally, consider case II, $k-i-1 \in S(\tau,l)$, $l>k$; $k-i-1$ can either be an intermediate term on a "left slope" or a "right slope". The two cases and their elementary transformations are shown in Figure 12.

– Insert Figure 12 about here –

It is left as a somewhat tedious exercise to show that the necessary composite transformations can be formed, and that the resulting tours in these cases are indeed no longer than τ. □

Exercises

17. [Klyaus 1976] Suppose that

$$c_{ij} + c_{ji} \geq 0 ,$$
$$c_{ik} \geq c_{ij} + c_{jk} ,$$
$$c_{ki} \geq c_{ji} + c_{kj} ,$$

for all $i < j < k$.

(a) Prove that these conditions are a special case of the Demidenko conditions.

(b) Prove directly that there exists an optimal tour that is pyramidal. (*Hint:* Structure a proof similar to that of Theorem 10 of Section 7.)

18. Prove Lemma 7.

19. Complete the proof of Theorem 11. (This exercise should be attempted only by those who have a great deal of stamina. An interesting open problem is

to find a more elegant proof for this result.)

9. The Theory of Subtour Patching

In this section we shall consider the following strategy for finding an optimal tour: First find an optimal assignment φ. If φ is a tour, it is clearly an optimal tour, and we are done. Otherwise, it consists of several cycles or subtours. Modify φ so as to patch these subtours together to yield a single tour τ that is optimal. Thus our strategy is: given an *optimal* assignment φ, find a ψ such that $\varphi\psi$ is an *optimal* tour.

Our first task is to investigate conditions on ψ under which $\varphi\psi$ is a tour. We shall assume that the reader is somewhat familiar with the notion of a hypergraph. (A hypergraph is like an ordinary graph except that its "hyperedges" may be incident to arbitrary subsets of vertices, instead of to only subsets of size two.)

Let $P = \{\rho_1, \rho_2, \ldots, \rho_m\}$ be a set of (not necessarily disjoint) cycles on subsets of $V = \{1,2,\ldots,n\}$. Let $H = (V, P)$ be a hypergraph with vertex set V and with hyperedges corresponding to cycles in P. The hyperedge for ρ_i is incident to exactly those elements of V on which ρ_i acts. For example, if we have $\rho_1 = (2,3,4)$, $\rho_2 = (1,5)$, $\rho_3 = (1,3,2,4)$, then $H = (V, P)$ is as shown in Figure 13. Note that $\tau = \rho_1 \rho_2 \rho_3 = (1,2,3,4,5)$ is a tour.

– Insert Figure 13 about here –

A hypergraph $H = (V, P)$ is *disconnected* if it is possible to partition its vertex set V into two nonempty parts S and T such that no hyperedge is incident to a vertex in S and also to a vertex in T. A hypergraph is *connected* if it is not disconnected.

Theorem 12 If $\tau = \rho_1 \rho_2 \ldots \rho_m$ is a tour, then $H = (V, P)$ is connected.

Proof: Consider the contrapositive. Suppose $H = (V, P)$ is disconnected. Then there exist nonempty S and T such that $\rho_i(j) \in S$ if and only if $j \in S$, for all cycles ρ_i and cities j. Thus $\tau(j) \in S$ if and only if $j \in S$, where $\tau = \rho_1 \rho_2 \ldots \rho_m$, and τ cannot be a tour. □

Corollary 5 Let φ, ψ have factors φ_i, $i = 1,2,\ldots,r$, and ψ_j, $j = 1,2,\ldots,s$, respectively. If $\varphi\psi$ is a tour then $H = (V, \{\varphi_i\} \cup \{\psi_j\})$ is connected.

Corollary 5 gives a necessary condition on ψ for $\varphi\psi$ to be a tour. We now seek a sufficient condition.

A hypergraph $H = (V, P)$ with n vertices and m edges $\rho_1, \ldots, \rho_m$ is a *tree* if it is connected and if

$$\sum_{i=1}^{m} (|\rho_i| - 1) = n - 1 ,$$

where $|\rho_i|$ is the number of vertices incident to ρ_i. Note that this definition generalizes the well-known condition for a graph G to be a tree, i. e. G is

connected and $m = n - 1$. Also note that this definition allows a tree to have edges that are incident to a single vertex and that such edges (self loops) can be added to or deleted from the hypergraph without affecting its status as a tree. It is a straightforward exercise to show that if $H = (V,P)$ is connected, then

$$\sum_{i=1}^{m} (|\rho_i| - 1) \geq n - 1.$$

It follows that if $H = (V,P)$ is a tree, then the deletion of any edge ρ_i, with $|\rho_i| \geq 2$, disconnects H. Moreover, any edge ρ_i must be incident to exactly one vertex in each of the subtrees formed by its deletion. With these observations, it is not difficult to provide an inductive proof of the following.

Theorem 13 If $H = (V, P)$ is a tree, then $\tau = \rho_1 \rho_2 \ldots \rho_m$ is a tour (where the order in which the cycles ρ_i are multiplied to obtain τ is immaterial).

Corollary 6 Let φ, ψ have factors φ_i, $i = 1,2,\ldots,r$, and ψ_j, $j = 1,2,\ldots,s$, respectively. If $H = (V, \{\varphi_i\} \cup \{\psi_j\})$ is a tree then $\varphi\psi$ is a tour.

We can now reinterpret the procedure for upper triangular matrices presented in Section 5. The permutation ψ that is found is in the form of a single cycle acting on exactly one city in each subtour of φ. It follows that $H = (V, \{\varphi_i\} \cup \{\psi_j\})$ is a tree and, by Corollary 6, $\varphi\psi$ is a tour. The tour $\varphi\psi$ is optimal because $c(\varphi)$ is a lower bound on the length of a tour and $c(\varphi\psi) = c(\varphi)$. (Recall that in this case φ is not necessarily an optimal assignment, but is only optimal subject to the condition that $\varphi(n) = 1$; the fact that $c(\varphi)$ is a lower bound is a nontrivial result.)

The case of upper triangular matrices suggests that we investigate conditions under which there exists a ψ such that $H = (V, \{\varphi_i\} \cup \{\psi_j\})$ is a tree and $\varphi\psi$ is an optimal tour either of the ordinary or the bottleneck variety. For simplicity, when considering the bottleneck TSP, we shall always assume that $\bar{c}(\varphi) = 0$, in order to have $\bar{c}\,\varphi(\psi) = \bar{c}(\varphi\psi)$. (If this is not so, apply the transformation $c_{ij} := \max\{0,\ c_{ij} - k\}$, where k is the value of a bottleneck optimal assignment.)

Theorem 14 Let ψ have factors ψ_j, $j = 1,2,\ldots,s$. Then

$$c\,\varphi(\psi) = \sum_j c\,\varphi(\psi_j),$$

$$\bar{c}\,\varphi(\psi) = \max_j \{\bar{c}\,\varphi(\psi_j)\}.$$

Proof: Note that

$$\begin{aligned} c\,\varphi(\psi_j) &= c(\varphi\psi_j) - c(\varphi) \\ &= \sum_i c_{i,\varphi\psi_j(i)} - \sum_i c_{i,\varphi(i)} \\ &= \sum_{i:\psi_j(i)\neq i} c_{i,\varphi\psi_j(i)} - \sum_{i:\psi_j(i)\neq i} c_{i,\varphi(i)}. \end{aligned}$$

It follows that

$$\sum_j c\,\varphi(\psi_j) = \sum_{i\,\psi(i)\neq i} c_{i,\varphi\psi(i)} - \sum_{i\,\psi(i)\neq i} c_{i,\varphi(i)}$$

$$= \sum_i c_{i,\varphi\psi(i)} - \sum_i c_{i,\varphi(i)}$$

$$= c(\varphi\psi) - c(\varphi)$$

$$= c\,\varphi(\psi).$$

The proof for $\bar{c}\,\varphi(\psi)$ is similar. □

Theorem 14 tells us that we can deal with the factors of ψ independently of each other. This fact is useful, but by itself does not help us much in finding a ψ such that $\varphi\psi$ is an optimal tour. We shall adopt the approach of building up ψ as the product of transpositions (cycles of length two). Of necessity, these transpositions will generally not be factors of ψ (i. e. they will not be disjoint), so Theorem 14 will not apply.

As an example, suppose we have an eight-city problem, with $\varphi = (1,2,3), (4,5), (6,7), (8)$. The hypergraph $H = (V, \{\varphi_i\})$ is as shown in Figure 14(a). If we add edges for the transpositions (2,4), (5,7) and (7,8), we obtain the hypergraph tree shown in Figure 14(b). By Theorem 13, postmultiplication of φ by the transpositions (2,4), (5,7), (7,8), in any order, results is a tour. The order in which the transpositions are multiplied determines ψ. For example, we can have either

$$\psi = (2,4)\,(5,7)\,(7,8) = (2,4)\,(5,7,8)$$

or

$$\psi = (2,4)\,(7,8)\,(5,7) = (2,4)\,(5,8,7).$$

But no matter in what order the transpositions are multiplied, the cyclic factors ψ correspond to the connected components of the graph of transpositions, as shown in Figure 14(c).

– Insert Figure 14 about here –

With respect to a given optimal assignment φ, let us assign a *length* to each transposition (i,j):

$$c\,\varphi((i,j)) = c_{i\varphi(j)} + c_{j\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)},$$

or

$$\bar{c}\,\varphi((i,j)) = max\,\{c_{i\varphi(j)}, c_{j\varphi(i)}\}.$$

We can find a minimum length set of transpositions $\rho_1\rho_2 \ldots, \rho_t$ such that the hypergraph $H = (V, \{\varphi_i\} \cup \{\rho_j\})$ is a tree, by solving a minimum spanning tree

problem for the (multi)graph that is obtained by contracting each of the hyperedges of $H = (V, \{\varphi_i\})$. (Note that after contraction of the subtours (1,2,3), (4,5), (6,7) the transpositions (2,4), (5,7), (7,8) form a tree as shown in Figure 14(d).) We shall say that a set of transpositions is a *minimum spanning tree* (with respect to φ) if it is an optimal solution to such a spanning tree problem.

What relationship is there between the length of a minimum spanning tree and the length of an optimal tour? In the following two theorems we state conditions under which a minimum spanning tree yields a lower bound or an upper bound on the length of an optimal tour.

Theorem 15 Given an optimal assignment φ with respect to matrix C, suppose for any cyclic permutation ρ there exists a set T of transpositions connecting the same subtours in which there are cities on which ρ acts, such that $c\varphi(T) \le c\varphi(\rho)$ (or $\bar{c}\varphi(T) \le \bar{c}\varphi(\rho)$). Then if T is a minimum spanning tree, $c\varphi(T) \le c\varphi(\tau)$ (or $\bar{c}\varphi(T) \le \bar{c}\varphi(\tau)$), where τ is an optimal tour.

Proof: Let τ be an optimal tour and let $\psi = \varphi^{-1}\tau$ have factors $\psi_1, \ldots, \psi_s$. Then $c\varphi(\psi) = \sum c\varphi(\psi_j)$, by Theorem 14. By hypothesis, there exists a connecting set of transpositions T_j for each ψ_j, with $c\varphi(T_j) \le c\varphi(\psi_j)$. Moreover, there exists a tree $T \subseteq T_1 \cup T_2 \cup \ldots \cup T_s$ that spans all the subtours of φ with $c\varphi(T) \le c\varphi(\psi)$. The proof of the bottleneck case is similar. □

Theorem 16 Given an optimal assignment φ with respect to matrix C, suppose for any set T of transpositions there exists a cyclic permutation ρ, acting on the same cities acted on by the transpositions in T such that $c\varphi(\rho) \le c\varphi(T)$ (or $\bar{c}\varphi(\rho) \le c\varphi(T)$). Then if T is a minimum spanning tree, $c\varphi(T) \ge c\varphi(\tau)$ (or $\bar{c}\varphi(T) \ge \bar{c}\varphi(\tau)$), where τ is an optimal tour.

Proof: Let T be a minimum spanning tree. Consider the graph with vertex set V and edge set T (as in Figure 14(c)). Each connected component of this graph is a tree T_j and by hypothesis there exists a cyclic permutation ψ_j, acting on the same cities spanned by T_j, for which $c\varphi(\psi_j) \le c\varphi(T_j)$. By Corollary 6, $\varphi\psi$ is a tour, where the permutation ψ has the ψ_j as its factors. By Theorem 14, $c\varphi(\psi) = \sum_j c\varphi(\psi_j)$. Hence there exists a tour $\varphi\psi$ whose length is bounded from above by $c\varphi(T)$. The proof of the bottleneck case is similar. □

In the sections that follow we shall consider some special classes of matrices for which the hypotheses of Theorems 15 or 16, or both, are satisfied.

Exercise

20. Show that if $H = (V,P)$ is connected then

$$\sum_{i=1}^{m} (|\rho_i| - 1) \ge n - 1.$$

10. The Bottleneck TSP for Graded Matrices

We shall now apply the theory developed in the previous section to obtain an efficient algorithm for solving the bottleneck TSP for graded matrices. (Recall that in Section 6 we gave an algorithm for obtaining an approximate solution for the ordinary TSP for the same class of matrices.)

Let C be graded up its columns and let φ be a bottleneck optimal assignment. (Such an assignment can be found in $O(n^2)$ time; see Exercises.) Without loss of generality, assume

$$c_{ij} \geq \bar{c}(\varphi), \qquad \text{for all } i,j. \tag{12}$$

If (12) does not hold, then apply the transformation

$$c_{ij} := \max\{0, c_{ij} - \bar{c}(\varphi)\}.$$

(This transformation preserves grading.)

Suppose we now permute the columns of C into the order $\varphi(1), \varphi(2), \ldots, \varphi(n)$. Assuming that (12) holds, we now have a *permuted* upper triangular matrix C^{φ} that is graded up its columns. (Note the c_{ij} still refers to the (i,j)th entry of the original matrix; $c_{i\varphi(j)}$ designates the (i,j)th entry in the permuted matrix.) As an example, consider the permuted version of the matrix (3):

$$C^{\varphi} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 2 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix} \tag{13}$$

Here, as suggested by the listing of indices at the bottom of the matrix, $\varphi(1) = 6$, $\varphi(2) = 5$, ..., $\varphi(6) = 1$.

We first want to show that the lower bound property of Theorem 15 holds. We must show that, for any cyclic permutation ρ, there exists a tree T of transpositions, spanning the same subtours of φ in which there are cities on which ρ acts, such that $\bar{c}_{\varphi}(T) \leq \bar{c}_{\varphi}(\rho)$. So let ρ be an arbitrary cycle. Because C^{φ} is a permuted upper triangular matrix,

$$\bar{c}_{\varphi}(\rho) = \max\{c_{i,\varphi\rho(i)} \mid i < \rho(i)\}.$$

Because C is graded up its columns, we have for all $i \leq i' < \rho(i)$,

$$\bar{c}_{\varphi}((i',\rho(i))) = c_{i',\varphi\rho(i)} \leq c_{i,\varphi\rho(i)} = \bar{c}_{\varphi}((i,\rho(i))).$$

It follows that the set of transpositions

$$T = \{(i',\rho(i)) \mid i \le i' < \rho(i)\} \tag{14}$$

connects the cities acted on by ρ, with $\bar{c}\,\varphi(T) = \bar{c}\,\varphi(\rho)$. If this set of transpositions is not a tree then transpositions can be removed from the set to obtain a tree T satisfying the hypotheses of the theorem.

In our example (13), φ has three subtours: (1,6), (2,5), (3,4). Let $\rho = (1,6,2,4)$, with $\bar{c}\,\varphi(\rho) = \bar{c}\,\varphi((1,6)) = c_{11} = 5$. The set defined by (14) is

$$T = \{(1,6), (2,6), (3,6), (4,6), (5,6), (2,4), (3,4)\}.$$

From this set we can select, for example, (4,5) and (5,6) to obtain a tree. Note that $\bar{c}\,\varphi((4,5)) = c_{42} = 1$ and $\bar{c}\,\varphi((5,6)) = c_{52} = 0$, so the bottleneck length of this tree is unity.

Now let us show that the upper bound property of Theorem 16 holds. We must show that for any tree T of transpositions there exists a cyclic permutation ρ, acting on the same cities spanned by T, such that $\bar{c}\,\varphi(\rho) \le \bar{c}\,\varphi(T)$. Let us define a partial order "$\le$" on transpositions: $(i,j) \le (i',j')$ if $i' \le i$ and $j \le j'$, where $i < j$ and $i' < j'$. Now remove from T all transpositions that are not maximal with respect to the partial order. The remaining transpositions are of the form $(i_1,j_1),(i_2,j_2), \ldots, (i_r,j_r)$, where $i_1 < i_2 < \cdots < i_k$, and i_1, j_r are the cities of smallest and largest index spanned by T. Because the transpositions in T connect the cities on which they act, we must have $j_1 \ge i_2$, $j_2 \ge i_3$, ..., $j_{r-1} \ge i_r$. Let $k_1 > k_2 > \cdots > k_s$ be the cities different from $i_1, i_2, \ldots, i_r, j_1, j_2, \ldots, j_r$ that are acted on by transpositions in T.

For example, suppose $T = \{(1,3), (2,4), (3,7), (4,5), (5,6)\}$. The transpositions in T that are maximal with respect to "$\le$" are $(i_1, j_1) = (1,3)$, $(i_2, j_2) = (2,4)$, $(i_3, j_3) = (3,7)$, and $k_1 = 6$, $k_2 = 5$. Note that $j_1 = 3 > i_2 = 2$, $j_2 = 4 > i_3 = 3$.

We assert that if we take the sequence $(i_1, j_1, i_2, j_2, \ldots, i_r, j_r, k_1, \ldots, k_s)$, strike out the second occurrence of any index from within it, and treat the result as a cycle ρ, we have $\bar{c}\,\varphi(\rho) \le \bar{c}\,\varphi(T)$, as required. (The reader is asked to verify that this fact as an exercise.) In the case of our running example (13), suppose $T = \{(4,5), (5,6)\}$. Then we obtain $\rho = (4,5,6)$, with $\bar{c}\,\varphi(\rho) = 1 \le \bar{c}\,\varphi(T) = 1$.

The algorithm for solving the bottleneck TSP for graded matrices is as follows:

(1) Find a bottleneck optimal assignment φ. This requires $O(n^2)$ time.

(2) Determine the subtours of φ. This can be done in $O(n)$ time.

(3) Find a minimum spanning tree T of transpositions spanning the subtours of φ. This can be done in essentially $O(n^2)$ time by the current champion algorithm given [Galil & Gabow 1984].

(4) For each connected component T_j of the graph $G(V,T)$, find a cyclic permutation ψ_j, with $\bar{c}\,\varphi(\psi_j) \le \bar{c}\,\varphi(T_j)$, as described above. This requires at most $O(n \log n)$ time.

(5) Multiply φ by ψ in $O(n)$ time.

It is seen that the overall running time is essentially $O(n^2)$, the time required for the minimum spanning tree computation. As we shall see, this time

bound can be reduced for the more specialized case of permuted doubly graded matrices.

Recall that a matrix is ***doubly graded*** if it is both graded across its rows and up its columns, i.e. both $c_{ij} \leq c_{i,j+1}$ and $c_{ij} \geq c_{i+1,j}$.

Theorem 17 If C is doubly graded then a bottleneck optimal tour is given by the permutation $(1,2,\ldots,n-1,n)$.

Proof: Suppose we apply the nearest neighbor rule, starting at city 1. We shall show by induction that the path $1,2,\ldots,k$ contains no arc longer than the longest arc in a bottleneck optimal tour. Combining this result for $k=n$ with the fact that $(n,1)$ must be as short as any arc from n, we get the desired result. The basis is easy, since $(1,2)$ must be as short as any arc leaving 1. Suppose, by the inductive assumption, the path $1,2,\ldots,j$ contains no arc longer than the longest arc in a bottleneck optimal tour. Since arc $(j,j+1)$ is as short as any arc from the subset of cities $\{1,2,\ldots,j\}$ to the subset $\{j+1,\ldots,n\}$ the path may be extended to $j+1$. □

A matrix C is ***permuted doubly graded*** if there exists a permutation φ such that both $c_{i\varphi(j)} \leq c_{i\varphi(j+1)}$ and $c_{i\varphi(j)} \geq c_{i+1,\varphi(j)}$.

Theorem 18 If C is permuted doubly graded with respect to φ, then φ is a bottleneck optimal assignment.

Proof: Left as an exercise. □

The principal difference between the ordinary graded case, discussed above, and the permuted doubly graded case is that in the latter case there exists a minimum spanning tree composed of only transpositions of the form $(i,i+1)$. Since there are at most $n-1$ transpositions to consider the time bound for the spanning tree computation can be reduced to essentially $O(n)$.

In order to prove the assertion of the previous paragraph, it is sufficient to show that

$$\bar{c}\,\varphi(i',i'+1) \leq \bar{c}\,\varphi(i,j),$$

for all $i \leq i' < j$. We leave the details to the reader as an exercise.

Exercises

21. Show that it is possible to solve the bottleneck assignment problem for graded matrices in $O(n^2)$ time. (*Hint:* Assuming that C is graded up its columns, consider the following procedure. Find the smallest element in row 1. Suppose that this is in column k. Then cross out row 1 and column k and repeat.)
22. Justify the assertion that the sequence $(i_1, j_1, i_2, j_2, \ldots, i_r, j_r, k_1, \ldots, k_s)$, with the second occurrence of any index removed, yields a cycle ρ with $\bar{c}\,\varphi(\rho) \leq \bar{c}\,\varphi(T)$.
23. Note that the proof of Theorem 17 shows that the nearest neighbor rule constructs a bottleneck optimal tour if one starts from city 1. Show that

the nearest neighbor rule does not necessarily yield a bottleneck optimal tour if one starts from any other city.

24. Let C be graded up its columns and "contragraded" across its rows, i.e. both $c_{ij} \geq c_{i,j+1}$ and $c_{ij} \geq c_{i+1,j}$. Show that $(n, n-1, n-2, \ldots, 2, 1)$ is a bottleneck optimal tour.
25. Prove Theorem 18.

11. An Application: Cutting Wallpaper

The problem dealt with in this section was formulated and solved in the context of reading records from a rotating storage device [Fuller 1972]. A much more specialized version of this problem was dealt with in [Garfinkel 1977].

Suppose we are to cut $n-1$ sheets of wallpaper from a very long roll of stock with a pattern that repeats at intervals of one unit. Sheet i, $i = 1,2,\ldots,n-1$, starts at s_i (mod 1) and finishes at f_i (mod 1), with reference to the zero point of the pattern. If we cut sheet i from the roll immediately before sheet j, the intersheet waste is the distance from f_i to s_j, i. e.

$$c_{ij} = \begin{cases} s_j - f_i & \text{if } f_i \leq s_j, \\ 1 + s_j - f_i & \text{otherwise,} \end{cases} \qquad (15)$$
$$= s_j - f_i \pmod 1 .$$

Suppose the roll begins at the zero point on the pattern and after cutting our $n-1$ sheets from the roll we must make one more cut to restore the roll to the zero point. In other words, we must minimize the total intersheet waste, rounded up to the nearest pattern unit. To formulate this problem as a TSP, we introduce an nth dummy sheet with $s_n = f_n = 0$.

For example, suppose we wish to cut four sheets of wallpaper, with $s_1 = .1$, $s_2 = .8$, $s_3 = .6$, $s_4 = .4$ and $f_1 = .8$, $f_2 = .7$, $f_3 = .7$, $f_4 = .2$. After creating a dummy fifth sheet with $s_5 = f_5 = 0$, we obtain a five-city TSP with

$$C = \begin{bmatrix} .3 & 0 & .8 & .6 & .2 \\ .4 & .1 & .9 & .7 & .3 \\ .4 & .1 & .9 & .7 & .3 \\ .9 & .6 & .4 & .2 & .8 \\ .1 & .8 & .6 & .4 & 0 \end{bmatrix}.$$

Let us apply a linear admissible transformation to the matrix $C = (c_{ij})$ by adding f_i to row i and subtracting s_j from column j. The result is a (0, 1) matrix C', where

$$c'_{ij} = \begin{cases} 0 & \text{if } f_i \leq s_j, \\ 1 & \text{otherwise.} \end{cases} \qquad (16)$$

If the sheets are indexed so that $f_1 \geq f_2 \geq \ldots \geq f_n$, the matrix C' is graded up its columns. In the case of our example, we now have

$$C' = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For any tour τ, $c(\tau) = c(\tau') + \sum s_j - \sum f_i$. In other words, the cost of an optimal tour for the TSP for C' differs from the cost of an optimal tour for the TSP for C by a constant, $\sum s_j - \sum f_i$.

Since the matrix C' is graded up its columns, we can apply Theorem 9 and solve an assignment problem over C' to obtain an approximate solution to the TSP whose length differs from that of an optimal tour by no more than the value of the largest element in C', namely one unit. However, we can do better than this. In the following we shall show how to obtain a strictly optimal solution and, moreover, to obtain it in $O(n \log n)$ time.

There is a trick that simplifies matters and this involves changing the zero point of the pattern. Suppose we add a constant δ to each of the f_i and s_j values and redefine the problem in terms of f_i' and s_j' values, where

$$f_i' = f_i + \delta \pmod 1,$$
$$s_j' = s_j + \delta \pmod 1.$$

Such a translation does not affect the matrix C, as defined by (15), since

$$s_j' - f_i' = s_j - f_i \pmod 1.$$

However, this translation does change the matrix C' as defined in (16).

In the case of our example, if we take $\delta = .3$, we obtain $s_1' = .4$, $s_2' = .1$, $s_3' = .9$, $s_4' = .7$, $s_5' = .3$, and $f_1' = .1$, $f_2' = 0$, $f_3' = 0$, $f_4' = .5$, $f_5' = .3$. The matrix C' then becomes

$$C' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Suppose it is possible to find a δ such that C' can be made upper triangular and doubly graded after (independent) permutations of rows and columns. If such permutations exist, they can be effected by renumbering so that $f_1' \geq f_2' \geq \ldots \geq f_n'$ and then applying a permutation of φ to the columns so that $s'_{\varphi(1)} \geq s'_{\varphi(2)} \geq \ldots \geq s'_{\varphi(n)}$. Since the permuted matrix is upper triangular, we have $c'(\varphi) = 0$. Since C' is a (0,1)-matrix, it follows from Theorem 9 that an optimal tour τ is such that either $c'(\tau) = 0$ or $c'(\tau) = 1$. But the case $c'(\tau) = 0$ holds if and only if there is a tour with bottleneck length zero. And we know how to find a bottleneck optimal tour, in essentially $O(n)$ time, by the methods of the previous section.

Now all that remains is to show that it is always possible to find a δ such that the matrix C' becomes upper triangular and doubly graded after permutations of its rows and columns. This will be achieved if we can find a δ such that the largest s_j' is no smaller than any of the n f_i' values, the second largest s_i' is no smaller than $n-1$ of the f_i' values, and so forth.

To see that there is such a δ, we adapt a problem and its solution from [Lovász 1979, Problem 21, p. 27]. Suppose we are to walk around a circle on which there are n points f_i at which we are paid one dollar and n points s_j at which we must pay one dollar. Is there any point on the circle at which we can start with an empty wallet and never be financially embarrassed? And if so, how can we find this point?

The solution: Take a wallet full of money and start walking around the circle, starting at any point. Since we take in $\$n$ and give out $\$n$, we have the same amount of money in our wallet when we return to the starting point. Now remember where on the circle we had the least money. This was surely in an interval between an s_j and an f_i. Make that f_i our new starting point, i.e. set $\delta = -f_i$ and we shall have accomplished our objective.

To summarize the procedure for the wallpapering problem:

(1) Sort the f_i and s_i values, in $O(n \log n)$ time.

(2) Find the value δ and an optimal assignment φ in $O(n)$ time.

(3) Apply the algorithm for solving the bottleneck TSP for permuted doubly graded matrices. This requires essentially $O(n)$ time.

(4) If the bottleneck optimal tour found in (3) has zero length, it is an optimal solution to the problem. Else apply the approximation method of Theorem 9, to obtain an optimal tour. In this case, since an optimal assignment φ is already known, only $O(n)$ additional time is required.

The running time for solving the problem is dominated by the $O(n \log n)$ time required to sort the f_i and s_i values.

In the case of our example, an optimal solution is given by the tour (1, 4, 3, 2, 5), with a length of 1.5.

Exercise

26. We have formulated the wallpaper cutting problem with the objective of minimizing total waste, rounded up to the nearest pattern unit. To do this, we introduced a dummy sheet n, with $s_n = f_n = 0$. Now suppose we wish to minimize the absolute amount of waste. That is, the roll begins at the zero point of the pattern and we are charged for the total amount of paper used, regardless of where we make our final cut with reference to the zero point. To do this, let us replace the dummy job with a "pseudo-job" n with $f_n = 0$ and s_n being equal to the f_i of whatever job precedes it. Then

$$\begin{aligned} c_{in} &= s_n - f_i \\ &= f_i - f_i \\ &= 0, \\ c_{nj} &= s_j - f_n \\ &= s_j. \end{aligned}$$

Investigate what happens to coefficients c'_{in}, c'_{nj} for the pseudo-job when the pattern origin is translated. In particular, show that it is unnecessary to consider the pseudo-job in finding a new origin.

12. Permuted Distribution Matrices

We say that C is a *permuted* distribution matrix if there exists a permutation φ such that $C^{\varphi} = (c_{i\varphi(j)})$ is a distribution matrix. Recall that in Section 7 it was shown that the identity permutation is an optimal assignment for a distribution matrix; that is, the main diagonal of the matrix constitutes an optimal assignment. Therefore, if C is a permuted distribution matrix where C^{φ} is a distribution matrix, then φ is an optimal assignment for C. Throughout this section φ will be used to denote the optimal assignment for which C^{φ} is a distribution matrix.

In this section we show that permuted distribution matrices always have optimal tours that are of a specific form. In the following two sections we use this result to compute optimal tours for special cases of permuted distribution matrices.

Consider a permutation $\tau=\varphi\psi$ where φ is an optimal assignment. The assignment τ is said to be *basic* relative to φ if $H=(V,\{\varphi_i\}\cup\{\psi_i\})$ is a tree. Furthermore, τ is *pyramidal* with respect to φ if each of the factors of ψ is a pyramidal cycle. Finally, τ is *dense* with respect to φ if each of the subtours of ψ acts on a set of cities of the form $\{i,i+1,\ldots,i+k\}$.

The following theorem is the main result of this section.

Theorem 19 Let C be a permuted distribution matrix. Then C has an optimal tour $\tau=\varphi\psi$ where τ is basic, pyramidal, and dense with respect to φ.

This theorem will follow from a number of intermediate results about the structure of assignments and tours for permuted distribution matrices.

Lemma 9 Let C be a permuted distribution matrix. If $\tau=\varphi\psi$ is a permutation such that $H=(V,\{\varphi_i\}\cup\{\psi_i\})$ is connected, then there exists a permutation $\sigma=\varphi\rho$ where $H=(V,\{\varphi_i\}\cup\{\rho_i\})$ is connected, σ is pyramidal with respect to φ, and $c(\sigma)\leq c(\tau)$.

Proof: This result follows from Theorem 10, which states that any distribution matrix has a pyramidal optimal tour. Consider the distribution matrix C^{φ}; any submatrix of C^{φ} is also a distribution matrix. Furthermore, the cost $c\,\varphi(\psi_i)$ is the cost of the subtour ψ_i with respect to the matrix C^{φ}. This factor ψ_i is a tour for some submatrix of C^{φ}. Thus, by Theorem 10, there is some pyramidal tour ρ_i for this submatrix of C^{φ} with no greater cost. Therefore, $c\,\varphi(\rho_i)\leq c\,\varphi(\psi_i)$.

By repeating this procedure for each factor ψ_i we obtain a permutation $\rho=\prod_i\rho_i$ such that $\sigma=\varphi\rho$ is pyramidal with respect to φ, and $c\,\varphi(\rho)\leq c\,\varphi(\tau)$. Furthermore, since ρ_i and τ_i act on an identical set of cities, $H=(V,\{\varphi_i\}\cup\{\rho_i\}) = H=(V,\{\varphi_i\}\cup\{\psi_i\})$ which is connected. □

Throughout this section we will refer to the following example.

$$C = \begin{bmatrix} 8 & 24 & 16 & 32 & 40 & 56 & 48 & 64 \\ 7 & 21 & 14 & 28 & 35 & 49 & 42 & 56 \\ 6 & 18 & 12 & 24 & 30 & 42 & 36 & 48 \\ 5 & 15 & 10 & 20 & 25 & 35 & 30 & 40 \\ 4 & 12 & 8 & 16 & 20 & 28 & 24 & 32 \\ 3 & 9 & 6 & 12 & 15 & 21 & 18 & 24 \\ 2 & 6 & 4 & 8 & 10 & 14 & 12 & 16 \\ 1 & 3 & 2 & 4 & 5 & 7 & 6 & 8 \end{bmatrix}.$$

This has an optimal assignment $\varphi = (2,3)\ (6,7)$ and is generated by a density matrix $D = (d_{ij})$ where $d_{ij} = 1$ for all i,j. Consider the tour $\tau = \varphi\psi = (2,3)\ (6,7)\ (1,4,2,5)\ (3,7,8) = (1,4,3,6,7,8,2,5)$; $c(\tau) = 160$. The factor $(1,4,2,5)$ is not pyramidal. It does correspond to a tour in the submatrix of C^{φ},

$$\begin{bmatrix} 8 & 16^{*} & \mathbf{32} & 40 \\ 7 & 14 & 28 & \mathbf{35}^{*} \\ 5^{*} & \mathbf{10} & 20 & 25 \\ \mathbf{4} & 8 & 16^{*} & 20 \end{bmatrix}$$

where the bold-face entries indicate the costs used. Using the techniques described in Section 7, we find an optimal pyramidal tour for this matrix. This optimal tour, indicated by the starred entries corresponds to the factor $(1,2,5,4)$. Thus the transformation indicated in Lemma 9 yields the permutation $\tau_2 = (2,3)\ (6,7)\ (1,2,5,4)\ (3,7,8) = (1,3,6,7,8,2,5,4)$; $c(\tau_2) = 151$.

By the definition of a permuted distribution matrix, if C is a permuted distribution matrix, then for all $i<l$ and $j<m$

$$c_{i\varphi(m)}+c_{l\varphi(j)} \geq c_{i\varphi(j)}+c_{l\varphi(m)}. \tag{17}$$

Lemma 10 Let C be a permuted distribution matrix. If $\tau=\varphi\psi$ is a permutation that is pyramidal with respect to φ such that $H=(V,\{\varphi_i\}\cup\{\psi_i\})$ is connected, then there exists a permutation $\sigma=\varphi\rho$ that satisfies the following conditions:

(a) $H=(V,\{\varphi_i\}\cup\{\rho_i\})$ is connected;
(b) σ is pyramidal and dense with respect to φ; and
(c) $c(\sigma)\leq c(\tau)$.

Proof: Assume that $\tau=\varphi\psi$ is not dense with respect to φ. We first check whether there exist two factors of ψ, ψ_p and ψ_q, where the peaks of ψ_p and ψ_q are j_p and j_q, respectively, and the valleys of ψ_p and ψ_q are i_p and i_q, where $i_p<i_q<j_p<j_q$.

If there are two such overlapping factors, we shall show how to construct a permutation $\sigma=\varphi\rho$ where the factors of ρ are identical to ψ, with the exception of ψ_p and ψ_q; these factors have been combined into one new factor ψ_{pq} that acts on the set of cities that is precisely the union of the sets of cities acted on by ψ_p and ψ_q. Furthermore, $c\,\varphi(\psi_p\psi_q)\geq c\,\varphi(\psi_{pq})$.

First consider the simplest such case, where j_p is the only city on which ψ_p acts that is greater than i_q, and i_q is the only city on which ψ_q acts that is

smaller than j_p. Let $l=\psi_p^{-1}(j_p)$ and $m=\psi_q^{-1}(i_q)$; by our assumptions about ψ_p and ψ_q we know that $l<i_q<j_p<m$. Form $\psi_{pq}=\psi_p\psi_q(l,m)$; this patches the two pyramidal factors into one pyramidal factor. It is not hard to see that the cost of this patching operation, $c\,\varphi(\psi_{pq})-c\,\varphi(\psi_p\psi_q)$, is $c_{m\varphi(j_p)}+c_{l\varphi(i_q)}-c_{l\varphi(j_p)}-c_{m\varphi(i_q)}$, which is nonpositive, since C satisfies (17). Therefore, $\rho=(\prod_{i\neq p,q}\psi_i)\psi_{pq}$ satisfies the properties specified in the previous paragraph.

Next consider the case where ψ_p acts on some other city k where $j_p>k>i_q$. In this case we shall transform ψ_p and ψ_q into two new pyramidal factors ψ_p' and ψ_q' where j_p has been inserted into ψ_q (and has been deleted from ψ_p); furthermore, $c\,\varphi(\psi_p\psi_q)\geq c\,\varphi(\psi_p'\psi_q')$. By repeated use of this transformation, and the analogous one that inserts the valley of ψ_q into ψ_p, we eventually reach the easy situation dealt with above. At this stage, the two factors can be patched into one pyramidal factor that acts on all of the cities acted on by ψ_p and ψ_q.

Therefore, suppose that ψ_p acts on k, $j_p>k>i_q$. Let $j_1=\psi_p(j_p)$, $j_2=\psi_p^{-1}(j_p)$, and $\psi_p'=\psi_p(j_p,j_2)$; this deletes j_p from the factor ψ_p. Find the city of ψ_q, say l, such that $l>j_p$ but $m=\psi_q(l)<j_p$. (Since $i_q<j_p$ and $j_q>j_p$ there must be some such l.) Let $\psi_q'=\psi_q(j_p,l)$; this inserts j_p into ψ_q between l and m. The difference of the costs is

$$\begin{aligned} c\,\varphi(\psi_p\psi_q)-c\,\varphi(\psi_p'\psi_q') &= c_{j_2\varphi(j_p)}+c_{j_p\varphi(j_1)}-c_{j_2\varphi(j_1)}+c_{l\varphi(m)}-c_{l\varphi(j_j)}-c_{j_p\varphi(m)} \\ &= (c_{j_2\varphi(j_p)}+c_{j_p\varphi(j_1)}-c_{j_2\varphi(j_1)}-c_{j_p\varphi(j_p)})+(c_{j_p\varphi(j_p)}+c_{l\varphi(m)}-c_{l\varphi(j_p)}-c_{j_p\varphi(m)}). \end{aligned}$$

Since C satisfies (17), both parenthesized quantities are nonnegative. Notice that these transformations can be used obtain a permutation $\sigma=\varphi\rho$ where no two factors of ρ overlap in this interlaced manner, where σ is pyramidal with respect to φ and $H=(V,\{\varphi_i\}\cup\{\rho_i\})$ is connected.

Next consider two pyramidal factors ψ_p and ψ_q where the peaks of ψ_p and ψ_q are j_p and j_q, respectively, the valleys of ψ_p and ψ_q are i_p and i_q, respectively, and $i_p<i_q<j_q<j_p$. Two such nested factors can be transformed in much the same way as was done above, to form a new factor ψ_{pq} that acts on all of the cities contained in ψ_p and ψ_q, and $c\,\varphi(\psi_{pq})\leq c\,\varphi(\psi_p\psi_q)$. The essential idea is that the valley of ψ_q, i_q, can be deleted from that factor and inserted in the appropriate place into ψ_p while maintaining pyramidality, without increasing the cost. The details are exactly the same as the case above, and are left to the reader. By repeating this procedure, the two factors will be merged into one factor ψ_{pq}. As in the case above, it is easy to see that the required connectivity property is maintained as well.

As a result of the transformations mentioned above, we may assume without loss of generality that the factors ψ_r can be ordered so that the peak of ψ_r is less than the valley of ψ_{r+1}. If $\tau=\varphi\psi$ is not dense with respect to φ, then one factor ψ_r must act on cities i and j, but not on k, $i<k<j$. There must exist some city l acted on by ψ_r such that $l<k$ but $m=\psi_r(l)>k$. Let $\psi_r'=\psi_r(l,k)$; this inserts k in between l and k in ψ_i. The change in the costs,

$$\begin{aligned} c\,\varphi(\psi_r)-c\,\varphi(\psi_r') &= c\,(\varphi\psi_r)-c\,(\varphi\psi_r') \\ &= c_{k\varphi(k)}+c_{l\varphi(m)}-c_{k\varphi(m)}-c_{l\varphi(k)}, \end{aligned}$$

is nonnegative, since C satisfies (17). This transformation does not affect the desired connectivity and pyramidality properties. As a result, we can transform

$\tau=\varphi\psi$ into a permutation $\sigma=\varphi\rho$ where σ satisfies properties (a) through (c). □

Let us return to our example. The factors (1,2,5,4) and (3,7,8) overlap; in fact, both 4 and 5 are greater than 3, the valley of (3,7,8). As a result we first use the transformation that deletes 5 from (1,2,5,4) and inserts it into (3,7,8). This yields τ_3 = (2,3) (6,7) (1,2,4) (3,7,8,5) = (1,3,6,7,8,5,2,4); $c(\tau_3)$ = 142. Next we merge (1,2,4) and (3,7,8,5) into (1,2,4) (3,7,8,5) (2,5) = (1,2,3,7,8,5,4). Therefore τ_4 = (2,3) (6,7) (1,2,3,7,8,5,4) = (1,3,6,7,8,5,4) which is not a tour; $c(\tau_4)$ = 139. Note that although τ_2 and τ_3 are tours, Lemmas 9 and 10 do not insure that the resulting permutations will be tours. Finally, τ_4 is still not dense with respect to φ; the factor (1,2,3,7,8,5,4) omits 6. This is rectified by the last transformation (1,2,3,7,8,5,4) (3,6) = (1,2,3,6,7,8,5,4). So τ_5 = (2,3) (6,7) (1,2,3,6,7,8,5,4) = (1,3,7,8,5,4); $c(\tau_5)$ = 136.

A *(hyper)cycle* in a hypergraph $H=(V,\{\rho_i\})$ is an alternating sequence of hyperedges and vertices, $(\rho_0,i_1,\rho_1,i_2,\ldots,i_m,\rho_m)$ where $\rho_0=\rho_m$, but $\rho_j\neq\rho_k$ and $i_j\neq i_k$ for all $\{j,k\}\subset\{1,\ldots,m\}$ and $i_j\in\rho_j\cap\rho_{j+1}$ for all $j=1,\ldots,m$, $m>1$. Note that this is just an extension of the usual definition for a cycle in a graph.

Lemma 11 If a hypergraph is connected and is not a tree, then it contains a cycle.

Proof: Left as an exercise to the reader. □

Corollary 7 Let $\tau=\varphi\psi$ be a permutation that is dense and pyramidal with respect to φ and $H=(V,\{\varphi_i\}\cup\{\psi_i\})$ is connected. Then there exists a permutation $\sigma=\varphi\rho$ that is dense, pyramidal, and basic with respect to φ where $c(\sigma)\leq c(\tau)$.

Proof: Assume that τ is not basic with respect to φ. Since $H=(V,\{\varphi_i\}\cup\{\psi_i\})$ is not a tree, and it is connected, there must exist some cycle in $H=(V,\{\varphi_i\}\cup\{\psi_i\})$. Some hyperedge of this cycle must be a factor of ψ, say ψ_k; suppose that i and j are the cities (vertices) that link the hyperedge ψ_k in the cycle, where $i<j$. It is clear that either $\psi_k(i)>i$ or $\psi_k^{-1}(i)>i$; without loss of generality suppose that the former is true. Let $p=\psi_k(i)$ and let l be the city such that $l>i$ but $\psi_k(l)=m\leq i$. (Note that either p or l is $i+1$.) Then form $\psi_k'=\psi_k(i,l)$; ψ_k' has two pyramidal, dense factors ψ_{k1} and ψ_{k2} that together act on the same set of cities as the one factor ψ_k. Furthermore, the change in the costs,

$$c\varphi(\psi_k)-c\varphi(\psi_{k1}\psi_{k2})=c_{l\varphi(m)}+c_{i\varphi(p)}-c_{l\varphi(p)}-c_{i\varphi(m)}$$

which is nonnegative, since C satisfies (17). Most importantly, since ψ_k was part of a cycle, the transformation does not destroy the required connectivity constraint. In addition, the quantity $\sum(|\psi_i|-1)$ must decrease by 1 as result of this transformation. Therefore, after a sufficient number of such splitting operations, the resulting permutation ρ is such that $\sigma=\varphi\rho$ is basic with respect to φ. □

Let us return one last time to our example. The hypergraph $H=(V,\{\{2,3\},\{6,7\},\{1,2,3,6,7,8,5,4\}\})$ contains the cycle $(\{2,3\},3,\{1,2,3,6,7,8,5,4\},2,\{2,3\})$. We perform the splitting operation (1,2,3,6,7,8,5,4) (2,4) = (1,2) (3,6,7,8,5,4). So τ_6 = (2,3) (6,7) (1,2) (3,6,7,8,5,4) = (1,3,7,8,5,4,2) and $c(\tau_6)$ = 132. The

hypergraph $H = (V, \{\{2,3\}, \{6,7\}, \{1,2\}, \{3,6,7,8,5,4\}\})$ contains the cycle $(\{6,7\}, 6, \{3,6,7,8,5,4,\}, 7, \{6,7\})$. The factor (3,6,7,8,5,4) is split into (3,6,5,4) and (7,8) to give the tour τ_7 = (2,3), (6,7) (1,2), (3,6,5,4), (7,8) = (1,3,7,8,6,5,4,2) which is basic, pyramidal and dense with respect to (2,3) (6,7). Furthermore $c(\tau_7) = 128$.

By combining Lemma 9, Lemma 10 and Corollary 7 we get the desired result that every permuted distribution matrix has an optimal tour that is basic, dense, and pyramidal with respect to φ.

Exercises

27. Prove Lemma 11.
28. Construct a polynomial-time algorithm to determine if C is a permuted distribution matrix; if C is a permuted distribution matrix the algorithm should output the corresponding density matrix and permutation φ.

13. An Application: Sequencing a Single State-Variable Machine

A certain factory manufactures specialty refractory products. There are $n-1$ different jobs that are ready for burning in the kiln. Job i requires a starting temperature s_i and after some prescribed variations in temperature (over which we have no control) the job is finished at temperature f_i. If job j immediately follows job i in the kiln, the temperature must be changed from f_i to s_j. The cost of changing the kiln temperature between jobs i and j is

$$c_{ij} = \begin{cases} \int_{f_i}^{s_j} f(x)dx & \text{if } f_i \le s_j, \\ \int_{s_j}^{f_i} g(x)dx & \text{if } s_j \le f_i. \end{cases} \tag{18}$$

where f and g are cost density functions. It is natural that f and g should be different functions, since the cost density for raising the temperature of the kiln is probably quite different from that for lowering it. Neither f nor g need be strictly nonnegative, but it conforms to reality to require that

$$f(x) + g(x) \ge 0$$

for all x. (Else one could cycle the temperature of the kiln up and down and make money without manufacturing anything.)

Our objective, of course, is to prescribe a sequence for the jobs that minimizes the total cost of changing the temperature of the kiln between jobs. In order to properly set this problem, we must specify an initial temperature f_n at which we find the kiln and a final temperature s_n at which we must leave it. We let s_n and f_n prescribe an nth dummy job and we have a proper TSP.

Let us index the jobs (which we hereafter call cities) so that $f_1 \le f_2 \le \ldots f_n$. Let φ be a permutation such that $s_{\varphi(1)} \le s_{\varphi(2)} \le \ldots \le s_{\varphi(n)}$. We assert that when the columns of C have been permuted in this way, C^{φ} is equivalent by a linear admissible transformation to a distribution matrix. That is, for $1 \le i \le n-1$ and $2 \le j \le n$,

$$d_{ij}^{\varphi} = d_{i\varphi(j)} = c_{i\varphi(j)} + c_{i+1,\varphi(j-1)} - c_{i\varphi(j-1)} - c_{i+1,\varphi(j)} \ge 0.$$

(Recall Exercise 13, Section 7.) Specifically, for C defined by (18) we have

$$d_{i\varphi(j)} = \int_a^b [f(x) + g(x)]\,dx \ge 0,$$

where $a = \max\{f_i, s_{\varphi(j-1)}\}$, $b = \min\{f_{i+1}, s_{\varphi(j)}\}$ and $d_{i,\varphi(j)} = 0$ if $a > b$.

Because C^{φ} is a distribution matrix, it follows from the results of the previous section that there exists an optimal tour $\varphi\psi$, where ψ is minimal, dense, and pyramidal. Moreover, the lower bound property is satisfied and there exists a minimal spanning tree composed of transpositions of the form $(i, i+1)$. In order to obtain an efficient algorithm, we must now establish the upper bound property for such minimal spanning trees.

Our task quite simply is the following. Show that for all i, j, $i < j$, there exists a pyramidal cyclic permutation ψ acting on cities $i, i+1, \ldots, j$ such that

$$c\,\varphi(\psi) = \sum_{h=i}^{j-1} c\,\varphi((h, h+1)) = \sum_{h=i}^{j-1} d_{h\varphi(h+1)}. \tag{19}$$

Our proof is by induction on the value of $j - i$. For $j - i = 1$, we have $c\,\varphi((i, i+1)) = d_{i\varphi(i+1)}$, so $\psi = (i, i+1)$. So assume there exists a pyramidal cycle ψ acting on $i, i+1, \ldots, j$ that satisfies (19). We shall show that this implies the existence of such a ψ' on $i, i+1, \ldots, j+1$.

Suppose we let $\psi' = (j, j+1)\,\psi$. Then city $j+1$ is inserted between j and its immediate predecessor $\psi^{-1}(j)$ in ψ. Let $\psi^{-1}(j) = k$ and we have

$$\begin{aligned} c\,\varphi((j, j+1)\psi) &= c\,\varphi(\psi) + c_{k\varphi(j+1)} + c_{j+1,\varphi(j)} - c_{k\varphi(j)} - c_{j+1,\varphi(j+1)} \\ &= c\,\varphi(\psi) + d_{j\varphi(j+1)} + \sum_{h=k}^{j-1} d_{h\varphi(j+1)}. \end{aligned}$$

On the other hand, if we let $\psi = \varphi(j, j+1)$, then city $j+1$ is inserted between j and its immediate successor $\psi(j)$ in ψ. Let $\psi(j) = k$ and we have

$$\begin{aligned} c\,\varphi(\psi(j, j+1)) &= c\,\varphi(\psi) + c_{j\varphi(j+1)} + c_{j+1,\varphi(k)} - c_{j\varphi(k)} - c_{j+1,\varphi(j+1)}. \\ &= c\,\varphi(\psi) + d_{j\varphi(j+1)} + \sum_{h=k+1}^{j} d_{j\varphi(h)}. \end{aligned}$$

But for $1 \le h \le j - 1$ we have

$$d_{h\varphi(j+1)} = \int_a^b (f(x) + g(x))dx ,$$

where $a = \max\{f_h, s_{\varphi(j)}\}$, $b = \min\{f_{h+1}, s_{\varphi(j+1)}\}$. And for $2 \le h \le j$ we have

$$d_{j\varphi(h)} = \int_{a'}^{b'} (f(x) + g(x))dx ,$$

where $a' = \max\{f_j, s_{\varphi(h-1)}\}$, $b' = \min\{f_{j+1}, s_{\varphi(h)}\}$.

There are two possibilities:

(1) $f_j \le s_{\varphi(j)}$, in which case $f_{h+1} \le s_{\varphi(j)}$, $a \ge b$, and

$$d_{h\varphi(j+1)} = 0, \quad \text{for } 1 \le h \le j - 1;$$

(2) $f_j > s_{\varphi(j)}$, in which case $f_j > s_{\varphi(h)}$, $a' \ge b'$ and

$$d_{j\varphi(h)} = 0, \quad \text{for } 2 \le h \le j .$$

Thus if $f_j \le s_{\varphi(j)}$, the desired permutation is $\psi' = (j, j+1)\psi$, whereas if $f_j > s_{\varphi(j)}$ we want $\psi' = \psi(j, j+1)$. We have thus proved the following.

Theorem 20 Let C^* be a distribution matrix defined by (18). For any i, j, $i < j$, let $i \le i(1) < i(2) < \ldots < i(r) \le j - 1$ be such that $f_{i(h)} \le s_{\varphi(i(h))}$, $1 \le h \le r$, and let $i \le j(1) < j(2) < \ldots < j(s) \le j - 1$ be such that $f_{j(h)} > s_{\varphi(j(h))}$, $1 \le h \le s$. Then

$$\psi = (i(r), i(r)+1)\,(i(r-1), i(r-1)+1) \ldots (i(1), i(1)+1)$$
$$(j(1), j(1)+1) \ldots (j(s-1), j(s-1)+1)\,(j(s), j(s)+1)$$

is a pyramidal cyclic permutation acting on $i, i+1, \ldots, j$, with

$$c\,\varphi(\psi) = \sum_{k=i}^{j-1} d_{k\varphi(k+1)} .$$

We are now prepared to state the algorithm for solving the single state-variable machine sequence problem [Gilmore & Gomory 1964]:

(1) Sort the s_i, f_i values so that $f_1 \le f_2 \le \ldots \le f_n$, $s_{\varphi(1)} \le s_{\varphi(2)} \le \ldots \le s_{\varphi(n)}$. This can be done in $O(n \log n)$ time.

(2) Find the subtours of the optimal assignment φ. This requires only $O(n)$ time.

(3) Find a minimum length tree T of $(i, i+1)$ transpositions spanning the subtours of φ. This requires computation of the values $d_{i\varphi(i+1)}$, which we assume can be done in $O(n)$ time. A minimum spanning tree can be found in essentially $O(n)$ time.

(4) For each connected component T_j of the graph $G(V, T)$, find a pyramidal cyclic permutation as indicated by Theorem 20 and thereby find ψ. This requires only $O(n)$ time.

(5) Multiply φ by ψ to obtain an optimal tour. $O(n)$ time.

Exercises

29. (a) You are given $n-1$ trapezoids, where each trapezoid j is specified by two parameters, a_j and b_j, $j = 1,2,\ldots,n-1$, as shown in Figure 15(a). The object is to arrange these trapezoids in a line, no two of them overlapping (a typical feasible arrangement is shown in Figure 15(b)), so that the total length of the arrangement is as short as possible. Show that this problem can be formulated and solved as an n-city TSP of the Gilmore-Gomory type, with a distance matrix of the form (18).
(b) Consider the following scheduling problem. There are $n-1$ jobs to be worked on by each of two machines. Job j requires a_j units of processing on the first machine and b_j units on the second. As soon as the processing of job j is completed on the first machine, its processing must begin on the second machine. (There is no buffering of jobs possible between the two machines.) The object is to find a sequence for the jobs such that all the jobs will be completed as early as possible. Verify that the trapezoid problem of part (a) is an appropriate model for this problem. (Note: the three-machine generalization of this problem is $\underline{\underline{\text{NP}}}$-hard [Röck 1984; Chapter 3]).

-- Insert Figure 15 about here --

30. [Gilmore & Gomory 1964]. Let τ be an optimal tour for a problem instance with density functions f and g. Prove that τ is also an optimal tour for any problem instance with functions f' and g' where $f(x) + g(x) = f'(x) + g'(x)$. (The parameters f_i, s_j remain unchanged.)

31. [Gilmore & Gomory 1964]. Suppose that a number of jobs are to be sequenced on a machine with costs related to a single state-variable x. Assume that $g(x) = 0$. Prove the following:

(a) If the initial value of the state-variable must be f_0 but its final value is unrestricted, then the sequencing problem is equivalent to a TSP with an additional job 0 with starting value s_0, where $s_0 \leq \min_{1 \leq i \leq n} \{s_i\}$ and final value f_0.

(b) If any value of the state-variable is available at the beginning, but the final value must be s_0, then the sequencing problem is equivalent to a TSP with an additional job 0 with starting value s_0 and final value f_0 where $f_0 \geq \max_{1 \leq i \leq n} \{f_i\}$.

(c) If any value of the state-variable is available at the beginning and its final value is unrestricted, the sequencing problem is equivalent to a

TSP with an additional job 0 with

$$s_0 \leq \min_{1 \leq i \leq n} \{f_i\},$$

$$f_0 \geq \max_{1 \leq i \leq n} \{s_i\}.$$

32. Modify the algorithm to solve the bottleneck version of the Gilmore-Gomory TSP, subject to the assumption that $f(x) \geq 0$ and $g(x) = 0$.

33. Construct an example to show that a bottleneck optimal tour is not necessarily a shortest tour, for the Gilmore-Gomory TSP. Show that in the case $s_j \geq f_{\varphi(i)}$, for all i, there is a tour that is optimal with respect to both criteria. (Assume $f(x) \geq 0$, $g(x) = 0$.)

14. The TSP for Product Matrices

An $n \times n$ matrix $C=(c_{ij})$ is called a *product* matrix if there exists two n-dimensional vectors a and b such that $c_{ij}=a_i b_j$ for all i,j. Recall that an $n \times n$ matrix C is called a *permuted distribution* matrix if there exist a nonnegative matrix D, and a permutation φ, such that $c_{i\varphi(j)} = \sum_{k=i}^{n} \sum_{l=1}^{j} d_{kl}$. It is a straightforward exercise to show that all product matrices can be transformed into a permuted distribution matrix by a linear admissible transformation. In this section, we present negative and positive results for product matrices. We show that, in general, computing an optimal tour is NP-hard, but for many special cases, including symmetric product matrices, there exists a polynomial-time algorithm.

Theorem 21 [Sarvanov 1980] The TSP restricted to product matrices is NP-hard.

To prove this result, an intermediate problem is introduced. Before this can be done, some additional terminology must be defined. Let $\Pi=\{N_1,\ldots,N_m\}$ be a partition of $N = \{1,\ldots,n\}$. For any partition Π, let the *spine graph* of the partition be $G(\Pi)=(N,E)$ where $E=\{\{i,i+1\},\ i \in N_k,\ i+1 \in N_l,\ k \neq l\}$. Let $G'=(N,E')$ be a subgraph of G. The vertex set of G' is the entire vertex set of G. Define the *partition graph* of G' to be $P(G')=(\Pi,E_P)$ where $E_P = \{\{N_k,N_l\} \mid$ there exists an i such that $i \in N_k,\ i+1 \in N_l,\ \{i,i+1\} \in E'\}$. All of these graphs are undirected. As an example, consider $\Pi = \{\{1,4\}, \{2,3\}, \{5\}\}$. The corresponding spine graph $G(\Pi)$ is given in Figure 16(a). For the subgraph G' shown in Figure 16(b), the corresponding partition graph $P(G')$ is given in Figure 16(c). In addition, call G' a *matching* if the degree of every vertex is 0 or 1.

-- Insert Figure 16 about here --

Consider the following decision problem.

PARTITION GRAPH SPANNING TREE

INSTANCE: $N = \{1,\ldots,n\}$ and $\Pi = \{N_1,\ldots,N_m\}$, a partition of N.

QUESTION: If $G(\Pi)=(N,E)$ is the spine graph of Π, does there exist a

subgraph $G'=(N,E')$ of $G(\Pi)$ that is a matching, such that the partition graph $P(G') = (\Pi, E_P)$ is a spanning tree of the vertex set Π?

A matching is said to be *good* if its partition graph is a spanning tree.

Notice that there is a connection between the tools that we have defined for this problem and those that were defined for the theory of subtour patching. The fact that a given set of $\{i,i+1\}$ edges form a spanning tree of $P(G')$ implies that those edges, together with the hyperedges induced by the partition (which corresponds to a permutation φ) form a spanning (hypergraph) tree. Furthermore, the fact that the $\{i,i+1\}$ edges form a matching in the spine graph implies that they could be factors of ψ and therefore their costs are independent of each other.

Lemma 12 PARTITION GRAPH SPANNING TREE is NP-complete.

Proof: It is easy to verify that PARTITION GRAPH SPANNING TREE is in NP. To show that it is NP-hard, we reduce from the Hamiltonian path problem for cubic graphs (see Chapter 3). Recall that a cubic graph is a graph where the degree of every vertex is three. Given a cubic graph $G=(V,E)$ we will construct an instance of PARTITION GRAPH SPANNING TREE. Note that $|V|=n$ must be even and that $|E|=3n/2$. The set N constructed is $\{1,2,\ldots,9n\}$ and there will be $3n$ parts in the partition; n each of types 1, 2, and 3 as we shall define below. For each part N_{ij}, $i=1,2,3$, $j=1,\ldots,n$, the second index may be thought of as corresponding to a vertex in V. From G, construct a multigraph $G^+=(V,E^+)$ where $E^+=E\cup\{\{1,2\},\{3,4\},\ldots,\{n-1,n\}\}$. The $n/2$ new edges will remain distinguished throughout this construction. Since the degree of every vertex of G^+ is 4, it is Eulerian, so an Eulerian tour $T=\{e_1,e_2,\ldots,e_{2n}\}$ can be constructed in polynomial time. The edges e_i are now oriented in the direction of the tour. The partition $\Pi=\{N_{ij} \mid i=1,2,3,\ j=1,\ldots,n\}$ is constructed by "following" the Eulerian tour. It is easiest to understand the construction in terms of the corresponding spine graph. To differentiate between vertices of G and vertices of the spine graph, those of the spine graph will be called *elements*. Suppose $e_1=(v,w)$ and is not a distinguished edge; in this case the initial portion of the resulting spine graph is shown in Figure 17(a). If (v,w) is distinguished, then the alternate construction is shown in Figure 17(b).

-- Insert Figure 17 about here --

In general, for any edge e_i, the next six elements will be used if it is distinguished and four otherwise. Furthermore, the same construction as above will be used. Suppose that $j-1$ elements have been used before reaching $e_i=(v,w)$. Then add the construction shown in either Figure 18(a) or 18(b) depending on whether e_i is undistinguished or distinguished, respectively.

-- Insert Figure 18 about here --

Every N_{1v} and N_{2v} contains four elements, whereas N_{3v} contains only one. Since the tail vertex of e_i is equal to the head vertex of e_{i-1}, the spine has $2n$ components, each of which corresponds to an edge of the tour. Now we must show that a suitable matching can be found if and only if G has a Hamiltonian path.

Suppose that $\{v_1,v_2,\ldots,v_n\}$ is Hamiltonian path of G. For every distinguished edge, include in the matching the marked edges shown in Figure 19(a). For every edge in the Hamiltonian path, choose the edges as shown in Figure 19(b).

-- Insert Figure 19 about here --

There are $\frac{n}{2}+1$ components of the spine that have not yet been marked. By a simple counting argument, these components contain elements that belong to every N_{1v} and N_{2v}. All but one of the components will be marked as shown in Figure 20(a), and the last will be marked as depicted in either Figure 20(b) or 20(c). It is easy to see that the corresponding partition graph is the spanning tree given in Figure 21.

-- Insert Figures 20 and 21 about here --

Suppose that there exists a good matching. It will be shown that the spanning tree must be of the above form. Consider the vertices of the partition graph of type 3. Each set N_{3v} contains only one element; thus there are only two possible sets that the vertex N_{3v} could be adjacent to in the spanning tree. Consider the component of the spine in which this element appears (see Figure 22).

-- Insert Figure 22 about here --

If the edge between the N_{3v} and N_{3w} elements were in the matching, the corresponding edge in the partition graph must be a separate component, and therefore could not be in the spanning tree. Thus, for every distinguished component of the spine $\{v,w\}$, the matching must contain the edges between the N_{2v}, N_{3v} and N_{2w}, N_{3w} elements, as shown in Figure 19(a). Similarly, by examining the structure of the spine, it is easy to see that any N_{1v} must be adjacent only to the corresponding N_{2v} vertex in the spanning tree. Thus, the structure of the spanning tree is forced to be as depicted in Figure 23, and the tree is connected by forming a spanning tree on the vertices of type 2.

-- Insert Figure 23 about here --

But what form can the tree take? In creating the edges shown, a maximal set of edges from the distinguished components of the spine has been forced to be in the matching. Thus all the $\{N_{1v},N_{2v}\}$ edges have been taken from undistinguished components. Since the undistinguished components correspond to edges of G, each N_{2v} contains exactly three elements from these components. To form the $\{N_{1v},N_{2v}\}$ edges, one of these has been used. Thus, for each v there are at most two available elements in the spine that are contained in N_{2v}. Therefore, in the spanning tree on vertices of type 2 in the partition graph, these vertices must all have degree at most 2. But the only spanning tree with each degree 1 or 2 is a Hamiltonian path. Thus the spanning tree must be of the form claimed above. For each edge $\{N_{2v},N_{2w}\}$ in the Hamiltonian path, there exists an edge $\{v,w\}$ in the original graph G. Therefore, G has a Hamiltonian path. □

Lemma 13 PARTITION GRAPH SPANNING TREE α TSP for product matrices

Proof: Given an instance of PARTITION GRAPH SPANNING TREE, we must construct a corresponding instance of the TSP for product matrices; that is, we must construct a product matrix C and a bound K such that C has a tour of length at most K if and only if the instance of PARTITION GRAPH SPANNING TREE admits a good matching.

Let $\Pi = \{N_1, \cdots, N_m\}$ be the partition of an instance of PARTITION GRAPH SPANNING TREE. Let φ be any permutation with the same cycle structure as Π. For example, if $\Pi = \{\{1,4\},\{2,3\},\{5\}\}$, then $\varphi=(1,4)(2,3)(5)$. In general there may be many such permutations. The matrix C is given by a and b where $c_{ij}=a_i b_j$ and $a_i=n-i+1$ and $b_{\varphi(j)}=j$. The bound for this instance of the TSP is $K = \sum_{i=1}^{n} i(n-i+1) + m - 1$.

Since C is a permuted distribution matrix, an optimal assignment can be specified as the assignment that permutes the columns to yield a distribution matrix. It is not hard to see that φ is this permutation for C. This assignment has the cycle structure of Π and these cycles must be patched together to form an optimal TSP tour. The cost of the optimal assignment will always be $\sum_{i=1}^{n} i(n-1+1)$.

Suppose that there exists a good matching, $E=\{e_1,\ldots,e_{m-1}\}$. If $e_i=\{j,j+1\}$, let ψ_i be the interchange $(j\ j+1)$. It follows that $c\,\varphi(\psi_i)=1$. Since the partition graph generated by E is a spanning tree, it follows that $\tau = \varphi\psi=\varphi\psi_1\psi_2 \cdots \psi_{m-1}$ is a Hamiltonian cycle. Furthermore, since the interchanges ψ_i are disjoint, the cost of this tour is precisely K.

Now suppose that there is a tour τ with $c(\tau)\le K$. Since C is a permuted distribution matrix, there must exist a tour τ' that is pyramidal and dense relative to φ with $c(\tau') \le K$. Therefore, assume that $\tau=\varphi\psi=\varphi\psi_1\psi_2 \cdots \psi_p$ where the ψ_i are dense and pyramidal.

We will show that if ρ is not an $(i,i+1)$ transposition then $c\,\varphi(\rho)\ge|\rho|$. The proof is by induction on $|\rho|$. For the basis of the proof consider $|\rho|=3$. Since there are only two different dense pyramidal tours on three elements it is simple to verify this case, and we leave it as an exercise to the reader. Next, we complete the induction by showing that if there exists a dense pyramidal permutation with $c\,\varphi(\rho)<|\rho|=k+1$ then there exists a dense pyramidal permutation ρ' with $c\,\varphi(\rho')<|\rho'|=k$. Suppose that ρ is a dense pyramidal permutation on the cities $\{i,i+1,\ldots,i+k\}$ such that $c\,\varphi(\rho)<k+1$. Suppose that $\rho(i+k)=i+j$ and that $\rho^{-1}(i+k)=i+l$. Let $\rho'=\rho(i+l,i+k)$; ρ' is simply the cycle formed by deleting $i+k$. A straightforward computation gives that $c\,\varphi(\rho)-c\,\varphi(\rho')=(k-j)(k-l)\ge1$ (actually it is at least 2) which shows that $c\,\varphi(\rho')<|\rho'|=k$. This completes the proof that $c\,\varphi(\rho)\ge|\rho|$ for $|\rho|\ge3$. For transpositions ρ not of the form $(i,i+1)$ it is easy to see that $c\,\varphi(\rho)\ge2$. Thus we have proved the claim that for all factors ρ that are not $(i,i+1)$ transpositions, $c\,\varphi(\rho)\ge|\rho|$.

Furthermore, since τ is a tour it follows from the fact that $H = (N,\{\varphi_i\}\cup\{\psi_i\})$ must be connected, that $\sum_{i=1}^{p} |\psi_i| \ge m-1+l$. However, since the cost of the tour is at most K, $c\,\varphi(\psi)\le m-1$. Suppose that q of the ψ_i are $(j,j+1)$ transpositions. Then we find that

$$m-1 \ge c\,\varphi(\psi) = \sum_{i=1}^{p} c\,\varphi(\psi_i) \ge \sum_{i=1}^{p} |\psi_i| - q \ge m-1+p-q.$$

Therefore, $p=q$ and all of the factors of ψ are $(i,i+1)$ transpositions. These transpositions must correspond to a matching that generates a spanning tree of the partition graph. □

This completes the proof of Theorem 21.

In the remainder of this section we turn to some more positive results for special classes of product matrices. The result presented here shows that there is a broad class of product matrices for which there is a polynomial-time algorithm.

Let φ be an arbitrary permutation and let Π be the corresponding partition of $\{1,\ldots,n\}$. For simplicity of notation, let G_φ denote $P(G(\Pi))$, the partition graph of the spine graph of Π.

Theorem 22 Let C be a permuted distribution matrix, and let φ be the optimal assignment where C^φ is a distribution matrix. If G_φ is a tree, then there is a polynomial-time algorithm to find an optimal tour for C.

Proof: Left to the reader as an exercise. □

Corollary 8 [Gaikov 1980] Let C be a product matrix such that $c_{ij}=a_i b_j$ and $a_1 \le a_2 \le \cdots \le a_n$. Let φ be the optimal assignment. If G_φ is a tree, then there is a polynomial-time algorithm to find an optimal tour for C.

The Soviet literature contains a great number of papers with results that were superseded by Corollary 8. An easy corollary to this result is the case of the symmetric product matrices; that is $c_{ij}=a_i b_j$ and $c_{ij}=c_{ji}$.

Corollary 9 For a symmetric product matrix C where $c_{ij}=a_i b_j$ where $a_1 \le a_2 \le \cdots \le a_n$, there exists a polynomial-time algorithm to find an optimal tour.

Proof: As noted above, an optimal assignment φ is given by the permutation that sorts the b's; that is $b_{\varphi(1)} \ge b_{\varphi(2)} \ge \ldots \ge b_{\varphi(n)}$. Since C is symmetric, $a_i b_j = b_i a_j$ for all i,j. Equivalently,

$$\frac{a_i}{b_i} = \frac{a_j}{b_j} = \lambda.$$

Thus, $a_i=\lambda b_i$. Therefore, φ is $(1)(2)\ldots(n)$ or $(1,n)(2,n-1)\ldots(\lfloor\frac{n}{2}\rfloor, \lceil\frac{n}{2}\rceil)$ depending on whether λ is negative or positive, respectively. In either case, G_τ is a path. □

Note that this shows that there are cases where the symmetric case of the TSP is provably easier than the asymmetric case, of course under the assumption that $\underline{\underline{P}} \neq \underline{\underline{NP}}$.

Exercises

34. Construct a polynomial-time algorithm that either constructs a and b, or shows that C is not a product matrix.

35. Prove Theorem 22. *Hint:* The following observations may be useful in constructing a polynomial-time algorithm based on dynamic programming.
   (a) There exists an optimal tour $\tau=\varphi\psi$ that is basic, pyramidal and dense with respect to φ (see Section 12).
   (b) Given the set of cities on which a factor of ψ acts, it is possible to compute the factor in polynomial time (see Section 7).
   (c) Since more than one $(i,i+1)$ transposition may support the existence of a particular edge in the tree G_φ, it may be convenient to view G_φ as a "multitree". Note however, that exactly one factor of ψ will contain both endpoints of a set of multiedges of the tree.
   (d) Every factor of ψ corresponds to a path in G_φ. Determining the factors of ψ amounts to covering the edges of G_φ with paths. (Note that the paths will be edge-disjoint, but not vertex-disjoint.)
   (e) Since the costs $c\,\varphi(\psi_i)$ and $c\,\varphi(\psi_j)$ are independent for disjoint factors, subproblems for edge-disjoint trees can be solved independently.
   (f) Root G_φ at an arbitrary node. For each node v of G_φ, at most one edge to a child of v is contained on the path that includes v and the parent of v. Suppose that the edge to the child u is used on that path. Then the path covering problem for the subtree rooted at v with the subtree rooted at u deleted can be solved as an independent subproblem.
   (g) Suppose that v is the endpoint of the path that includes its parent. (This is, in effect, the case of the root of G_φ as well.) The edge to each child of v must be covered by some path, and some (possibly more than one) of these paths pass through v. Thus, there is a matching problem among the children of v to determine which children (if any) get paired on a path through v.

15. Bandwidth Limited Networks

Let A be the adjacency matrix of a digraph G. That is,

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is an edge,} \\ 0 & \text{otherwise.} \end{cases}$$

It is customary to say that A and G have *bandwidth* k if $|i-j| > k$ implies $a_{ij} = 0$. The principal result of this section is to show that for any fixed k, there exists an algorithm with $O(n)$ running time that will solve the TSP for networks with bandwidth k. (The distance matrices for such a class of problem instances have the property that $c_{ij} = +\infty$ if $|i-j| > k$.)

Let H be a Hamiltonian cycle in an undirected graph of bandwidth k. Consider the subgraph H_j that H induces on vertices $1,2,\ldots,j$ where $1 \le j \le n-1$. Each of the vertices $1,2,\ldots,j-k-1$ has degree 2, because of the bandwidth of G. Moreover, H_j contains no cycles (else H is not a Hamiltonian cycle), hence each connected component of H_j is a directed path. The endpoints of these paths are in the set $\{j-k,\ j-k+1,\ldots,j\}$.

Let us define an equivalence relation on the subgraphs induced by Hamiltonian cycles. For given j, subgraphs H_j and H_j' are *equivalent* if

(1) the degrees of vertices $j-k$, $j-k+1,\ldots,j$ are the same, and

(2) for each path (connected component) in H_j there is a path in H_j' with the same endpoints, and conversely.

The significance of the subgraphs H_j and the equivalence relation we have defined on them is suggested by the following observation. Let H be an optimal Hamiltonian cycle and let H induce H_j on vertices $1,2,\ldots,j$. Then for each H_j' in the same equivalence class as H_j, it must be the case that $c(H_j') \geq c(H_j)$. Else a shorter tour H' could be obtained by removing the edges of H_j' from H and substituting those of H_j'. It follows that we can state a necessary condition for the optimality of tours in terms of the lengths of shortest equivalent subgraphs.

It turns out that knowledge of shortest equivalent subgraphs is also sufficient to enable us to find an optimal tour. Our strategy is as follows. Having found a shortest subgraph in each equivalence class for vertices $1,2,\ldots,j$, we then use this information to find a shortest subgraph in each equivalence class for $1,2,\ldots,j+1$. Finally, having found a shortest subgraph in each equivalence class for $1,2,\ldots,n$, we find an optimal tour.

Let S denote an equivalence class of subgraphs on $1,2,\ldots,j+1$. Each subgraph H_{j+1} in S induces a unique subgraph H_j on vertices $1,2,\ldots,j$. The equivalence class to which H_j belongs depends upon the set Δ of zero, one, or two edges incident to $j+1$ in H_{j+1}. Conversely, for each equivalence class S' of subgraphs on $1,2,\ldots,j$, there are various subsets Δ of edges incident to $j+1$ that yield legitimate subgraphs on vertices $1,2,\ldots,j+1$, and each Δ yields a subgraph in a different equivalence class. Thus we can define a mapping τ where $\tau(S', \Delta)$ denotes the equivalence class of a subgraph in S', when the subgraph is augmented by the edges in Δ. ($\tau(S', \Delta)$ is undefined if the augmented subgraph is not legitimate.)

Let $C(S, j+1)$ denote the length of a shortest subgraph in the equivalence class S on $1,2,\ldots,j+1$, and let $c_{j+1}(\Delta)$ denote the length of the subset Δ of edges incident to $j+1$. Then, by the usual sort of dynamic programming argumentation we have

$$C(S, j+1) = \min_{\Delta}\{C(S', j) + c_{j+1}(\Delta) \mid S = \tau(S', \Delta)\}. \tag{20}$$

Now notice that every Hamiltonian cycle H contains exactly two edges incident to vertex n, say (a,n) and (b,n). These edges determine the equivalence class $S_{a,b}$ to which H_{n-1} belongs. There are only a finite number of choices of a and b. Minimizing over them, we find that the length of a shortest tour is

$$\min_{a,b}\{C(S_{a,b}, n-1) + c_{an} + c_{bn}\}. \tag{21}$$

With appropriate initial conditions, equations (20) and (21) provide the basis for a dynamic programming solution to the TSP. We note that, for any fixed bandwidth k, the number of equivalence classes S, the number of sets Δ, and the number of choices of vertices a, b, is each fixed. It follows that the equations (20) and (21) can be solved in $O(n)$ time.

Of course, we should like to have an estimate of how rapidly the computational effort grows with k. The most difficult part of making such an estimate is to determine the number $N(k)$ of equivalence classes S for a given bandwidth k. We make a counting argument as follows. Let us partition the equivalence classes of subgraphs on $1,2,\ldots,j$ into three groups, determined by the degree of vertex $j-k$:

(0) Vertex $j-k$ cannot have degree 0 unless $k \le j \le n-1$, so we ignore this possibility.

(1) In the case vertex $j-k$ has degree 1, it is the endpoint of a path and the other endpoint is l, where $j-k+1 \le l \le j$. For each of the $k-1$ possibilities for l, there are $N(k-2)$ equivalence classes, determined by the other $k-2$ vertices between $j-k+1$ and j.

(2) In the case vertex $j-k$ has degree 2, there are $N(k-1)$ equivalence classes, determined by vertices $j-k+1,\ldots,j$.

Thus we have

$$N(k) = N(k-1) + (k-1)\,N(k-2)\ , \tag{22}$$

with the initial conditions $N(1) = 0$, $N(2) = 1$. A tabulation of values as determined by (22) is as follows:

k	1	2	3	4	5	6	7	8	9	10	11	12	13
$N(k)$	0	1	1	7	11	46	112	434	1,130	5,236	18,536	76,132	298,564

Although $N(k)$ grows quite slowly at first, its growth rate soon becomes quite explosive. Noting that $N(k)$ is $O((k-1)!)$ and that the number of Δ's is $O(k^2)$, we can easily verify that the running time of the dynamic programming computation, expressed in terms of both k and n, is $O((k+1)!n)$.

The dynamic programming computation described here was suggested by ideas of [Monien & Sudborough 1981; Ratliff & Rosenthal 1983]. We leave certain extensions and generalizations as exercises.

Exercises

36. Extend the dynamic programming computation to the asymmetric bandwidth limited TSP. Compute $N(k)$ for this case.

37. Consider the *stripe-width* limited problem: $(j-i) > k \pmod n$ implies $c_{ij} = +\infty$. Show that the difficulty of solving the TSP for stripe-width k is about the same as for bandwidth $2k$.

16. Reducible Networks

Some instances of the TSP can be easily solved, or at least significantly simplified, because the underlying network is wholly or partially *reducible*, as we shall describe.

Let $G = (V,A)$ be a directed graph. A *directed cut* of G is a bipartition of its vertex set V into nonempty subsets S,T such that no arc extends from a vertex

in T to a vertex in S. (That is, all arcs extending across the cut are directed from S to T.) Let us say that a bipartition (S,T) defines an *almost directed cut* if there is exactly one arc (u,v) such that $u \in T, v \in S$.

If an instance of the TSP is defined over a network that contains a directed cut (S,T), then it clearly has no feasible solution, since there is no way to reach a city in S from a city in T. Suppose we have a network that contains an almost directed cut (S,T). Then we know that (u,v), the only arc from T to S, must be contained in any feasible tour and hence in an optimal tour, if one exists. It follows that we can delete from the network all arcs (u,y), where $y \neq v$, and (x,v), where $x \neq u$, because such arcs cannot be contained in a feasible tour. The elimination of such arcs may create additional almost directed cuts, where such cuts did not exist before.

Let us call a directed network *reducible* if it can be transformed to a tour by the deletion of arcs through the repeated discovery of almost directed cuts. (This notion of reducibility is, effectively, a generalization of the notion of reducibility for so-called flow graphs. See, for example, [Hecht & Ullman 1972].) As a very simple example, the directed graph shown in Figure 24 is reducible and contains exactly one tour, namely (1,2,3,4). Even if a network is not reducible, the deletion of arcs through the discovery of almost directed cuts may yield a significant simplification.

-- Insert Figure 24 about here --

17. Halin Graphs and 3-Edge Cutsets

A *Halin graph* is constructed as follows: Start with a tree T in which each nonleaf vertex has a degree of at least three. Embed the graph in the plane and then add new edges to form a cycle C containing all the leaves of T in such a way that the resulting graph $H = T \cup C$ remains planar (see Figure 25.) These graphs were introduced by R. Halin as an example of a class of edge-minimal planar 3-connected graphs. Halin graphs are Hamiltonian and remain so if any single vertex is deleted. Also, every edge belongs to a Hamiltonian cycle and the number of such cycles can grow exponentially with the size of the graph.

In [Cornuéjols, Naddef & Pulleyblank 1984] it is shown how to solve the TSP when the underlying network is a Halin graph, or more generally, when it can be decomposed by the discovery of 3-*edge cutsets*; we shall follow their exposition closely. First some definitions.

An *edge cutset* of a connected (undirected) graph G is a minimal set of edges whose deletion leaves a disconnected graph. If it contains exactly k edges, then it is a *k-edge cutset*. The following is apparent: *Every Hamiltonian cycle contains exactly two edges of every* 3-*edge cutset.*

Let $H = T \cup C$ be a Halin graph. If T is a star, i.e. a single vertex v joined to $n - 1$ other vertices, then H is a *wheel* and is the simplest type of Halin graph. Suppose H is not a wheel, and let w be a nonleaf that is adjacent to exactly one other nonleaf of T. (At least two such nonleafs must exist.) The set of leaves of T adjacent to w, which we denote by $C(w)$, comprises a consecutive subsequence of the cycle C. We call the subgraph of H induced by $\{w\} \cup C(w)$ a *fan* and call w the *center* of the fan. In Figure 25 the black nodes are the centers of the fans indicated by dotted lines. Now notice that there are exactly three edges extending between a fan and the remainder of the Halin graph. These

edges constitute a 3-edge cutset.

-- Insert Figure 25 about here --

Let u and v be the endpoints of the portion of the cycle C that is induced by $C(w)$. We know that any tour of H must enter and leave the fan at either (a) u and w, (b) v and w, or (c) u and v, and describe a Hamiltonian path within the fan between those pairs of vertices. For cases (a) and (b) the Hamiltonian path is uniquely prescribed, as shown in Figure 26. And in case (c) there are only a limited number of possibilities for the Hamiltonian path, as shown in the figure. It follows that we can easily compute the length of a shortest Hamiltonian path for each of these three cases.

-- Insert Figure 26 about here --

Now what we propose to do is this: We shall condense all the vertices of a fan $\{w\} \cup C(w)$ into a single vertex x, thereby obtaining a smaller Halin graph H'. The edges of the 3-edge cutset associated with the fan will have their lengths modified in H' so that an optimal tour in H' is identified with an optimal tour in h. We shall continue condensing fans in this way until finally we obtain a Halin graph that is a wheel. (Wheels are easy to solve.) We shall then backtrace our steps, constructing an optimal tour in the original Halin graph.

In order to modify the lengths of the edges in the 3-edge custset, we need only solve a system of three linear equations in three unknowns. Let (w, w'), (u, u'), (v, v') be the edges of the 3-edge cutset, with lengths $c_{ww'}$, $c_{uu'}$, $c_{vv'}$, respectively. These edges become (x, w'), (x, u'), (x, v') in the condensed graph H', with lengths $c_{xw'}$, $c_{xu'}$, $c_{xv'}$. Let $C^*(u, w)$, $C^*(v, w)$, $C^*(u, v)$ denote the lengths of shortest Hamiltonian paths in $C(w)$ between u,w, between v,w and between u,v. Then we have the system

$$c_{xw'} + c_{xu'} = c_{ww'} + c_{uu'} + C^*(u, w),$$
$$c_{xw'} + c_{xv'} = c_{ww'} + c_{vv'} + C^*(v, w),$$
$$c_{xu'} + c_{xv'} = c_{uu'} + c_{uv'} + C^*(u, v).$$

-- Insert Figure 27 about here --

We now illustrate with a small example. Consider the Halin graph in Figure 27(a). In order to condense the fan whose center is w_1 into a single node x_1, we must solve the equations:

$$\begin{aligned} c_{x_1w_3} + c_{x_1v_2} &= c_{w_1w_3} + c_{u_1v_2} + C^*(u_1, w_1) \\ &= 1 + 2 + 9 \\ &= 12, \\ c_{x_1w_3} + c_{x_1v_3} &= c_{w_1w_3} + c_{v_1v_3} + C^*(v_1, w_1) \\ &= 1 + 7 + 8 \\ &= 16, \\ c_{x_1v_2} + c_{x_1v_3} &= c_{u_1v_2} + c_{v_1v_3} + C^*(u_1, v_1) \end{aligned}$$

$$= 2 + 7 + 7$$

$$= 16.$$

This yields the solution

$$c_{x_1 w_3} = 6,$$

$$c_{x_1 v_2} = 6,$$

$$c_{x_1 v_3} = 10,$$

and the network shown in Figure 27(b). Carrying the process one step further, we obtain the wheel shown in Figure 27(c) which has a shortest tour as shown by wiggly lines. This implies that the tour shown in Figure 27(b) is optimal and that in turn implies that the tour shown in Figure 27(a) is optimal.

In [Cornuéjols, Naddef & Pulleyblank 1984], an explicit description of the traveling salesman polytope for a Halin graph is also given.

Exercise

38. Consider the generalization of Halin graphs in which nonleaf nodes may have degree 2. Show how the TSP for such graphs can be solved by the approach described here.

18. Conclusion

In this chapter we have shown that a number of special cases of the TSP can indeed be easily solved. Among these cases are the constant TSP, the upper triangular TSP, the small TSP, the Demidenko TSP, the bottleneck TSP for graded matrices, the Gilmore-Gomory TSP, the bandwidth-limited TSP, and the TSP for reducible networks and Halin graphs. Many of our results were obtained through subtour patching and we have indicated some basic theory of this technique.

We have noted some interesting demarcations between easy problems and hard problems. The ordinary TSP is $\underline{\underline{\text{NP}}}$-hard for graded matrices, whereas the bottleneck TSP is easy. The TSP is $\underline{\underline{\text{NP}}}$-hard for asymmetric product matrices but easy for symmetric ones. It is easy to find a shortest Hamiltonian path for circulants, but there is not much we can say about the TSP for these matrices.

We hope that at least a few of the special cases dealt with in this chapter may have direct practical application. We believe, however, that in the long run the greatest importance of these special cases will be for approximation algorithms. Much remains to be done in this area.

Bibliography

[Berenguer, 1979] X. Berenguer, "A Characterization of Linear Admissible Transformations for the m-Travelling Sales Problem," *Eur. J. Operational Res.* 3 (1979) 232-249.

[Cornuéjols et al, 1983] G. Cornuéjols, D. Naddef and W.R. Pulleyblank, "Halin Graphs and the Traveling Salesman Problem", *Math. Programming*, 26 (1983) 287-294.

[Demidenko, 1979] V. M. Demidenko, "The Traveling Salesman Problem with Asymmetric Matrices", *Vesit. Akad. Navuk BSSR Ser. Fiz.-Mat. Navuk* 1, (1979) 29-35.

[Fuller, 1972] Samuel H. Fuller, "An Optimal Drum Scheduling Algorithm," *IEEE Trans. Computers*, C-21 (1972) 1153-1165.

[Gabovich, 1970] E. Ya. Gabovich, "The Small Travelling-Salesman Problem" (in Russian), *Trudy Vychisl. Tsentra Tartusk. Gos. Univ.* 19, (1970) 27-51.

[Gabovich, 1976] E. Ya. Gabovich, "Constant Discrete Programming Problems on Substitution Sets", *Cybernetics* (1977) 786-793.

[Garfinkel, 1977] R.S. Garfinkel, Minimizing Wallpaper Waste, part I: a class of traveling salesman problems. *Oper. Res.* 25 (1977) 741-751.

[Gilmore and Gomory, 1964] P. C. Gilmore and R. E. Gomory, "Sequencing a One State-Variable Machine: A Solvable Case of the Travelling Salesman Problem," *Operations Res.* 12 (1964) 655-679.

[Hecht & Ullman, 1972] M.S. Hecht and J.D. Ullman, "Flow Graph Reducibility", *SIAM J. Comput.* 1 (1972) 188-202.

[Herstein, 1975] I. Herstein, *Topics in Algebra*, Xerox College Pub., Lexington, MA (1975).

[Johnson, 1954] S. M. Johnson, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Log. Quart.* 1 (1954) 61-68.

[Klyaus, 1976] P.S. Klyaus, "Structure of the Optimal Solutions of Certain Classes of Traveling Salesman Problems,"', *Vesti Akad. Nauk BSSR* 6, (1980) 95-98.

[Lawler, 1971] Eugene L. Lawler, "A Solvable Case of the Travelling Salesman Problem," *Math. Programming* 1 (1971) 267-269.

[Lenstra and Rinnooy Kan, 1975] J. K. Lenstra and A. H. G. Rinnooy Kan, "Some Simple Applications of the Traveling Salesman Problem," *Op. Res. Q.*, 26 (1975) 717-733.

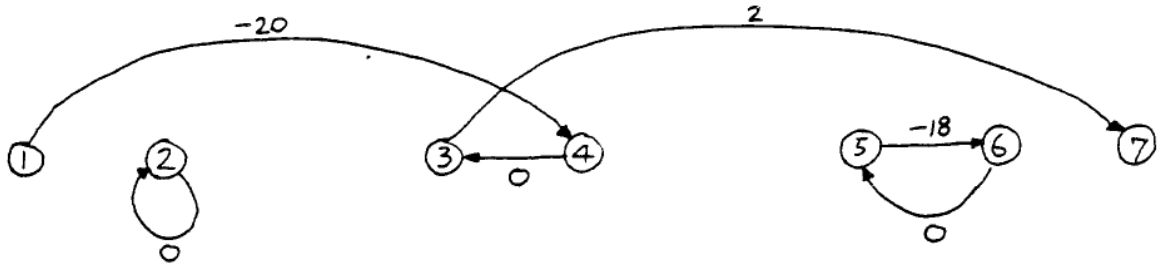
[Lenstra and Rinooy Kan, 1979] J. K. Lenstra and A. H. G. Rinnooy Kan, "A Characterization of Linear Admissible Transformations for the m-Travelling Salesmen Problem: A Result of Berenguer," *Eur. J. Operational Res.* 3 (1979) 250-252.

[Monien & Sudborough ???] To be supplied.

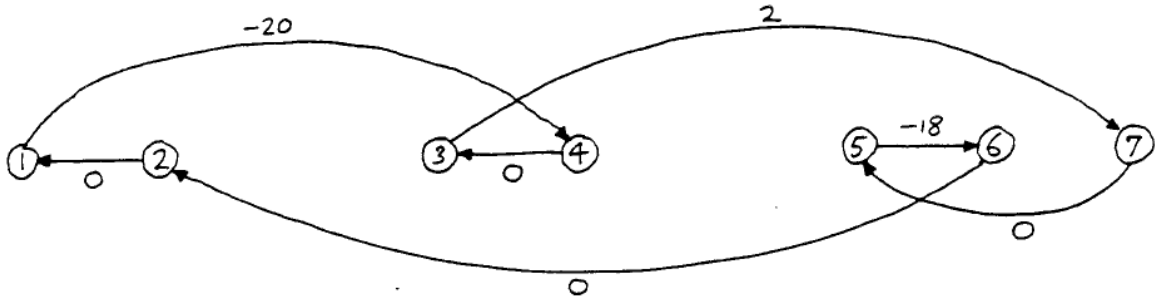
[Ratliff and Rosenthal ???] To be supplied.

[Sarvanov, 1980] V.I. Sarvanov, "On the Complexity of Minimizing a Linear Form on a Set of Cyclic Permutations", *Dokl. Akad. Nauk SSSR* 253, (1980) ?? (Translation: Soviet Math. Dokl. 22, (1980) 118-120.)

[Syslo, 1973] Maciej M. Syslo, " A New Solvable Case of the Travelling Salesman Problem," *Math. Programming*, 4 (1973) 347-348.



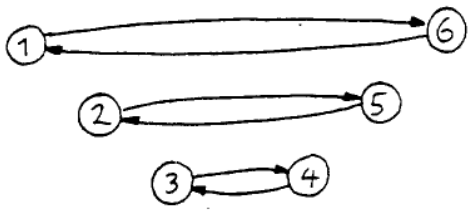
(a) Optimal solution to assignment problem



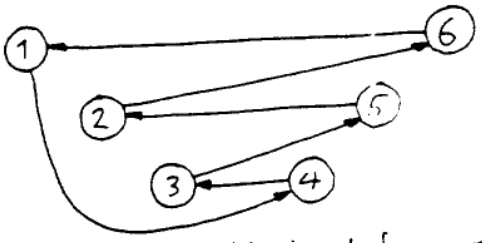
(b) Optimal solution to TSP

FIGURE 1

CHAPTER 4



(a) Optimal assignment φ



(b) Tour τ obtained from φ

FIGURE 2

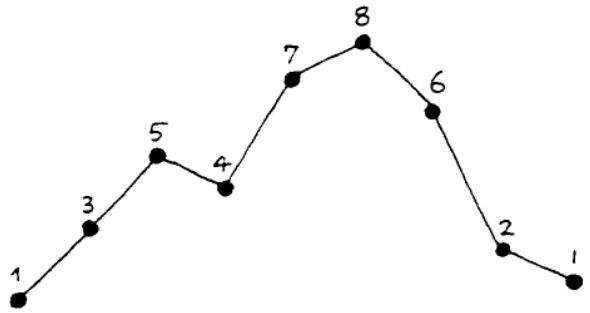
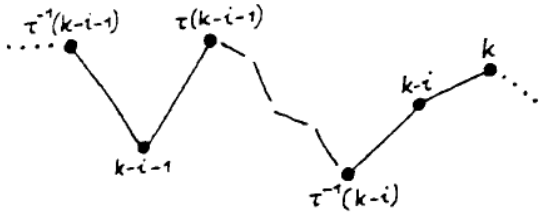
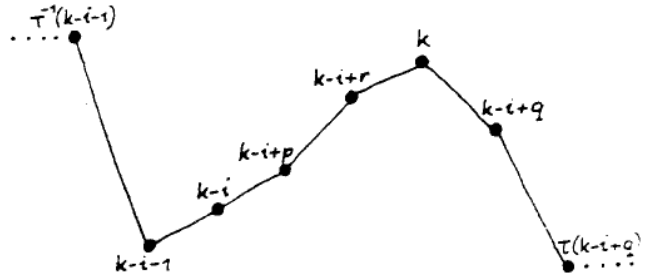


FIGURE 3

Case I(a)



Case I(b)



Case I(c)

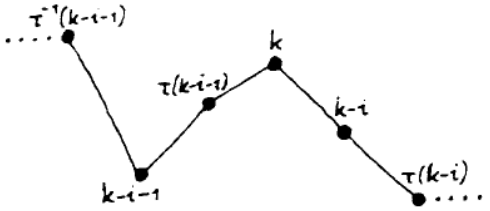


FIGURE 4

CHAPTER 4

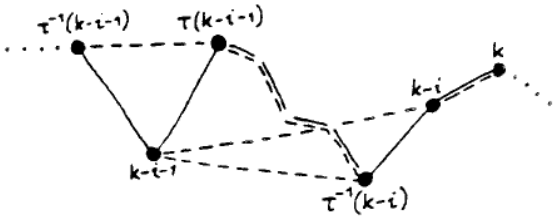


FIGURE 5

FIGURE 6

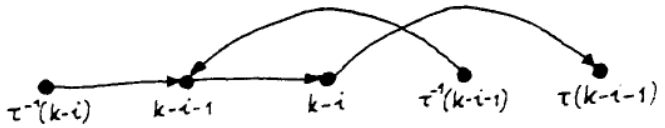


FIGURE 7

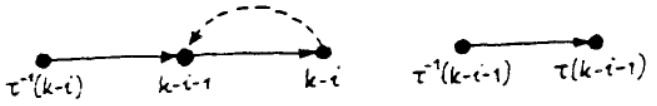


FIGURE 8

CHAPTER 4

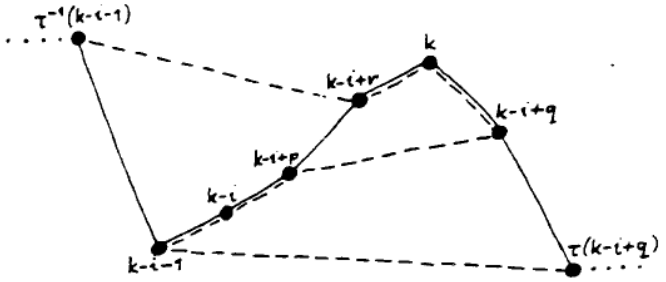


FIGURE 9

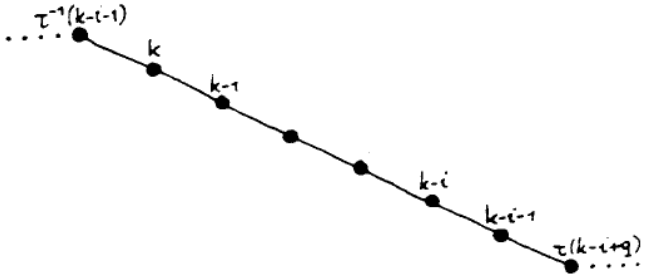


FIGURE 10

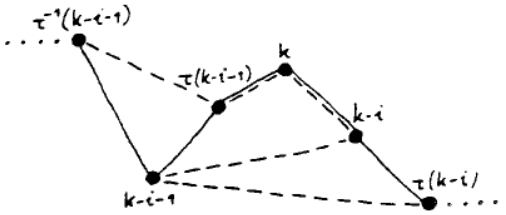
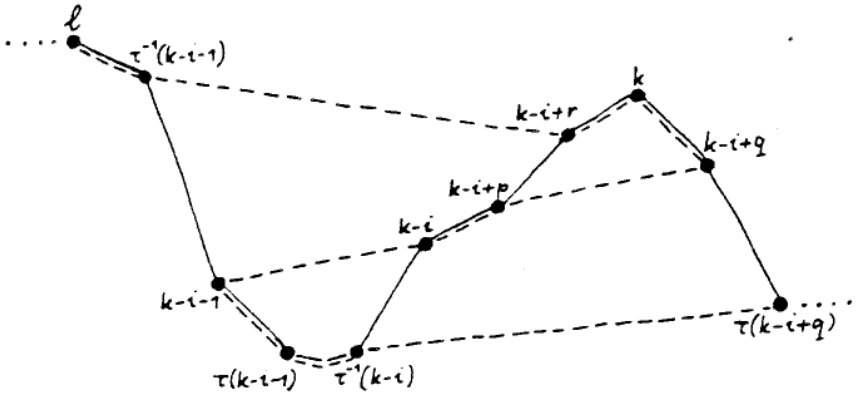


FIGURE 11

Case II(a)



Case II(b)

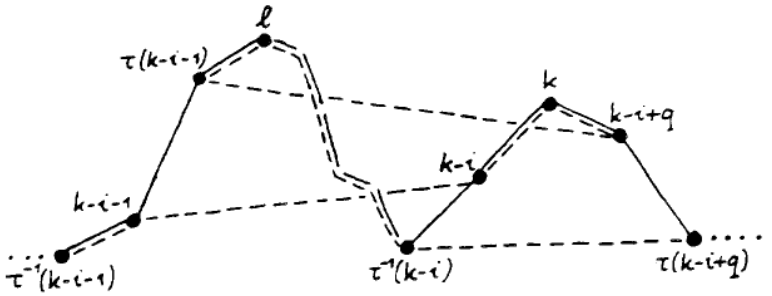


FIGURE 12

CHAPTER 4

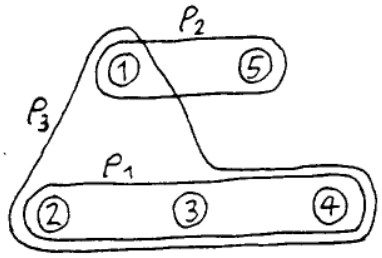
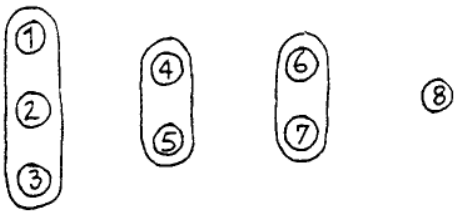
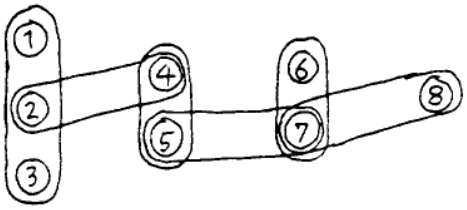


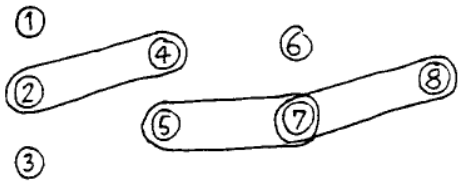
FIGURE 13. Hypergraph $H(V, P)$



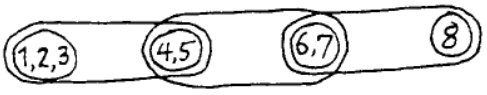
(a) Hypergraph $H(V, \{\Phi_i\})$



(b) Tree obtained by adding transpositions



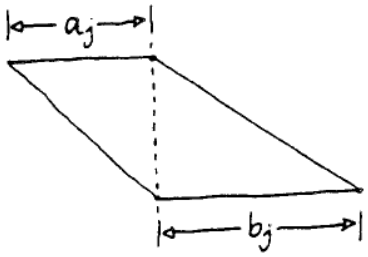
(c) Graph of transpositions



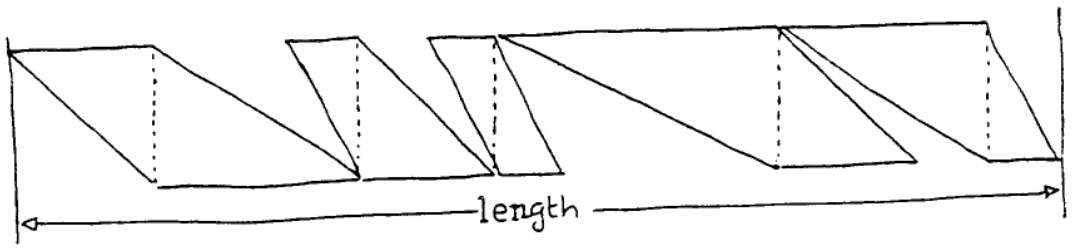
(d) Tree of transpositions after contraction of subtours Φ_i

FIGURE 14

CHAPTER 4



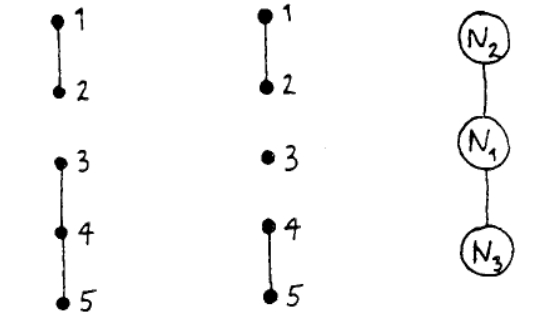
(a) Typical trapezoid



(b) Feasible s

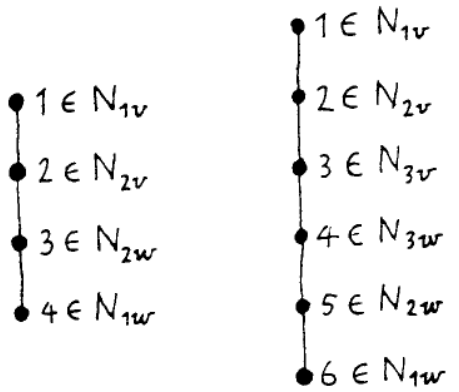
FIGURE 15

CHAPTER 4



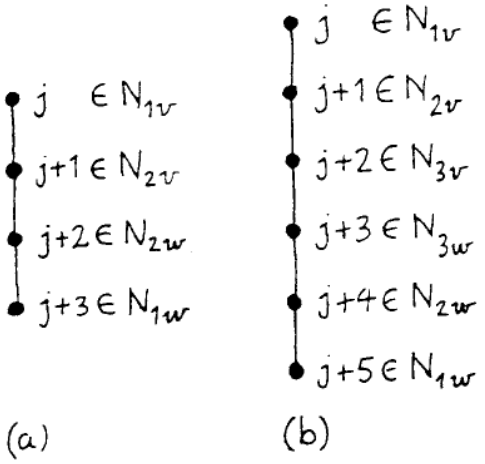
(a) $G(\Pi)$ (b) G' (c) $P(G')$

FIGURE 16



(a) (b)

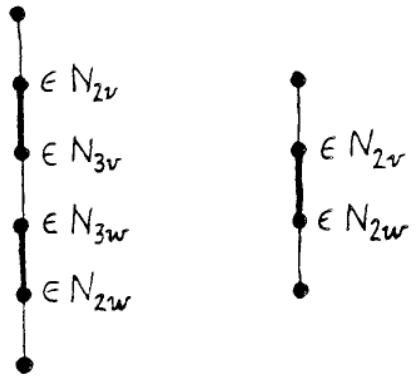
FIGURE 17



(a) (b)

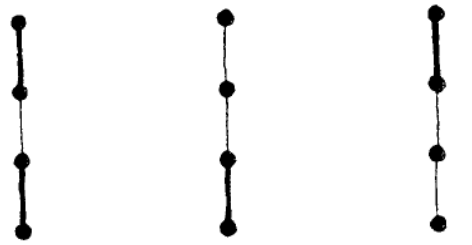
FIGURE 18

CHAPTER 4



(a) (b)

FIGURE 19



(a) (b) (c)

FIGURE 20

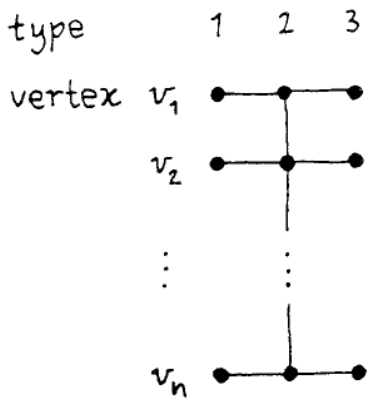


FIGURE 21

CHAPTER 4

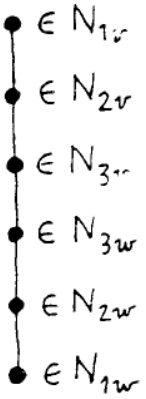


FIGURE 22

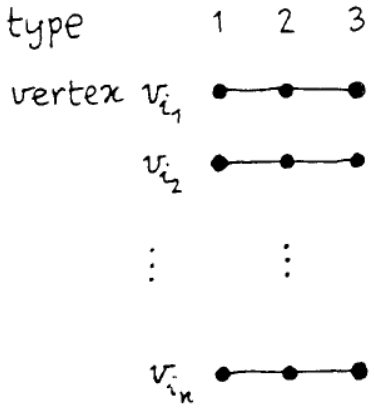


FIGURE 23

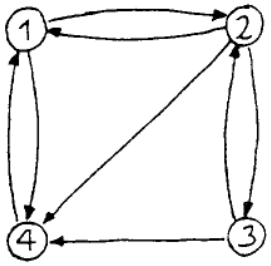


FIGURE 24. A reducible digraph

CHAPTER 4

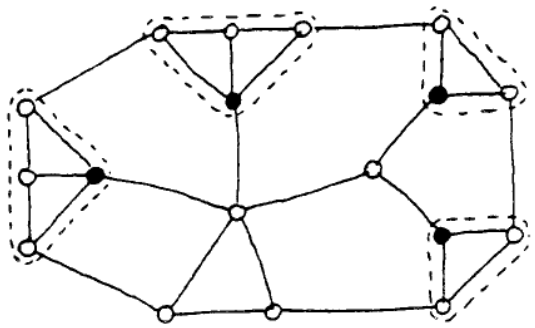
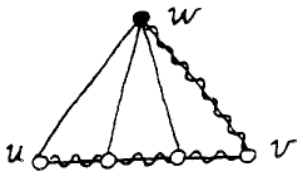
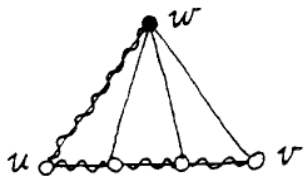


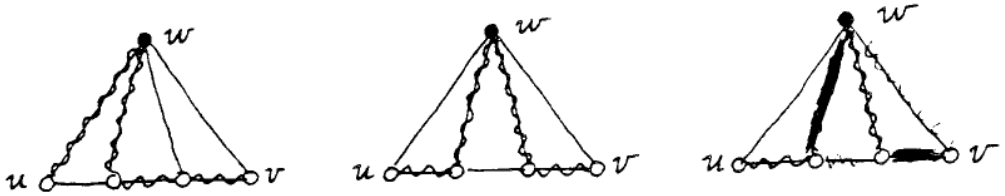
FIGURE 25. A Halin graph



(a) Unique Hamiltonian path between u and w



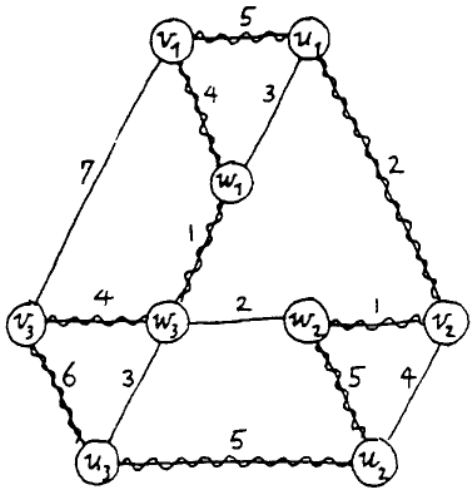
(b) Unique Hamiltonian path between v and w



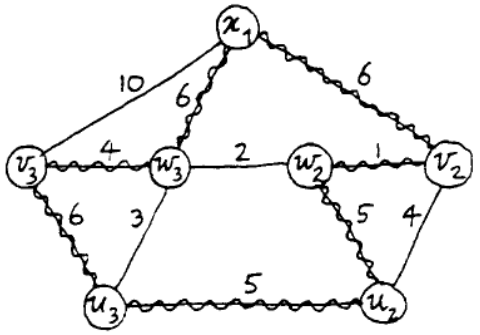
(c) All Hamiltonian paths between u and v

FIGURE 26

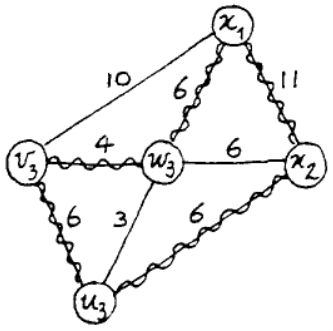
CHAPTER 4



(a)



(b)



(c)

FIGURE 27. Solution of TSP for Halin graph