ANALYSIS OF A COMBINATORIAL APPROACH TO THE TRAVELLING SALESMAN PROBLEM

by

Glen R. Thompson B.Sc.(Hons.) (Adel.)

Thesis submitted for the Degree of
Doctor of Philosophy
in the University of Adelaide,
Department of Mathematics,
February, 1968.

TABLE OF CONTENTS

SUMMARY

The subject of this thesis is the study of the recently introduced concept of p-optimality of tours in a network and the application of this concept to the Travelling Salesman Problem.

This study consists of two parts. The first is the theoretical derivation of the number of trials required to test whether a given tour is p-optimal, subject to various assumptions on the nature of an algorithm for carrying out such a test. The main part of this derivation consists of the solution of a number of combinatorial problems concerning the permutations and combinations of points and arcs on the circumference of a circle. These combinatorial results are proved using the notation of tours in a network, although the results themselves can be stated quite generally. These general statements and some physical interpretations are included in an appendix.

The second part consists of a study of the practical application of p-optimality to the Travelling Salesman Problem. A general algorithm for generating p-optimal tours is described and its applicability to the Travelling

Salesman Problem is discussed. A practical version of this algorithm and its application to finding suboptimal solutions to Travelling Salesman Problems is then described. This algorithm is based on the generation of 3-optimal tours, and is a modification of an existing method due to S. Lin. It differs from Lin's method both in the manner in which 3-optimal tours are generated and in the way in which the probability of having obtained an optimal tour at a given stage is estimated. A summary of extensive computational results enables some empirical conclusions to be drawn on the applicability of this algorithm to different sized networks. Finally, an accelerated algorithm is described. This algorithm differs from the above systematic algorithm in that instead of generating 3-optimal tours, it generates "almost 3-optimal" tours by selectively omitting parts of the systematic algorithm. The effect of these omissions in practice is a great increase in computational efficiency, accompanied by only a slight increase in the number of trial tours requiring to be generated. Further computational results demonstrate the efficacy of the accelerated algorithm.

SIGNED STATEMENT

This thesis contains no material which has been accepted for the award of any other degree or diploma in any University. To the best of my knowledge and belief, the thesis contains no material previously published or written by any other person, except where due reference is made in the text of the thesis.

(Glen R. Thompson)

ACKNOWLEDGEMENTS

The author is indebted to his supervisor Professor R.B. Potts for suggesting the topic which led to the results contained in this thesis, and for his encouragement and assistance over the past two years. The author also wishes to thank his acting supervisor Dr R.G. Keats for many helpful suggestions during Professor Pott's absence. He is also indebted to P.G. Pak-Poy of Traffic Planning and Research Pty. Ltd. for the opportunity to gain practical experience of the problem of scheduling deliveries from a single depot. The author also wishes to thank Mrs W. Hind for her efficient typing of the manuscript.

CHAPTER 1

INTRODUCTION

1.1 General

The subject of this thesis is the concept of p-optimality of tours in a network, and its application to the now-classic Travelling Salesman Problem.

The term p-optimality (where p is a positive integer) was introduced in 1965 by S. Lin [13], who obtained some elementary results concerning p-optimality, and developed an efficient computer programme for computing approximate solutions to Travelling Salesman problems by generating 3-optimal tours. An earlier method due to Croes [6] (1958) included a procedure for generating a 2-optimal tour by the application of operations which Croes called inversions.

In this thesis the subject of p-optimality is studied for arbitrary p, and an improved algorithm for generating approximate solutions to Travelling Salesman problems is presented.

1.2 The Travelling Salesman Problem

In view of the existing review papers on the Travelling Salesman Problem, only the major events in the history of the problem are given here. Comprehensive reviews of the problem are given by Flood [8] (1955), Arnoff and Sengupta [2] (1961) and by Bellmore and Nemhauser [4]

(1966). The last paper also contains an extensive bibliography.

Attention by mathematicians to the Travelling Salesman Problem dates back to 1930, when Menger [16] presented the problem at a mathematical colloquium in Vienna, and stated that there was no known way of reducing the number of trials below the number of permutations of the given points (quoted by F. Bock in Graves and Wolfe [5]). In 1954, Dantzig, Fulkerson and Johnson [7] published a method of solution using linear programming. This method was the fore-runner of the class of methods involving integer programming, one of the three main classes of methods of solution. Methods involving dynamic programming form the second class. Earliest methods in this class were those of Bellman [3] and Held and Karp [10], both in 1961. The third and most recent of the classes of method is the class of branch and bound algorithms introduced in 1963 by Little, Murty, Sweeney and Karel [14].

Each of the above three classes has certain disadvantages, and it is difficult to determine whether one method is necessarily better than the others. A detailed comparison of the computational results of current methods has been carried out by Bellmore and Nemhauser [4]. Their conclusion is that for problems involving less than 13 nodes, a dynamic programming method is preferable because of its predictable computation time. (Using Lin's modification of Held and Karp's method, this method is also more efficient, computationally). For larger problems (up to 70 nodes) Bellmore and Nemhauser recommend the branch and bound algorithm of Shapiro [20]. A recent integer programming method, by Martin [15], appears to be approximately as efficient, computationally, as the branch and bound methods, in some examples. However, integer programming methods have the disadvantage of widely differing computation times for different problems.

Such a comparison of computational efficiencies is of limited value, as computation times depend on the nature of the computers used, and on the ingenuity with which the various algorithms are programmed. However this comparison does give a qualitative indication of the applicability of the various algorithms.

The Travelling Salesman Problem arises in many practical situations; for example the scheduling of deliveries from a single depot to a number of customers, the scheduling of jobs in a machine shop (see Gilmore and Gomory [9]), and the orientation of an orbiting telescope to photograph a set of stars, using a minimal amount of fuel (see Staudhammer and Ash [21]). In practical applications there is frequently the need for a method of generating "good" solutions: solutions which, although not necessarily optimal, are almost optimal in some sense, and are obtained by a computationally efficient procedure. If a Travelling Salesman algorithm is applied in practice, to reduce the cost of some delivery operation for example, then the cost of the computation becomes part of the total cost of the operation. Using the above algorithms it is possible for the cost of computation to become a significant part of the overall cost. The cheapest route in practice is therefore not necessarily the optimum tour.

Consequently a large number of computational techniques have been evolved in recent years. These practical methods vary widely in their nature and in the extent to which it is possible to guarantee their results. Despite the wide variation between individual methods, they may be grouped in three categories.

The first category contains those methods which are obtained by modifying optimal methods. The branch and bound methods are particularly amenable to such modifications, for two reasons. Firstly, in most branch and bound methods (see, for example, Little et al [14] and Sweeney [22]), a sequence of tours of decreasing length is generated, and together with each tour is calculated a lower bound on the length of the optimal tour. Thus, if a branch and bound procedure is terminated before optimality has been proved, an upper bound on the possible error is available. Secondly it is observed in practice that in many cases a major part of the procedure consists of proving that the final tour obtained is in fact optimal. Thus, in many cases, it is possible for a tour resulting from an abbreviated branch and bound procedure to be optimal. In a given amount of computing time, however, better results tend to be obtained from methods in the remaining two categories.

The second category consists of methods in which a good starting tour is generated, and then improved by the application of some set of operations. The methods in this category are usually based on intuitively reasonable assumptions, and their success can only be measured by results obtained in practice. Recent examples of such methods are those of Roberts and Flores [19], and Staudhammer and Ash [21].

Methods in the third category differ from those in the second in that a random initial tour is subjected to a rapid tour-improvement procedure. The process is then repeated a number of times in order to generate a random sample of a set of short tours, which can be shown, or is presumed, to contain an optimal tour. Examples are the methods of Reiter and Sherman [23] and Lin [13]. Methods in this category are applicable to comparatively large problems (up to 150 nodes) and at present the method of Lin, in particular, appears to be computationally the most efficient. Another advantage of these methods lies in the fact that the result is a set of short tours, rather than a single tour. It is convenient in some applications to select from such a set of tours a "best tour", taking into consideration various factors in addition to tour length.

The two algorithms described in Chapter 3 of this thesis are modifications of Lin's algorithm, and fall into the third category. The simple modification applied in the accelerated algorithm in particular results in a marked increase in computational efficiency, even though a larger number of short tours needs to be generated.

1.3 p-Optimality of Tours in a Network

In this section the term p-optimality is defined and some elementary results are obtained. Other basic terminology and notation are also introduced.

Let $\mathcal{N}$ be a given network consisting of n nodes i, $1 \leqslant i \leqslant n$, and links (i,j) joining every pair of nodes i, j, $i \neq j$. Let the length (a function of distance and/or time) of each link (i,j) be d_{ij}. By convention, suppose $d_{ii} = \infty$ for all i, and if there is no direct path between nodes i and j, let $d_{ij} = \infty$. Throughout this thesis it is assumed that $d_{ij} = d_{ji}$ for all i, j. This assumption is adhered to even though in some instances it is necessary to distinguish between the two directed links (i,j) and (j,i).

A <u>tour</u> is defined to be a circuit which includes every node of the network, and may be represented by a cyclic permutation $(i_1 \; i_2 \; i_3 \; \ldots \; i_n)$ of the node numbers (where $i_1 = 1$, for uniqueness of representation). Equivalently a tour may be represented by a list (which need not be ordered) of the n links occurring in the tour. The <u>length</u> of a tour $(i_1 \; i_2 \; \ldots \; i_n)$ is the sum of the lengths of the links in the tour, viz.

$$d_{i_n i_1} + \sum_{\nu=1}^{n-1} d_{i_\nu i_{\nu+1}} .$$

A tour is <u>optimal</u> in $\mathcal{N}$ if it has minimum length.

If p is an integer and $0 \leqslant p \leqslant n$, a tour is defined to be <u>p-optimal</u> if it is not possible to transform the tour into a shorter tour by replacing a set of p links in the tour by any other set of p links. It should be noted that the second set of links is not necessarily disjoint from the first. It follows that if a tour is p-optimal, then it is q-optimal for all integers q, $0 \leqslant q \leqslant p$.

A useful equivalent definition of p-optimality is the following: A given tour is p-optimal if and only if for every tour shorter than the given tour, the given tour contains at least $p + 1$ links which are not contained in the shorter tour, or equivalently, if and only if every tour shorter than the given tour contains at least $p + 1$ links which are not contained in the given tour.

The latter alternative definition immediately implies the following result: If a tour is p-optimal then <u>either</u> it is optimal <u>or</u> it has at most $n - p - 1$ links in common with an optimal tour.

Two more elementary results are the following: A tour is optimal if and only if it is n-optimal (since any tour may be transformed to any other tour by replacing up to n of its links). Every tour is 0-optimal and 1-optimal (since it is not possible to alter a tour by replacing 0 or 1 of its links). The above results are stated by Lin [13], who also states two further results giving physical interpretations of the meaning of 2-optimality and 3-optimality. It may be noted that part (c) of Lin's Theorem 3 is incorrect. It is possible to construct a 2-dimensional Euclidean network containing a tour which does not intersect itself and is not 2-optimal.

In order to prepare for Chapter 2, some further terminology is now introduced. Given an arbitrary tour, which for the present will be called the <u>initial tour</u>, let every tour which has exactly $n - p$ links in common with the initial tour be called a <u>p-tour</u>. In other words, a p-tour is a tour which can be obtained from the initial tour by the replacement of p links in the initial tour by p different links. It follows that a tour is p-optimal if and only if its length is less than or equal to the length of every q-tour for $0 \leqslant q \leqslant p$.

The theoretical results of Chapter 2 are mainly concerned with the problem of determining the number of trials required in an enumerative procedure for testing a tour for p-optimality. At the present time the only result through which some of the trials may be avoided is the result on degeneracy in section 2.5. Apart from minor variations the procedures described in Chapter 2 consist of the enumeration of the set of q-tours, for $0 \leqslant q \leqslant p$ and the comparison of the length of each q-tour with that of the initial tour. The most difficult problem to be solved in connection with this enumeration is the determination of the number of distinct p-tours obtainable from each initial tour. This problem is solved in section 2.2, with the aid of a preliminary result proved in section 2.1. As the method used in section 2.2 does not enable the set of p-tours to be constructed unless another much larger set of tours is first constructed, a second method is given in section 2.3. This method gives rise to some repetitions; however it has the advantage of providing a practical means of constructing the set of p-tours. Section 2.4 contains a simplification of the method of section 2.3, obtained by using the idea of congruence of p-tests. This simplified method is particularly useful in practice for $p \geqslant 4$.

In the above methods the notation of segments is used. It has been stated earlier that for a given initial tour every p-tour can be obtained by replacing p links of the initial tour by p other links. For the purpose of the above enumerative procedures it is more convenient to consider every p-tour to be formed from the initial tour by the following two operations: Firstly p links are deleted from the initial tour to form exactly p chains, each containing one or more nodes. These chains are called segments of the initial tour. The second step is to specify the order and direction in which the p segments are to appear in the final tour. If the second step is carried out in such a way that no two segments which are consecutive in the initial tour are consecutive in the final tour, then the final tour does not contain any of the p deleted links, and is therefore a p-tour.

In section 2.5, on the other hand, it is necessary to consider the sets of deleted and inserted links, instead of segments, in order to define degeneracy of a p-test. However, in obtaining the numerical results in section 2.5, segment notation is again used.

Note that the first five sections of Chapter 2 concern the number of p-tours obtainable from an initial tour. It is only in section 2.6 that these results are used to give the number of trials required to test a tour for p-optimality.

1.4 p-Optimality and the Travelling Salesman Problem

There are two ways in which the concept of p-optimality may be applied to the Travelling Salesman Problem. The first guarantees an optimal solution, but is not computationally feasible at the present time, except for very small networks. The second is described in detail in Chapter 3.

The first method is mentioned here for completeness, and to suggest some unsolved problems. Clearly one way to generate an optimal tour is to generate an n-optimal tour, where n is the number of nodes in the network. For any given network there is a smallest value of p, say p', such that p'-optimality implies n-optimality. The problem of finding such a value of p is at present unsolved. It has not even been found possible to determine an upper bound on p' for given n. Lin [13] conjectures that every $(n-1)$-optimal tour is optimal, and has proved this conjecture for $n \leqslant 6$. If proved, this conjecture would imply that $p' \leqslant n-1$. Results obtained in practice, however, indicate that a much stronger result may in fact hold. For example, for $n = 6$ every 3-optimal tour generated in a large number of trial networks has been found to be optimal, and for $n = 7$ every 4-optimal tour has been found to be optimal. In 2-dimensional Euclidean networks, computational results from several hundred trial networks of up to 50 nodes indicate that p' is less than $\frac{1}{2}n$, and probably much less than $\frac{1}{2}n$ for large n.

This upper bound was obtained by inspection of sets of 3-optimal tours, on account of the difficulty of generating p-optimal tours for large p. In no case was there found a 3-optimal tour which had less than $\frac{1}{2}n$ links in common with an optimal tour. These results suggest that $p' \leqslant \frac{1}{2}n$. Also, it appears from the limited results available for moderately large n (up to $n = 50$) that the ratio $\frac{p'}{n}$ decreases as n increases. For example in the 48-node problem of Held and Karp, the greatest number of links which occurred in a 3-optimal tour, but not in the optimal tour, was found to be 17. This shows that if all 3-optimal tours have been found for this problem (which is likely, using the algorithm of section 3.2) then $p' \leqslant 17$ for this problem.

Although the above results are not conclusive in themselves, they do suggest that the value of p' may prove to be sufficiently small for the above method to be practicable. A second problem to be solved before this method can be used for large n is that of generating p-optimal tours efficiently for large p. The methods used in this thesis (see Chapter 2) are only practicable for $p \leqslant 6$ at present, and the number of repetitions of steps is very large unless p is very small compared to n. Also, results on degeneracy for $p \geqslant 6$ are not known at present. This first method is not investigated any further here.

The second method of applying p-optimality to the Travelling Salesman Problem consists basically of generating a random sample of p-optimal tours for a suitably small value of p, and estimating the probability that this sample contains an optimal tour. This method is discussed in detail in section 3.2, and uses the same general approach as that of Lin [13]. Following Lin's experience with $p = 2,3,4$, the value $p = 3$ is used in this method.

Lin assumes that the probability of a 3-optimal tour being optimal is a function of n, and obtains an empirical estimate of this probability for each n. However it is found in practice that the number of distinct 3-optimal tours varies widely in networks of a given number of nodes. For example, most randomly-generated 2-dimensional networks of 20 nodes are found to possess only a single 3-optimal tour, but some contain 4 distinct 3-optimal tours. Lin's empirical estimate depends only on the number of nodes, and therefore does not take account of this variation between networks.

A second way of estimating the above probability does not rely on any empirical estimates, and does take account of the variation between individual networks. This method is described fully in section 3.2. It consists basically of estimating the probability that an optimum has been obtained after ν distinct 3-optimal tours have been generated in μ trials, assuming that all 3-optimal tours are equally probable. Since shorter 3-optimal tours tend to occur more frequently than longer 3-optimal tours, the second estimate tends to be somewhat pessimistic. However this estimate appears to be more realistic than the purely empirical estimate, as it does not depend on stringent *a priori* assumptions.

A possible improvement to the second estimate may be achieved by obtaining empirically the probability of the occurrence of each of the ν 3-optimal tours in a network. These probabilities may then be used in the estimation of the probability that one of a given set of 3-optimal tours is optimal, instead of assuming equal probabilities. As an optimal tour almost invariably appears more frequently than other 3-optimal tours, this third method should give a more optimistic estimate. However a great amount of trial data is required to determine the above probabilities and to test the reliability of the resulting probability of optimality. This third method has therefore not been fully tested at the present time.

This second method of applying p-optimality to the Travelling Salesman Problem is of an essentially Monte-Carlo nature, and its success is attributable to the following two empirical facts.

(i) 3-optimal tours can be generated extremely rapidly (see section 3.2), and without requiring much computer storage.

(ii) The number of 3-optimal tours is comparatively small, even for medium-sized networks (up to 50 nodes).

The accelerated algorithm described in section 3.3 is a variation of the second method above. Although the tours obtained are not necessarily 3-optimal, and there is a possibility that in some networks the probability of obtaining an optimal tour could be reduced, the above methods may again be used to obtain estimates of this probability. Despite the abovementioned shortcomings of the accelerated algorithm its greatly increased computational efficiency is an important consideration in practical applications.

CHAPTER 2

SOME RESULTS CONCERNING p-OPTIMALITY

This chapter consists mainly of the derivation of results of a theoretical nature, leading eventually to the number of tours whose lengths need to be tested in order to test a given tour for p-optimality.

Section 2.1 is independent of the remainder of the chapter, apart from its application in section 2.2. Sections 2.2 to 2.4 are concerned with the set of p-tours obtainable from a given tour. In section 2.2, the exact number of distinct p-tours is derived for arbitrary p. In section 2.3 a practical method for generating the set of all p-tours is described, and the exact number of tours (including repetitions) obtained using this method is derived for arbitrary p. In section 2.4, the idea of congruence of p-tests is defined, and is used in a modification to the method of section 2.3 which is more practical for $p \geqslant 4$. An enumerative procedure for determining the number of steps in this method is described, and this number is computed for $p \leqslant 6$.

Section 2.5 contains the only known result which enables the p-optimality of a tour to be tested without testing the length of every q-tour for $0 \leqslant q \leqslant p$. This result depends on the idea of degeneracy of a p-test, and is of limited use at present, as there is no known way of enumerating all the degenerate p-tests, except for $p \leqslant 5$. The effect of this result is to reduce the number of p-tests which need to be applied at each stage of the methods of sections 2.3 and 2.4.

In section 2.6, the results of sections 2.2 to 2.5 are used to determine the number of tours whose lengths need to be tested in order to test a given tour for p-optimality, by means of five methods. The number of distinct tours to be tested is obtained from the result of section 2.2, and the numbers of tours (with repetitions) using two practical enumerative procedures are obtained from sections 2.3 and 2.4. The latter two methods are then modified, for $p \leqslant 5$, by deleting the degenerate p-tests, using the result of section 2.5.

2.1 Single-node Segments

The purpose of this section is to determine the number $g(n,p,q)$ of ways in which a tour containing n nodes may be divided into p segments, q of which consist of a single node, where p and q are given integers. This number is required in section (2.2) in the derivation of the exact number of p-tours obtainable from each initial tour.

In order to divide a tour into p segments it is only necessary to delete p distinct links from the tour. A segment consisting of a single node occurs wherever two of these deleted links are adjacent. Thus the number $g(n,p,q)$ is simply the number of ways of selecting p objects (points, links, numbers, etc.), from a set of n objects arrayed on a circle, in such a way that exactly q adjacent pairs of objects occur among the selected objects. This problem has been solved for the special case $q = 0$ by Kaplansky [11].

The derivation of $g(n,p,q)$ depends on the solution of three similar problems, defined on a line instead of a circle, and subject to certain conditions on the first and last objects. These three linear problems will now be considered.

Problems for a Linear Array of Points.

Given $n + 2$ points

$$0, 1, 2, \quad . \; . \; . \quad n, n + 1,$$

arrayed on a line, let $f_1(n,p,q)$ be the number of ways of selecting $p + 2$ of these points such that exactly q adjacent pairs occur, given that the selection must include points 0 and $n + 1$. Alternatively, $f_1(n,p,q)$ is the number of ways of selecting p of the points $1,2,\ldots,n$ such that q adjacent pairs occur, given that points 0 and $n + 1$ have already been selected. Let $f_2(n,p,q)$ be the corresponding number when only point 0 has been previously selected, (or equivalently, when only point $n + 1$ has been selected). Let $f_3(n,p,q)$ be the number of selections in which neither of the points 0 and $n + 1$ are selected.

Lemma 2.1. Given that $n \geqslant 2$, $p \geqslant 1$, and $q \geqslant 1$,

$$f_1(n,p,q) = f_1(n-1,p-1,q-1) + f_1(n-1,p,q) - f_1(n-2,p-1,q-1) + f_1(n-2,p-1,q) \quad (2.1)$$

Proof. Suppose a given selection of p points with q adjacent pairs contains point 1. Then it contains the pair $(0,1)$. It therefore contains $p - 1$ of the points $2,\ldots,n$ with $q - 1$ adjacent pairs occurring among these points. It follows from the definition of f_1 that there are $f_1(n-1,p-1,q-1)$ such selections. On the other hand every selection which does not contain point 1 contains p of the $n-1$ remaining points, with q adjacent pairs. It follows from the definition of f_2 that there are $f_2(n-1,p,q)$ selections not containing point 1. The total number of selections containing p points and q adjacent pairs is therefore

$$f_1(n,p,q) = f_1(n-1,p-1,q-1) + f_2(n-1,p,q). \tag{2.2}$$

Similar reasoning gives

$$f_2(n,p,q) = f_1(n-1,p-1,q) + f_2(n-1,p,q). \tag{2.3}$$

Subtracting (2.2) from (2.3) gives

$$f_2(n,p,q) = f_1(n,p,q) + f_1(n-1,p-1,q) - f_1(n-1,p-1,q-1). \tag{2.4}$$

Using (2.4) to express $f_2(n-1,p,q)$ in terms of f_1, and substituting the result in equation (2.2) gives equation (2.1). The conditions $n \geqslant 2$, $p \geqslant 1$, $q \geqslant 1$ ensure that the arguments of f_1 and f_2 are non-negative.

The recurrence formula (2.1) can be used, with suitable boundary values, to derive $f_1(n,p,q)$ for all non-negative integers n, p and q. Boundary values are now obtained by determining, for each n, the range of values of p,q for which $f_1(n,p,q)$ is zero.

<u>Range of p,q.</u>

From its definition, f_1 is only defined for non-negative integer values of its arguments. Since it is not possible to select more than n points, it follows that if $p > n$,

$$f_1(n,p,q) = 0 .$$

Also, if the n points are all chosen there are $n + 1$ adjacent pairs, and if no points are chosen there are no adjacent pairs (except in the trivial case $n = 0$, in which points 0 and $n + 1$ are adjacent). Thus, for $n \geqslant 1$,

$$f_1(n,n,q) = \begin{cases} 1 & \text{if } q = n+1 \\ 0 & \text{otherwise, and} \end{cases}$$

$$f\ (n,0,q) = \begin{cases} 1 & \text{if } q = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The remaining case $1 \leqslant p \leqslant n-1$ is now considered.

Lemma 2.2. If $1 \leqslant p \leqslant n-1$ then

$$f_1(n,p,q) = 0$$

whenever $q \geqslant p+1$ or $q \leqslant 2p-n$. That is, if $n + 2$ points lie on a line, then for every selection of $p + 2$ of these points, including both end points, the number q of adjacent pairs of selected points satisfies the inequalities

$$2p-n+1 \leqslant q \leqslant p .$$

Proof. As there is a total of $n + 1$ pairs of adjacent nodes, it follows from the definition of q that there are $n+1-q$ pairs of adjacent nodes in which at least one of the nodes is not selected. Now there are $n - p$ unselected nodes, and there are at most $2(n-p)$ pairs of adjacent nodes containing one or more of these. That is,

$$n+1-q \leqslant 2(n-p) ,$$

therefore $$q \geqslant 2p-n+1 .$$

The other inequality, namely $q \leqslant p$, follows at once from the definitions of p and q.

Table 2.1 shows the zero values of $f_1(n,p,q)$ for $n \leqslant 7$. The conditions $p \geqslant 0$, $q \geqslant 0$ define an $(n+1)\times(n+2)$ upper-left rectangle of this table.

Because of the triangular arrangement of the nonzero values of $f_1(n,p,q)$ and the nature of the recurrence formula (2.1), it is now possible to generate systematically all the values of $f_1(n,p,q)$. Before doing so, however, it is convenient to define new variables r, s as follows:

$$r = n - p \qquad p = n - r$$

$$s = p - q \qquad q = p - s = n - r - s .$$

Define

$$F_1(n,r,s) = f_1(n,n-r,n-r-s) .$$

Then the recurrence formula (2.1) becomes

$$F_1(n,r,s) = F_1(n-1,r,s) + F_1(n-1,r-1,s) - F_1(n-2,r-1,s) + F_1(n-2,r-1,s-1) . \quad (2.5)$$

Define

$$\Delta_{rs}(n) = F_1(n,r,s) - F_1(n-1,r,s) . \quad (2.6)$$

p \ q	n+1	n	n-1	n-2	n-3	n-4	n-5	n-6	n-7
n	X	o	o	o	o	o	o	o	o
n-1	o	o	X	o	o	o	o	o	o
n-2	o	o	o	X	X	o	o	o	o
n-3	o	o	o	o	X	X	X	o	o
n-4	o	o	o	o	o	X	X	X	X
n-5	o	o	o	o	o	o	X	X	X
n-6	o	o	o	o	o	o	o	X	X
n-7	o	o	o	o	o	o	o	o	X

TABLE 2.1 Zero values of $f_1(n,p,q)$ for $n \leqslant 7$. Nonzero values are indicated by X. Dotted line indicates the extent of the table for $n = 4$.

r \ s	0	1	2	3	4	5	6	7	8
1	X	o	o	o	o	o	o	o	o
2	X	X	o	o	o	o	o	o	o
3	X	X	X	o	o	o	o	o	o
4	X	X	X	X	o	o	o	o	o
5	X	X	X	X	X	o	o	o	o
6	X	X	X	X	o	o	o	o	o
7	X	X	X	o	o	o	o	o	o
8	X	X	o	o	o	o	o	o	o
9	X	o	o	o	o	o	o	o	o

TABLE 2.2 Zero values of $F_1(n,r,s)$ for $n \leqslant 9$. Dotted line indicates the extent of the nonzero values for $n = 5$.

Then (2.5) becomes

$$\Delta_{rs}(n) = \Delta_{r-1\ s}(n-1) + F_1(n-2,r-1,s-1) \qquad (2.7)$$

Equations (2.6) and (2.7) are clearly equivalent to (2.1).

The trivial case $p = n$ has already been dealt with. It is assumed now that $0 \leqslant p \leqslant n - 1$; that is, $1 \leqslant r \leqslant n$. From the restrictions on q (for nonzero $f_1(n,p,q)$), namely

$$0 \leqslant q$$
$$2p-n+1 \leqslant q \leqslant p\ ,$$

it follows that for nonzero $F_1(n,r,s)$,

$$s \leqslant n - r$$
$$0 \leqslant s \leqslant r - 1\ . \qquad (2.8)$$

The region defined by these inequalities is indicated by the nonzero values in Table 2.2, for $n \leqslant 9$. The restriction $s \leqslant n - r$ defines an upper-left triangle outside which F_1 is zero, as indicated for $n = 5$ by the dotted line on Table 2.2.

The procedure originally used to generate $F_1(n,r,s)$ was to work systematically from the top left corner of Table 2.2, calculating $\Delta_{rs}(n)$ (for all n) from equation (2.7) and then summing the $\Delta_{rs}(n)$ to obtain $F_1(n,r,s)$. The initial values used in this procedure were

$$\begin{aligned} F_1(n,1,0) &= n \\ F_1(n,r,-1) &= 0 \\ F_1(n,r,r) &= 0 \\ F_1(r+s-1,r,s) &= 0 \end{aligned} \qquad (2.9)$$

for all n, r, s. (Note that $F_1(n,r,s)$ is defined for negative as well as positive values of r, s.)

This procedure is lengthy and will not be described in further detail in view of the following result, which was suggested by numerical results obtained using the above procedure.

Lemma 2.3 For all $n \geq 1$, $r \geq 1$ and all s,

$$F_1(n,r,s) = \binom{n-r+1}{s+1}\binom{r-1}{s} . \tag{2.10}$$

Proof It may be shown (using the above recursive procedure, for example) that the recurrence relations (2.6) and (2.7) and the boundary conditions (2.9) determine $F_1(n,r,s)$ uniquely. Thus it is only necessary to prove that the above function satisfies these conditions.

(i) Boundary Conditions. From the fact that

$\binom{a}{b} = 0$ if $a < b$ or $b < 0$, (a, b integers)

it follows that $\binom{n-r+1}{s+1} = 0$ if $s > n - r$ or $s < -1$,

and $\binom{r-1}{s} = 0$ if $s > r - 1$ or $s < 0$,

from which it follows that the right hand side of (2.10) is zero unless conditions (2.8) hold. Also, when $r = 1$, $s = 0$, the right-hand side of (2.10) is equal to n (using the convention $\binom{0}{0} = 1$).

It follows that conditions (2.9) are satisfied.

(ii) Recurrence Relations. Assuming the above form for $F_1(n,r,s)$, equation (2.6) gives

$$\Delta_{rs}(n) = \binom{r-1}{s}\left\{\binom{n-r+1}{s+1}-\binom{n-r}{s+1}\right\} .$$

Applying the standard result

$$\binom{\lambda+1}{\mu} = \binom{\lambda}{\mu} + \binom{\lambda}{\mu-1} \qquad (2.11)$$

to the term in parentheses gives

$$\Delta_{rs}(n) = \binom{r-1}{s}\binom{n-r}{s} . \qquad (2.12)$$

It follows that

$$\Delta_{rs}(n) - \Delta_{r-1\ s}(n-1) = \binom{n-r}{s}\left\{\binom{r-1}{s}-\binom{r-2}{s}\right\} ,$$

and application of (2.11) to the term in parentheses gives

$$\Delta_{rs}(n) - \Delta_{r-1\ s}(n-1) = \binom{n-r}{s}\binom{r-2}{s-1}$$

$$= F_1(n-2,r-1,s-1) .$$

Thus (2.7) is satisfied, and the proof is complete.

The result of Lemma 2.3, written explicitly in terms of p,q, is

$$f_1(n,p,q) = \binom{p+1}{q}\binom{n-p-1}{p-q} , \qquad (2.13)$$

for $0 \leqslant p \leqslant n-1$, and all q. This completes the derivation of an explicit formula for $f_1(n,p,q)$ for all n,p,q.

The corresponding results for f_2 and f_3 are now derived from f_1, using the recurrence relation (2.4). For convenience the variables r,s are again used. Define

$$F_2(n,r,s) = f_2(n,n-r,n-r-s) ,$$
$$F_3(n,r,s) = f_3(n,n-r,n-r-s) .$$

In terms of r and s, equation (2.4) becomes

$$F_2(n,r,s) = F_1(n,r,s) + F_1(n-1,r,s-1) - F_1(n-1,r,s) , \tag{2.14}$$

or, using equation (2.6),

$$F_2(n,r,s) = \Delta_{r,s}(n) + F_1(n-1,r,s-1) \tag{2.15}$$

Lemma 2.4 For all $n \geq 0$, $0 \leq r \leq n$, and all s,

$$F_2(n,r,s) = \binom{r}{s}\binom{n-r}{s} \tag{2.16}$$

or, equivalently, in terms of p,q, if $0 \leq p \leq n$,

$$f_2(n,p,q) = \binom{n-p}{q}\binom{p}{n-p-q} . \tag{2.17}$$

Proof If $n \geq 1$, $r \geq 1$, equation (2.15), together with (2.10) and (2.12), gives

$$F_2(n,r,s) = \binom{r-1}{s}\binom{n-r}{s} + \binom{n-r}{s}\binom{r-1}{s-1}$$
$$= \binom{n-r}{s}\left\{\binom{r-1}{s}+\binom{r-1}{s-1}\right\} .$$

Application of the standard result (2.11) gives (2.16). The remaining case $n = r = 0$ is easily verified.

The values of p,q,r,s for which f_2 and F_2 are nonzero are found to be

$$\begin{array}{ll} 0 \leqslant p \leqslant n & 0 \leqslant r \leqslant n \\ 2p-n \leqslant q \leqslant p & 0 \leqslant s \leqslant n-r \\ 0 \leqslant q & s \leqslant r \ , \end{array}$$

using the argument of Lemma 2.2. In this case there are no exceptions to (2.16) and (2.17), unlike the corresponding results for F_1 and f_1.

<u>Lemma 2.5</u> For all $n \geqslant 1$, $0 \leqslant r \leqslant n-1$, $s \geqslant 1$,

$$F_3(n,r,s) = \binom{n-r-s}{s-1}\binom{r+1}{s} . \qquad (2.18)$$

Or, in terms of p,q, if $1 \leqslant p \leqslant n$, $q \leqslant 1-p$,

$$f_3(n,p,q) = \binom{n-p+1}{p-q}\binom{p-1}{q} . \qquad (2.19)$$

<u>Proof</u> It can be proved, using similar arguments to those in the proof of Lemma 2.1, that equations (2.2), (2.3) and (2.4) remain valid if f_1 is replaced by f_2, and f_2 by f_3. In terms of r,s, equation (2.4) becomes

$$F_3(n,r,s) = F_2(n,r,s) - F_2(n-1,r,s) + F_2(n-1,r,s-1) .$$

Substituting for F_2, using equation (2.16) and applying (2.11) twice to the result then gives equation (2.18).

The values of p,q,r,s for nonzero $f_3(n,p,q)$ and $F_3(n,r,s)$ are

$$\begin{array}{ll} 1 \leqslant p \leqslant n-1 & 0 \leqslant r \leqslant n \\ 0 \leqslant q \leqslant p & 0 \leqslant s \leqslant n-r \\ 2p-n-1 \leqslant q & s \leqslant r+1 \end{array}$$

The derivation of these inequalities is again similar to that for f_1 (see Lemma 2.2).

Comparison of the above inequalities with the restrictions in Lemma 2.5 shows that the only nonzero values of $F_3(n,r,s)$ not given by (2.18) occur when either $r = n$ or $s = 0$, that is, when either $p = 0$ or $p = q$. Now it is easily shown that $p = q$ can only occur if $p = 0$. Thus $F_3(n,n,0) = f_3(n,0,0)$ is the only nonzero value not given by (2.18), and it follows from the definition of f_3 that

$$f_3(n,0,0) = 1 .$$

This completes the solution of the problem for a linear array of points. The numbers $f_1(n,p,q)$ and $f_3(n,p,q)$ are now used to solve the problem for a circular array.

<u>Problem on a Circle</u>

Given n points arrayed on a circle, $g(n,p,q)$ is the number of selections of p of the points which include exactly q adjacent pairs of points.

It can be shown from first principles (see Lemma 2.2) that $g(n,p,q) = 0$ unless either

(i) $p = q = 0$

or (ii) $p = q = n$

or (iii) $1 \leqslant p \leqslant n-1$, $0 \leqslant q \leqslant p-1$, and $2p-n \leqslant q$. (2.20)

In the following theorem, $g(n,p,q)$ is derived for each of these three cases.

<u>Theorem 2.1</u> For all $n \geq 1$,

$$g(n,0,0) = g(n,n,n) = 1 , \tag{2.21}$$

and for $n \geq 2$, $1 \leq p \leq n-1$, and all q,

$$g(n,p,q) = \frac{n}{n-p}\binom{n-p}{p-q}\binom{p-1}{q} . \tag{2.22}$$

<u>Proof</u> The special cases (2.21) are trivial.

Suppose now that $n \geq 2$, and $1 \leq p \leq n-1$. Choose one of the n points. If a selection of p nodes with q adjacent pairs includes the chosen point, then it also includes $p-1$ of the remaining $n-1$ points. There are $f_1(n-1,p-1,q)$ such selections. If a selection of p points with q adjacent pairs does not include the chosen point, then it contains p of the remaining $n-1$ points. There are $f_3(n-1,p,q)$ such selections. Therefore

$$\begin{aligned} g(n,p,q) &= f_1(n-1,p-1,q) + f_3(n-1,p,q) \\ &= \binom{n-p-1}{p-q-1}\binom{p}{q} + \binom{n-p}{p-q}\binom{p-1}{q} \\ &= \binom{n-p}{p-q}\binom{p-1}{q}\left\{\frac{p-q}{n-p}\,\frac{p}{p-q} + 1\right\} \end{aligned}$$

which on simplification gives (2.22). Note that the value of $g(n,p,q)$ given by (2.22) is zero unless inequalities (2.20) are satisfied.

It has thus been shown that the number $g(n,p,q)$ of ways in which a tour containing n nodes may be divided into p segments, of which q consist of a single node, has the values given by (2.21) and (2.22).

2.2 Number of Distinct p-Tours

In this section the number $a(n,p)$ of distinct p-tours obtainable from an arbitrary initial tour of n nodes is derived. The results required for the purpose of this thesis are those for undirected tours. However it is convenient to consider directed tours first, and then derive the corresponding results for undirected tours.

Consider an arbitrary p-tour, $0 \leqslant p \leqslant n$. By definition there are exactly p links in this tour which do not occur in the initial tour. Deletion of these p links leaves p segments, all of which occur, possibly reversed, in the initial tour. It follows that every p-tour consists of p segments of the initial tour, arranged in some order, some subset (possibly empty) of the segments being reversed.

Thus, for example, if $n = 6$ and the initial tour is

$$(1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6),$$

the tour

$$(1 \quad 5 \quad 4 \quad 2 \quad 3 \quad 6)$$

is a 3-tour, because 3 of its links differ from those in the initial tour. It consists of the 3 segments

$$s_1 = \{6,1\}, \qquad s_2 = \{2,3\}, \qquad s_3 = \{4,5\}$$

in the order s_1, s_3, s_2, with s_3 reversed.

Conversely, if a tour is obtained from the initial tour by firstly deleting p links to divide the tour into p segments, and then rearranging these segments in such a way that none of the p deleted links are reinserted, then this tour is a p-tour.

An ordered set of p segments, formed from the initial tour by deleting p links, is defined to be a partition of the initial tour into p segments. The above results may now be summarized as follows.

Lemma 2.6 For each initial tour on a network of n nodes, the set of p-tours consists of the set of all tours obtainable from the initial tour by means of the following two steps.

Step (i). Formation of a partition of the initial tour into p segments.

Step (ii). Rearrangement of the p segments from step (i) (by means of a cyclic permutation), followed by the reversal of a subset (possibly empty) of the segments, in such a way that none of the links deleted in step (i) are reinserted.

(Note that in each partition there may be some segments consisting of a single node, and these single-node segments may not be included in any of the subsets of reversed segments. Otherwise repetitions will occur.)

<u>Corollary</u> For each initial tour on a network of n nodes, and $0 \leqslant p \leqslant n$, the number $a(n,p)$ of p-tours is the sum, over all partitions of the initial tour into p segments, of the number of ways of performing step (ii) on each partition.

If the number of ways of performing step (ii) on each partition could be found, the problem of determining $a(n,p)$ would now be solved. However, for a given partition, this number depends not only on the number of single-node segments, but also on the way in which they are distributed throughout the tour. For example, suppose there are four segments in a partition, two of the segments consisting of a single node. It can be found by trial that if the two single-node segments are adjacent then only one p-tour can be obtained from the partition, whereas if they are not adjacent then three p-tours can be obtained.

It therefore appears to be difficult, if at all possible, to use the above corollary to determine $a(n,p)$. The following alternative procedure avoids the necessity of considering the application of step (ii) to individual partitions. It is worth noting, however, that the constructive method described in section 2.3 does in fact use this approach, after step (ii) has been replaced by a more general step in which single-node segments are not treated as a special case.

The method used in the present section to evaluate $a(n,p)$ consists essentially of replacing step (ii) by another more general step, and deleting from the resulting set of tours all tours which are not p-tours. It must also be proved, of course, that this set of tours does not contain any repeated p-tours.

Let $\mathcal{B}(n,p)$ be the set of tours obtainable from a given initial tour of n nodes by means of step (i) above, followed by <u>step (ii)'</u>, which is the same as step (ii), except that the condition that no deleted link may be reinserted is omitted. Let $b(n,p)$ be the number of tours (including repetitions) contained in $\mathcal{B}(n,p)$.

<u>Derivation of $b(n,p)$</u>

If $p \geq 1$, the number of ways of rearranging the p segments of a partition is

$$(p-1)!$$

(including the trivial 'rearrangement' in which the order of the segments is left unaltered). For each of these rearrangements there are

$$2^{p-q}$$

ways of choosing a subset of segments to be reversed, where q is the number of single-node segments in the partition. Thus the number of ways of performing step (ii)' is

$$2^{p-q}(p-1)! \ . \qquad (2.23)$$

For each q, the number of partitions for which (2.23) gives the number of ways of performing step (ii)' is $g(n,p,q)$. The total number of tours obtained from the initial tour by steps (i) and (ii)' is therefore

$$b(n,p) = \sum_{q=0}^{p} 2^{p-q}(p-1)!\ g(n,p,q)$$

$$= 2^{p}(p-1)! \sum_{q=0}^{p} 2^{-q} g(n,p,q) \qquad (2.24)$$

for $1 \leqslant p \leqslant n$. If $p = 0$, i.e. no links are deleted, then the resulting tour is either the initial tour or its inverse; therefore

$$b(n,0) = 2 , \qquad (2.25)$$

if $n \geqslant 3$. In the following it will be convenient to assume $n \geqslant 3$, so that every tour is distinct from its inverse. The number $a(n,p)$ is now derived using the above values of $b(n,p)$.

<u>Derivation of $a(n,p)$</u>

It follows from the definition of step (ii)' that every tour in the set $\mathcal{B}(n,p)$ is a k-tour, where $0 \leqslant k \leqslant p$. Since, from Lemma 2.6, a tour is a p-tour if and only if none of the links deleted in step (i) are re-inserted in step (ii)', it follows that each p-tour occurs exactly once in $\mathcal{B}(n,p)$. The number of occurrences of each k-tour,

$0 \leqslant k \leqslant p-1$, is now determined.

Lemma 2.7 Every k-tour, $0 \leqslant k \leqslant p-1$, occurs in the set $\mathscr{B}(n,p)$ exactly $\binom{n-k}{p-k}$ times.

Proof Consider an arbitrary k-tour. By definition, the initial tour contains exactly k links which do not occur in the k-tour. If this k-tour can be obtained from a given partition by an application of step (ii), then the p links deleted from the initial tour to form this partition include the above k links. There are $\binom{n-k}{p-k}$ such partitions.

Since the order of the p segments contained in the k-tour is fixed, as is also the direction in which each segment is traversed, it follows that for each of the above $\binom{n-k}{p-k}$ partitions there is only one application of step (ii)' which results in the given k-tour. Thus the given k-tour occurs $\binom{n-k}{p-k}$ times.

Corollary For $n \geqslant 1$, and $0 \leqslant p \leqslant n$, the number $a(n,p)$ of distinct p-tours contained in $\mathscr{B}(n,p)$ satisfies

$$a(n,p) = b(n,p) - \sum_{k=0}^{p-1} \binom{n-k}{p-k} a(n,k) \ . \qquad (2.26)$$

Proof From Lemma 2.7, each of the $a(n,k)$ k-tours occurs in the set $\mathscr{B}(n,p)$ exactly $\binom{n-k}{p-k}$ times. The number of occurrences of k-tours in $\mathscr{B}(n,p)$, where $0 \leqslant k \leqslant p-1$, is therefore

$$\sum_{k=0}^{p-1} \binom{n-k}{p-k} a(n,k) \ .$$

Since the remaining tours are p-tours, result (2.26) follows.

Equation (2.26) may be used recursively to evaluate $a(n,p)$ for all $n \geqslant 3$ and $0 \leqslant p \leqslant n$, using the trivial result $a(n,0) = 2$ for all n as an initial value, and the values of $b(n,p)$ given by equations (2.24) and (2.25).

An explicit formula for $a(n,p)$ is derived in the following theorem. The previously derived expressions for $b(n,k)$ and $g(n,k,i)$ are included here for convenience.

<u>Theorem 2.2</u> The number of distinct p-tours obtainable from an initial tour of n nodes is given by

$$a(n,p) = \sum_{k=0}^{p} (-1)^{p-k}\binom{n-k}{p-k} b(n,k) , \qquad (2.27)$$

where

$$\begin{aligned} b(n,k) &= 2 \quad \text{if} \quad k = 0 , \\ &= 2^k (k-1)! \sum_{i=0}^{k} 2^{-i} g(n,k,i) \quad \text{if} \quad 1 \leqslant k \leqslant n , \end{aligned}$$

and

$$\begin{aligned} g(n,k,i) &= 1 \quad \text{if} \quad k = i = 0 \quad \text{or} \quad k = i = n , \\ &= \frac{n}{n-k}\binom{n-k}{k-i}\binom{k-1}{i} \quad \text{if} \quad 1 \leqslant k \leqslant n , \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

Proof From equation (2.26),

$$b(n,p) = \sum_{k=0}^{p} \binom{n-k}{p-k} a(n,k) \, . \tag{2.28}$$

The fact that (2.28) implies (2.27) is shown by proving the relation

$$\sum_{k=0}^{p} (-1)^{p-k} \binom{n-k}{p-k}\binom{n-j}{k-j} \begin{aligned} &= 1 \text{ if } p = k \\ &= 0 \text{ if } p > k \end{aligned} \tag{2.29}$$

(Equations (2.27), (2.28) and (2.29) can of course be written in matrix form). The proof of (2.29) uses the following elementary properties of binomial coefficients.

$$\binom{r}{s}\binom{s}{t} = \binom{r}{t}\binom{r-t}{s-t}, \quad t \leqslant r \leqslant s \, , \tag{2.30}$$

$$\sum_{r=0}^{s} (-1)^r \binom{s}{r} = 0, \quad \text{if} \quad s \geqslant 1. \tag{2.31}$$

The case $p = j$ is trivial. Consider the case $p \geqslant j + 1$. The left side of (2.29) may be written

$$\sum_{k=j}^{p} (-1)^{p-k} \binom{n-k}{n-p}\binom{n-j}{n-k}$$

(noting that $\binom{n-j}{n-k} = 0$ if $k < j$). This becomes, using equation (2.30),

$$\binom{n-j}{n-p} \sum_{k=j}^{p} (-1)^{p-k} \binom{p-j}{p-k}$$

which, using (2.31), is zero for $p \geqslant j + 1$. This completes the proof of Theorem 2.2, which is the main result of this section. Note that this result refers to directed tours.

Undirected tours

In the present section, in order to consider a tour as an ordered set of directed segments, it has been necessary to consider directed tours. Throughout the remainder of this thesis, attention is focussed on undirected tours, and the corresponding values of $a(n,p)$ and $b(n,p)$ for undirected tours are required. Let these values be denoted by $a_1(n,p)$ and $b_1(n,p)$, respectively. In later sections the subscripts will be omitted, except where ambiguity may occur.

The value of $b_1(n,p)$ can be found from $b(n,p)$ by observing that if a tour is obtainable from a given partition by means of step (ii)', then so is the inverse of the tour, since a tour and its inverse consist of the same segments, traversed in the opposite order and in the opposite direction. If $n \geqslant 3$, every tour is distinct from its inverse. Therefore for $n \geqslant 3$ the set $\mathcal{B}(n,p)$ consists entirely of tour-inverse pairs, each pair being equivalent to a single undirected tour. Thus

$$b_1(n,p) = \tfrac{1}{2}b(n,p) \;,$$

and likewise

$$a_1(n,p) = \tfrac{1}{2}a(n,p) .$$

The results for undirected tours are therefore as stated in Theorem 2.2 for directed tours, except that the expression for $b(n,k)$ is divided by 2. Note that equation (2.27) is unaffected when a and b are replaced by a_1 and b_1.

<u>A Check</u>

Since the number of undirected tours on a network is $\frac{1}{2}(n-1)!$, the sum over all p, $0 \leqslant p \leqslant n$, of the number of p-tours is $\frac{1}{2}(n-1)!$. It is of some value, after the lengthy foregoing derivation of $a_1(n,p)$, to verify that this result in fact holds.

From equation (2.27),

$$\sum_{p=0}^{n} a_1(n,p) = \sum_{p=0}^{n}\sum_{k=0}^{p} (-1)^{p-k}\binom{n-k}{p-k} b_1(n,k) .$$

Changing the order of summation and substituting $j = p - k$ gives

$$\sum_{p=0}^{n} a_1(n,p) = \sum_{k=0}^{n} b_1(n,k) \sum_{j=0}^{n-k} (-1)^j \binom{n-k}{j} .$$

By (2.31), the only nonzero term on the right side of this equation is the term in which $k = n$. Hence

$$\sum_{p=0}^{n} a_1(n,p) = b_1(n,n) ,$$

and therefore

$$\sum_{p=0}^{n} a_1(n,p) = \tfrac{1}{2}(n-1)! \qquad (2.32)$$

as required. This last result $b_1(n,n) = \frac{1}{2}(n-1)!$ follows from (2.24), using the fact that $g(n,n,q) = 0$ except when $q = n$.

Equation (2.32) can also be used as a check on calculations of values of $a_1(n,p)$

Particular values of $a_1(n,p)$

Sample values of $a_1(n,p)$ are given in Table 2.3, for $3 \leqslant n \leqslant 10$. Formulae for $a_1(n,p)$ in explicit polynomial form are as follows, for $0 \leqslant p \leqslant 4$, and $n \geqslant p + 1$.

$$a_1(n,0) = 1$$
$$a_1(n,1) = 0$$
$$a_1(n,2) = \tfrac{1}{2}n(n-3)$$
$$a_1(n,3) = \tfrac{1}{2}n(n-4)(2n-7)$$
$$a_1(n,4) = \tfrac{1}{24}n(n-5)(25n^2-229n+534)$$

Such formulae may be found if necessary for arbitrary p, either by expansion of (2.27) or by fitting a polynomial of degree p to the first $p + 1$ values of $a_1(n,p)$ for $n \geqslant p + 1$. These formulae are useful for hand calculation of $a_1(n,p)$ for large values of n.

n \ p	0	1	2	3	4	5	6	7	8	9	10
3	1	0	0	0							
4	1	0	2	0	0						
5	1	0	5	5	0	1					
6	1	0	9	20	15	12	3				
7	1	0	14	49	91	112	70	23			
8	1	0	20	96	302	640	740	544	177		
9	1	0	27	165	747	2439	4725	6003	4500	1553	
10	1	0	35	260	1550	7076	20810	41420	53585	41740	14963

<u>TABLE 2.3</u> Values of $a_1(n,p)$ for $3 \leqslant n \leqslant 10$, $0 \leqslant p \leqslant n$.

2.3 Constructive Enumeration of p-Tours

The number $a(n,p)$ is obtained in section 2.2 by a nonconstructive procedure involving exclusion. As mentioned following the corollary to Lemma 2.6, this procedure is made necessary by the difficulty of determining the number of ways of performing step (ii) on each partition. In the present section, step (ii) is replaced by a more general step which enables a constructive enumeration of the set of p-tours to be carried out. As a result of the generality of this constructive procedure, the set of trial tours obtained contains some tours which are not p-tours, as well as some repetitions among the p-tours. The exact number of these redundant tours is derived, and this number is shown to be comparatively small if p is small compared with n.

Before describing the procedure for constructing the set of p-tours obtainable from a given initial tour, some new notation is introduced.

p-Tests and Strict p-Tests

Consider an arbitrary partition consisting of p segments, where $p \geqslant 1$. From (2.23), the number of ways of performing step (ii)' has a maximum value of

$$u(p) = 2^{p-1}(p-1)!$$

when $q = 0$, i.e. when the partition contains no single-node segments. (In this section tours are assumed to be undirected, and under this assumption the number of ways of performing step (ii) is half the corresponding number for the case of directed tours).

In this maximal case, where every segment can be reversed, the operations comprising step (ii)' are called <u>p-tests</u>, and the corresponding operations comprising step (ii) are called <u>strict p-tests</u>. The number $v(p)$ of strict p-tests is now determined.

<u>Lemma 2.8</u> The number of strict p-tests on a partition of p segments, none of which consist of a single node, is given by

$$v(0) = 1 ,$$

$$v(p) = \sum_{k=0}^{p} (-1)^{p-k}\binom{p}{k}u(k) \qquad (2.33)$$

if $p \geqslant 1$, where

$$u(0) = 1 ,$$

$$u(k) = 2^{k-1}(k-1)!$$

if $k \geqslant 1$. Explicitly, for $p \geqslant 1$,

$$v(p) = (-1)^p + p!\sum_{k=1}^{p} (-1)^{p-k}\frac{2^{k-1}}{k(p-k)!} \quad . \qquad (2.34)$$

<u>Proof</u> The case $u(0) = v(0) = 1$ is trivial. Consider an arbitrary p-test on a given partition of p segments, where $p \geqslant 1$, and suppose the result is a k-tour, where

$0 \leq k \leq p$. As shown in the proof of Lemma 2.7, there is no other p-test (on the given partition) which results in the same k-tour. Thus the set of trial tours resulting from p-tests on a given partition contains no repetitions.

Now a k-tour can be obtained from a partition of p segments by re-inserting $p - k$ of the deleted links, and performing a strict k-test on the resulting k segments. The number of ways of re-inserting $p - k$ links is

$$\binom{p}{p-k} = \binom{p}{k} ,$$

and for each of these ways there are $v(k)$ strict k-tests. Therefore, for each k, $0 \leq k \leq p$, the number of p-tests resulting in k-tours is

$$\binom{p}{k} v(k) ,$$

and the total number of p-tests is

$$u(p) = \sum_{k=0}^{p} \binom{p}{k} v(k) . \qquad (2.35)$$

Equation (2.33) can now be derived from (2.35), using the same arguments as those in the proof of Theorem 2.2. The explicit formula (2.34) follows immediately from (2.33).

Sample values of $v(p)$ are

$$v(0) = 1$$
$$v(1) = 0$$
$$v(2) = 1$$
$$v(3) = 4$$

$$v(4) = 25$$
$$v(5) = 208$$
$$v(6) = 2121$$
$$v(7) = 25828$$

For hand calculation it is convenient to use formula (2.35) recursively, with the initial value $v(0) = 1$.

<u>Generation of p-Tours</u>

For an arbitrary initial tour of n nodes, the set of all p-tours is generated by the following two steps.

(i) Formation of a partition of p segments.

(ii)" Application of every strict p-test to each partition.

Step (ii)" differs from step (ii), its counterpart in section 2.2, in that single-node segments are no longer treated as a special case. In step (ii)", every subset of segments is reversed, whereas in step (ii), only those subsets containing no single-node segments are reversed.

Once the set of all strict p-tests is constructed, it is a routine matter (though perhaps lengthy) to apply this set of strict p-tests to every partition of the initial tour. The construction of the set of all strict p-tests has been carried out for $p \leqslant 7$ by an enumerative computer programme which makes use of the idea of congruence of p-tests (see section 2.4). The practical details of applying these p-tests efficiently to all partitions are dealt with in Chapter 3.

Number of Redundant Tours

Redundant tours result from the above procedure in two ways. Firstly, where single-node segments occur, these are reversed by some p-tests, without changing the tour. This causes repetitions of all tours, including p-tours, obtained from partitions containing a single-node segment. Secondly, although a strict p-test applied to a partition containing no single-node segments always results in a p-tour, this is not so for partitions containing single-node segments. For example a strict p-test may leave two segments in their original order, provided that one or both of the segments are reversed. If the reversed segments happen to be single-node segments then the resulting tour is not a p-tour. The total number of redundant tours from both these sources is now found, using the results of section 2.2 and the number $v(p)$ of strict p-tests.

As there are $\binom{n}{p}$ partitions, and $v(p)$ strict p-tests to each partition, the number of tours generated by steps (i) and (ii)" is

$$c(n,p) = \binom{n}{p} v(p) \tag{2.36}$$

Table 2.4 contains sample values of $c(n,p)$ for small n and p. Explicit formulae for $c(n,p)$ can be obtained from (2.36) and (2.34) for all $p \geqslant 0$ and $n \geqslant p$.

n \ p	0	1	2	3	4	5	6	7
3	1	0	3	4				
4	1	0	6	16	25			
5	1	0	10	40	125	208		
6	1	0	15	80	375	1248	2121	
7	1	0	21	140	875	4368	14847	25828
8	1	0	28	224	1750	11648	59388	206624
9	1	0	36	336	3150	26208	178164	929808
10	1	0	45	480	5250	52416	445410	3099360

<u>TABLE 2.4</u> Values of $c(n,p)$ for $3 \leqslant n \leqslant 10$ and $0 \leqslant p \leqslant 7$.

For example

$$c(n,0) = 1$$
$$c(n,1) = 0$$
$$c(n,2) = \tfrac{1}{2}n(n-1)$$
$$c(n,3) = \tfrac{2}{3}n(n-1)(n-2)$$
$$c(n,4) = \tfrac{25}{24}n(n-1)(n-2)(n-3) .$$

The number of redundant tours obtained using steps (i) and (ii)" is the difference between the total number of tours obtained and the number of distinct p-tours, namely

$$\delta(n,p) = c(n,p) - a(n,p) .$$

For example,

$$\delta(n,0) = \delta(n,1) = 0$$
$$\delta(n,2) = n$$
$$\delta(n,3) = n(3n-8)$$
$$\delta(n,4) = \tfrac{1}{2}n(17n^2-107n+210) .$$

It follows that the ratios $\frac{\delta(n,p)}{a(n,p)}$ for the above cases are, respectively

$$0 , \quad 0 , \quad \frac{1}{2n} , \quad \frac{9}{2n} , \quad \frac{204}{25n} .$$

It can be shown that the ratio $\frac{\delta(n,p)}{a(n,p)}$ is of order $\frac{1}{n}$ for all p. Thus for sufficiently large n, the proportion of redundant tours obtained using the above constructive procedure is comparatively small. If on the other hand n is not much greater than p, the number of redundant tours may be large, as can be seen from a comparison of Tables 2.3 and 2.4.

2.4 Congruence of p-Tests

The reason for introducing congruence of p-tests is to overcome the difficulty of implementing the enumerative procedure of section 2.3 in practice. It is necessary in this procedure to apply $v(p)$ strict p-tests to every partition of the initial tour. As $v(p)$ increases very rapidly with p (for example $v(4) = 25$, $v(5) = 208$, $v(6) = 2121$), the task of writing a computer programme to carry out this procedure becomes prohibitive, even for quite small values of p. The outcome of the present section is a means of classifying p-tests, and a simplified enumerative procedure in which the number of distinct p-tests to be applied at each stage is reduced by a factor of approximately p.

Consider an arbitrary partition of an initial tour into p segments, all of which may be reversed (assuming as in section 2.3 that all segments are treated alike, regardless of the fact that reversal of a single-node segment does not produce a change in the resulting tour). Let the segments be labelled consecutively

$$1, 2, 3, \ldots, p$$

where segment 1 is chosen arbitrarily. Let an arbitrary p-test be represented by the order and direction of the segments in the resulting tour as follows. Let the order of occurrence of the segments in the final tour be given by the p-cycle

$$s = (s_1 \quad s_2 \quad \ldots \quad s_p) \; ,$$

where $s_1 = 1$ for uniqueness of representation. Let the direction of each segment be specified by

$$\alpha = \{\alpha_1 \; , \alpha_2 \; , \quad \ldots \; , \quad \alpha_p\} \; ,$$

where

$$\alpha_i = \begin{cases} 1 \text{ if segment } s_i \text{ is not reversed,} \\ -1 \text{ if segment } s_i \text{ is reversed.} \end{cases}$$

For the purpose of this section let the above p-test be denoted by

$$s^{(\alpha)} = (s_1{}^{\alpha_1} \; s_2{}^{\alpha_2} \quad \ldots \quad s_p{}^{\alpha_p}),$$

and let α_i be called the <u>index</u> of segment s_i in the p-test, and α the <u>index</u> of the p-test. Thus for example the trivial p-test, in which the initial tour is unaltered, is denoted by

$$(1^1 \quad 2^1 \quad 3^1 \quad \ldots \quad p^1) \; ,$$

and its inverse, which results in the same tour traversed in the opposite direction, is

$$(1^{-1} \quad p^{-1} \quad (p-1)^{-1} \quad \ldots \quad 2^{-1}) \; .$$

Generally, the <u>inverse</u> of a p-test $s^{(\alpha)}$ is defined to be the p-test which results in the inverse of the tour resulting from $s^{(\alpha)}$, and is denoted by

$$(s^{-1})^{(-\alpha)} = (s_1^{-\alpha_1}\ s_p^{-\alpha_p}\ s_{p-1}^{-\alpha_{p-1}}\ \ .\ .\ .\ \ s_2^{-\alpha_2})\ .$$

In networks in which all tours are undirected, each p-test and its inverse result in the same undirected tour. Each p-test is therefore defined to be equivalent to its inverse, and ambiguity is avoided in the notation by specifying, for example, that $s_2 < s_p$. In numerical examples the above notation for p-tests is abbreviated by omitting superscripts $+1$, and replacing superscripts -1 by a dash. Thus the p-test $(1^{-1}\ \ 3^1\ \ 2^{-1}\ \ 5^1\ \ 4^1)$ is written $(1'\ \ 3\ \ 2'\ \ 5\ \ 4)$.

Two p-tests $s^{(\alpha)}$, $t^{(\beta)}$, on a given partition, are defined to be <u>congruent</u> if $t^{(\beta)}$ can be obtained from $s^{(\alpha)}$ by rotating the segment labels. For example the strict 6-test $(1\ \ 2'\ \ 3'\ \ 4'\ \ 5'\ \ 6')$ (see fig. 2.1, part (iv)), gives rise to the following 5 6-tests on clockwise rotation of the segment labels:

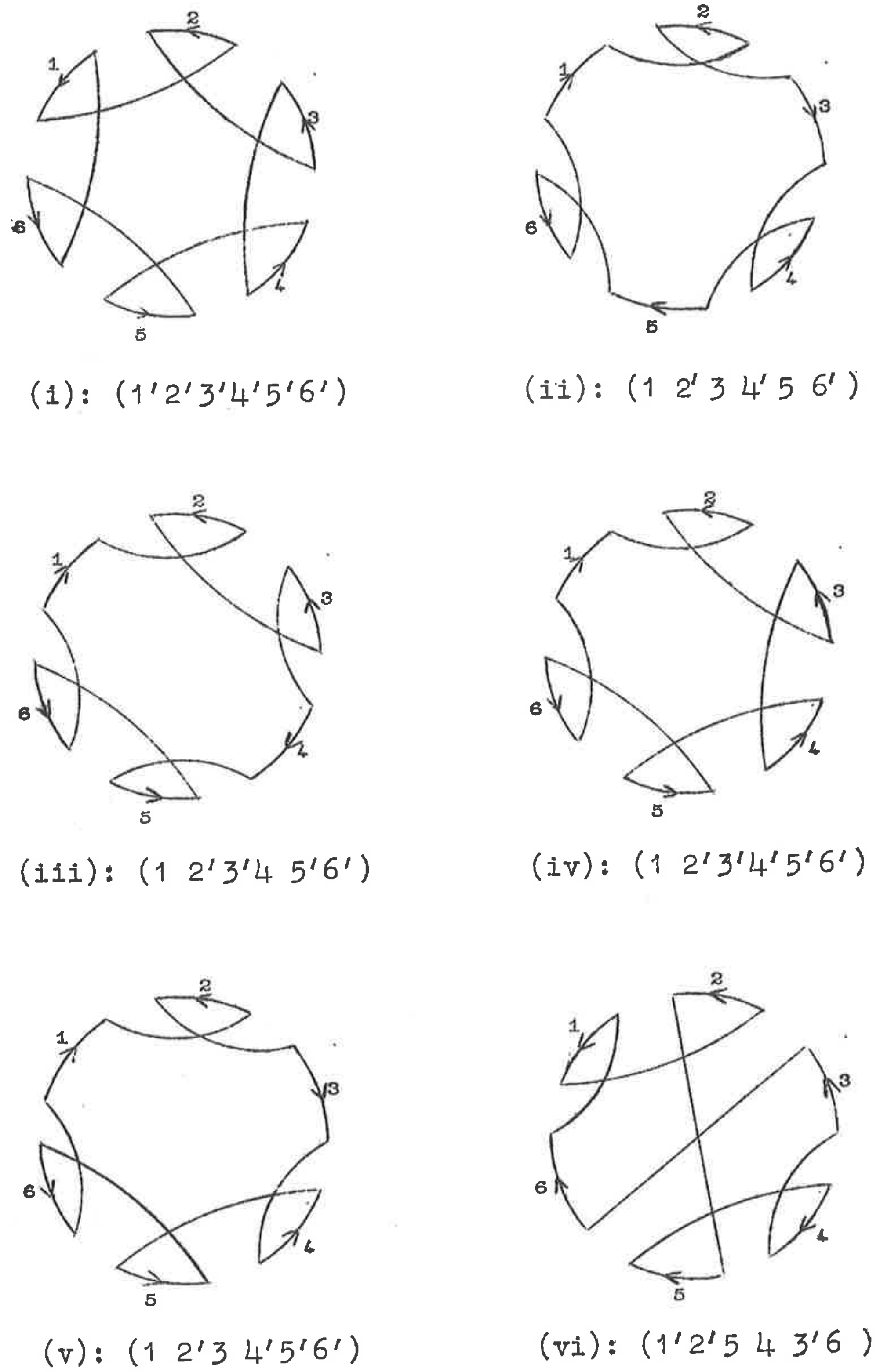
$$\begin{array}{l}
(1'\ \ 2'\ \ 3'\ \ 4'\ \ 5'\ \ 6\ ) \\
(1'\ \ 2'\ \ 3'\ \ 4'\ \ 5\ \ 6') \\
(1'\ \ 2'\ \ 3'\ \ 4\ \ 5'\ \ 6') \\
(1'\ \ 2'\ \ 3\ \ 4'\ \ 5'\ \ 6') \\
(1'\ \ 2\ \ 3'\ \ 4'\ \ 5'\ \ 6')\ .
\end{array}$$

Further examples illustrated in fig. 2.1 show the possible cases which can occur.

(i) $(1'\ 2'\ 3'\ 4'\ 5'\ 6')$ is congruent to itself only.

(ii) $(1'\ 2\ \ 3'\ 4\ \ 5'\ 6\ )$ is congruent to $(1\ 2'\ 3\ 4'\ 5\ 6')$.

(iii) $(1\ \ 2'\ 3'\ 4\ \ 5'\ 6')$ is congruent to $(1'\ 2'\ 3\ 4'\ 5'\ 6)$ and $(1'\ 2\ 3'\ 4'\ 5\ 6')$.



<u>Figure 2.1</u> Examples of strict p-tests for p=6.

(iv) (see above)

(v) (1 2′ 3 4′ 5′ 6′) is also congruent to 5 other p-tests.

Examples (i) to (v) give all the 18 strict 6-tests in which the order of the segments is unchanged. Example (vi) is another case in which a 6-test is congruent to two others. It differs from example (iii) in that after 3 rotations it is not the original tour itself but its inverse which is obtained. Tours of this form present one of the main difficulties in the enumeration of the congruence classes of p-tests in undirected networks.

Enumeration of Congruence Classes of p-Tests

One theoretically simple method of dividing the set of all p-tests into congruence classes is to generate, for each p-test, the congruence class containing the p-test and then choose a representative from this class by some rule. However, this procedure is extremely lengthy for $p > 3$, and the resulting list of representatives is not classified or ordered in any particular way. Instead, the following two-stage method is used. This method has the advantage of giving a means of classifying the congruence classes of p-tests, as well as saving computation time.

Before describing the method, the definition of congruence is put into an algebraic form, for computational purposes. Suppose the segments of an arbitrary partition are labelled 1, 2, . . . , p , and a p-test $t^{(\beta)}$ results from applying a p-test $s^{(\alpha)}$ after rotating the segment labels by adding k (modulo p) to each label. The resulting p-test is therefore, in terms of the original segment labels s_i,

$$((s_1+k)^{\alpha_1} (s_2+k)^{\alpha_2} \ . \ . \ . \ (s_p+k)^{\alpha_p}) .$$

To conform with the convention that the first segment should be segment 1, this p-test is rewritten

$$((s_j+k)^{\alpha_j} (s_{j+1}+k)^{\alpha_{j+1}} \ . \ . \ . \ (s_{j-1}+k)^{\alpha_{j-1}})$$

where j is chosen so that

$$s_j+k = 1 \pmod{p}$$

i.e. $$s_j = 1 + p - k .$$

Thus $s^{(\alpha)}$ and $t^{(\beta)}$ are congruent if there is an integer k, $1 \leqslant k \leqslant p$, such that

$$t_i = s_{j-1+i}+k \pmod{p}$$
$$\beta_i = \alpha_{j-1+i} \qquad (2.37)$$

for all i, where j is such that $s_j = 1 + p - k$. The set of all p-tests $t^{(\beta)}$ congruent to a given p-test $s^{(\alpha)}$ can therefore be generated using equations (2.37) for $k = 1,2,...,p-1$.

The two stages in the following procedure arise from the fact that for congruence of p-tests $s^{(\alpha)}$ and $t^{(\beta)}$, the conditions on the cycles s and t, and on the indices

α and β may be stated separately, as in (2.37). Let the first of these two conditions be referred to as <u>congruence</u> of the <u>cycles</u> s and t. Clearly if p-tests $s^{(\alpha)}$ and $t^{(\beta)}$ are congruent then the p-cycles s and t are congruent.

The <u>first stage</u> consists of the formation of the congruence classes of p-cycles. One representative from each class is then chosen. Where inverse pairs occur among the representatives, one of each pair is deleted (using some convenient rule).

In the <u>second stage</u> the set of all p-tests associated with each of the above representative p-cycles is generated. The resulting p-tests contain representatives of every congruence class of p-tests. Three cases can occur.

(i) If a cycle s is not congruent to its inverse, and it belongs to a congruence class of cycles containing p cycles, then it can be shown that no two p-tests $s^{(\alpha)}$ and $s^{(\beta)}$ (where $\beta \neq \alpha$) are congruent. It follows that in this case there is nothing further to be done: all the resulting p-tests belong to different congruence classes.

(ii) If a cycle s is not congruent to its inverse, and there are less than p cycles in the class containing s, then some pairs of p-tests $s^{(\alpha)}$, $s^{(\beta)}$ are congruent. If the number of cycles in the class containing s is m, then it can be shown that m is a divisor of p, and that rotation of the segment labels through a multiple of m leaves s unaltered. It follows that congruent p-tests $s^{(\alpha)}$ and $s^{(\beta)}$ can be detected by rotating the index α through multiples of m. A representative of each class of congruent p-tests is then chosen.

(iii) This case is the most difficult to deal with in practice. Fortunately only a relatively small number of p-cycles have this property. (It can be shown that this third case does not occur at all if p is odd.) If a cycle s is congruent to its inverse and there are $m(\leqslant p)$ cycles in the class containing s, then the following two steps are needed in order to detect pairs $s^{(\alpha)}$, $s^{(\beta)}$ of congruent p-tests. Firstly, if $m < p$ then each p-test needs to be rotated through multiples of m, as in case (ii). Secondly, for all m, the inverse of each p-test must also be rotated through multiples of m. (Again, as in case (ii), only the indices need to be considered in this rotation, as rotation through multiples of m leaves the cycles s and s^{-1} unaltered.) A representative of each of the resulting classes of p-tests is then chosen.

As described above, this procedure results in a set of representatives of the congruence classes of p-tests on an arbitrary partition of a tour in an undirected network.

For the purpose of generating p-tours, the corresponding result for <u>strict</u> p-tests is required. It is easily proved that in a given congruence class, either every p-test is strict or every p-test is not strict. It follows that the above procedure can be used to generate a set of representatives of the classes of strict p-tests by simply deleting all nonstrict p-tests at the beginning of the second stage.

A computer programme based on the above method was written to enumerate the congruence classes of strict p-tests for $p \leqslant 6$. The numbers of classes for $p = 3,4,5,6$ are, respectively

$$2\ ,\quad 9\ ,\quad 45\ ,\quad 363 \tag{2.38}$$

For $p = 3$, for example, there are two congruence classes of cycles, each containing only one cycle, namely $(1\ 2\ 3)$ and $(1\ 3\ 2)$ respectively. The second class contains the inverse of the cycle in the first class, and is ignored. There are four strict p-tests obtainable from $(1\ 2\ 3)$, namely

$$(1'2'3\ ),\ (1'2\ 3'),\ (1\ 2'3'),\ (1'2'3')\ .$$

Since $m = 1$ in this case, it is an example of case (ii) in the second stage of the procedure, and the result is that the first 3 3-tests are congruent and the last one is congruent only to itself. This simple example illustrates the main steps in the procedure. The results for $p = 4$ are as follows. The numbers in brackets are the numbers of elements in the respective congruence classes.

Representative cycles: (omitting inverses)

$(1\ 2\ 3\ 4)$	$(\times 1)$
$(1\ 3\ 2\ 4)$	$(\times 4)$

(note that the second class consists of two cycles and their inverses).

Representative strict p-tests:

$(1'\ 2'\ 3'\ 4')$	$(\times 1)$
$(1\ 2'\ 3'\ 4')$	$(\times 4)$
$(1\ 2'\ 3\ 4')$	$(\times 2)$
$(1\ 3\ 2\ 4')$	$(\times 4)$
$(1\ 3\ 2'\ 4)$	$(\times 4)$
$(1\ 3\ 2'\ 4')$	$(\times 4)$
$(1\ 3'\ 2'\ 4)$	$(\times 2)$
$(1\ 3'\ 2\ 4')$	$(\times 2)$
$(1'\ 3\ 2'\ 4)$	$(\times 2)$

Similar tables have been obtained for $p = 5,6$. The way in which these representatives are used is now described.

Simplified Method for Enumerating p-Tours

This method is a variation of the method of section 2.3, and consists of the following three steps.

(i) Form a partition of the initial tour into p segments, labelled consecutively $1,2, \ldots, p$.

(ii) Apply a set of representative p-tests to this partition.

(iii) Rotate the segment labels by adding 1 (mod p) and repeat step (ii). Repeat this step until segment labels have been rotated $p - 1$ times.

These steps are repeated for all partitions of the initial tour.

The number of p-tests applied to each partition in this procedure is p times the number of congruence classes; for example, for $p = 3,4,5,6,$ the numbers are

$$6\ ,\quad 36\ ,\quad 225\ ,\quad 2178\ ,$$

respectively. The number of partitions is $\binom{n}{p}$.

Sample values of the number $e(n,p)$ of resulting p-tours (including repetitions) are given in Table 2.5.

The only disadvantage of this method is that, because some congruence classes contain less than p p-tests, some p-tests are applied more than once to each partition. For example for $p = 3$, 6 3-tests are applied to each partition instead of 4. The proportion of repetitions is therefore $\frac{1}{2}$. However this is the **worst** case, and can be handled easily by the method of section 2.3. As p increases, however, and the method of section 2.3 becomes increasingly difficult to apply, the proportion of repetitions in the simplified method decreases, as a comparison of Tables 2.4 and 2.5 shows. These proportions are, for $p = 3,4,5,6,$

n \ p	0	1	2	3	4	5	6
3	1	0	3	6			
4	1	0	6	24	36		
5	1	0	10	60	180	225	
6	1	0	15	120	540	1350	2178
7	1	0	21	210	1260	4725	15246
8	1	0	28	336	2520	12600	60984
9	1	0	36	504	4536	28350	182952
10	1	0	45	720	7560	56700	457380

<u>TABLE 2.5</u> Values of $e(n,p)$ for $3 \leqslant n \leqslant 10$, $0 \leqslant p \leqslant 6$.

$$\frac{2}{4}\,,\quad \frac{11}{25}\,,\quad \frac{17}{208}\,,\quad \frac{57}{2121}\,.$$

Thus for $p \geqslant 5$, the number of repetitions due to the simplified method is very small.

The advantage of the simplified method for $p \geqslant 5$ is apparent from the number of p-tests requiring to be applied at each stage: for $p = 5$ the number is 45 instead of 208, and for $p = 6$ the number is 363 instead of 2121.

Note that the procedure used in this section to enumerate the set of congruence classes of strict p-tests relies largely on a lengthy enumeration, which is carried out by computer. Theoretical results concerning the number of congruence classes for arbitrary p (comparable, say, with the results of sections 2.1 - 2.3) are not known at the present time.

2.5 Degenerate p-Tests

The preceding sections have been devoted to the derivation of the number of distinct p-tours per initial tour, and the number of trial tours required to enumerate the p-tours in practice. These numbers are to be used in section 2.6 to determine the number of trial tours required in a purely enumerative procedure for testing a given tour for p-optimality. Although there are no powerful mathematical methods for testing a tour for p-optimality, there is one way of avoiding the need to consider certain trial tours in enumerative procedures.

Consider the following example for $n = 8$ and $p = 4$. Let the initial tour be (1 2 3 4 5 6 7 8) and the final tour be (1 7 6 4 5 3 2 8). The deleted links are (1,2), (3,4), (5,6) and (7,8), and the added links are (1,7), (6,4), (5,3) and (2,8). If the final tour is shorter than the initial tour, then

$$d_{17}+d_{64}+d_{53}+d_{28} < d_{12}+d_{34}+d_{56}+d_{78} .$$

Therefore either

$$d_{17}+d_{28} < d_{12}+d_{78} ,$$

or

$$d_{64}+d_{53} < d_{34}+d_{56} ,$$

or both. It follows that one or both of the tours

(1 7 6 5 4 3 2 8)

(1 2 3 5 4 6 7 8)

must be shorter than the initial tour. Since the above tours are both 2-tours, it follows that if the initial tour is 2-optimal, then the 4-tour (1 7 6 4 5 3 2 8) is longer than the initial tour. Thus, in testing a 2-optimal 8-node tour for 4-optimality it is not necessary to test the length of this tour. A p-test which results in a p-tour with the above property is called degenerate. Degeneracy is now discussed in detail, and the number of degenerate strict p-tests is derived, for $p \leqslant 5$.

Consider an arbitrary partition of an initial tour into p segments, none of which consist of a single node. Let the nodes be labelled so that the p deleted links, in order of their occurrence in the initial tour, are

$$\ell_1=(1,2),\ \ell_2=(3,4),\quad \ldots\ ,\ \ell_p=(2p-1,2p)$$

where node 1 is chosen arbitrarily. (Nodes not attached to a deleted link are ignored for the present.) Consider an arbitrary strict p-test consisting of the insertion of the p links

$$m_1=(i_1,i_2),\ m_2=(i_3,i_4),\quad \ldots\ ,\ m_p=(i_{2p-1},i_{2p})\ ,$$

written in order of their occurrence in the resulting p-tour, where $i_1 = 1$ for uniqueness of representation. Suppose $\mathcal{S}$ and $\mathcal{T}$ are subsets of the sets of deleted links and inserted links, respectively, and let $\mathcal{S}'$ and $\mathcal{T}'$ be the sets of nodes attached to the links in $\mathcal{S}$ and $\mathcal{T}$, respectively.

A strict p-test is defined to be <u>degenerate</u> if there exist nonempty sets $\mathcal{S}, \mathcal{T}$, as defined above, with the following properties.

(a) The sets $\mathcal{S}'$ and $\mathcal{T}'$ are identical.

(b) Replacement of the links in $\mathcal{S}$ by those in $\mathcal{T}$ results in a tour.

(c) Their complements $\sim\mathcal{S}$ and $\sim\mathcal{T}$ are nonempty, and replacement of the links in $\sim\mathcal{S}$ by those in $\sim\mathcal{T}$ results in a tour.

Let q be the number of links in $\mathcal{S}$ (or $\mathcal{T}$). Since the links in $\mathcal{T}$ belong to a strict p-tour, the links in $\mathcal{T}$ all differ from those in $\mathcal{S}$. The operation (b) is therefore a strict q-test. Similarly, the operation in (c) is a strict r-test where $r = p-q$. If the length of the initial tour is L, then the length of the p-tour is

$$L - \sum_{i=1}^{p} d(\ell_i) + \sum_{i=1}^{p} d(m_i)$$

$$= L - \sum_{\ell_i \in \mathcal{S}} d(\ell_i) + \sum_{m_i \in \mathcal{T}} d(m_i) - \sum_{\ell_i \in \sim\mathcal{S}} d(\ell_i) + \sum_{m_i \in \sim\mathcal{T}} d(m_i)$$

Therefore for this length to be less than L it is necessary that either

$$\sum_{\ell_1 \in \mathcal{S}} d(\ell_1) > \sum_{m_1 \in \mathcal{J}} d(m_1)$$

or

$$\sum_{\ell_1 \in \sim\mathcal{S}} d(\ell_1) > \sum_{m_1 \in \sim\mathcal{J}} d(m_1)$$

or both. This implies that one or both of the q-tour and r-tour resulting from steps (b) and (c) are shorter than the initial tour. The following result has thus been proved.

Lemma 2.9 If a given tour is (p-1)-optimal, then every p-tour resulting from a degenerate strict p-test is longer than the given tour.

Thus, if a tour is being tested for p-optimality by systematically applying strict q-tests, for $q = 2,3,\ldots,p$, then all degenerate strict q-tests may be omitted.

Number of degenerate strict p-tests

The problem of determining the number of degenerate strict p-tests is in general unsolved, although it is possible to obtain the number by an enumerative procedure, for sufficiently small values of p. For $p = 4$ it is easily shown by trial that the only two degenerate p-tests are

(1 7 6 4 5 3 2 8) ,

(1 3 2 4 5 7 6 8)

or, in the segment notation of section 2.4

(1′ 4 3′ 2) , (1 4′ 3 2′) .

These 4-tests are congruent under rotation, and therefore the number of degenerate strict 4-tests under congruence is 1.

For p = 5, the above sets $\mathscr{S}, \mathscr{T}$ each contain 2 or 3 links (if such sets exist at all). Since the rôles of these sets and their complements may be interchanged, it may be assumed that $\mathscr{S}$ and $\mathscr{T}$ each contain 3 links. The set of degenerate strict 5-tests is now enumerated by combining a strict 3-test and a strict 2-test on a partition of 5 segments in every possible way. Of the $\binom{5}{3} = 10$ ways of choosing 3 of the 5 deleted links,

(i) there are 5 ways in which the 2 remaining links are adjacent, and

(ii) in the other 5 ways, the 2 remaining links are separated by 1 chosen link.

In case (i) there are 4 strict 3-tests on the chosen links, and for each of these 3-tests there is one strict 2-test on the remaining 2 links. Listing the resulting 20 tours shows that they are distinct.

In case (ii) there are again 4 strict 3-tests on the chosen links; however it is only possible to apply a strict 2-test to the remaining links for 2 out of the 4 3-tests. In the remaining two, an attempt to apply a 2-test to the remaining 2 links results in the formation of two cycles.

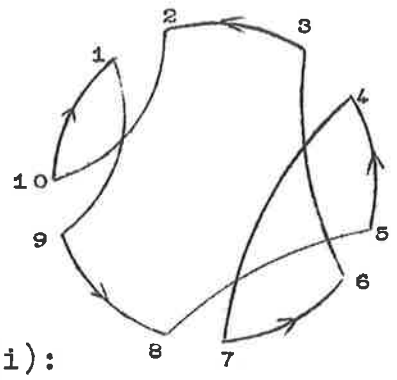
There are thus 10 5-tests resulting from case (ii), giving a total of 30 distinct degenerate strict 5-tests. Under congruence, the number of distinct degenerate strict 5-tests becomes 6. These are illustrated in Fig. 2.2. In nos. (i)-(iv) the 2-test is applied to the adjacent links (1,2) and (9,10). In (v) and (vi) the 2-test is applied to the nonadjacent links (3,4) and (7,8).

For $p = 6$ the task is more difficult owing to the fact that there may be more than one way of obtaining a 6-test as a result of applying two 3-tests. The actual number of degenerate strict 6-tests has not yet been determined; however it is known to be at least 498.

The main application of degeneracy at the moment is to reduce the number of p-tests requiring to be applied to each partition of a tour in order to test the tour for p-optimality. However it is also of interest to note that the proportion of degenerate p-tests appears to be increasing with p. For $p = 2,3,4,5,6$ the proportions are

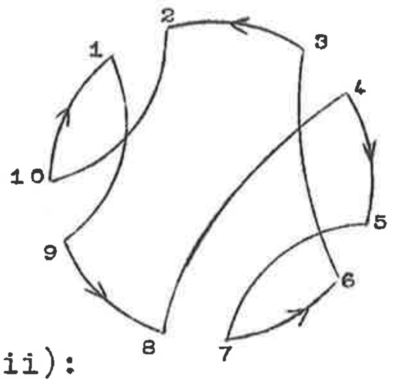
$$0 , \quad 0 , \quad \frac{2}{25} , \quad \frac{30}{208} , \quad \text{at least } \frac{498}{2121} ,$$

respectively. This suggests the possibility that for moderately large n the number of tests required to prove



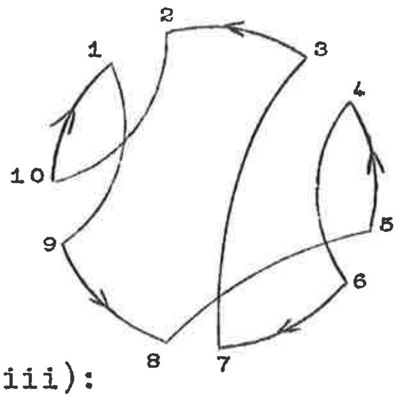
(i):

(1 9 8 5 4 7 6 3 2 10)



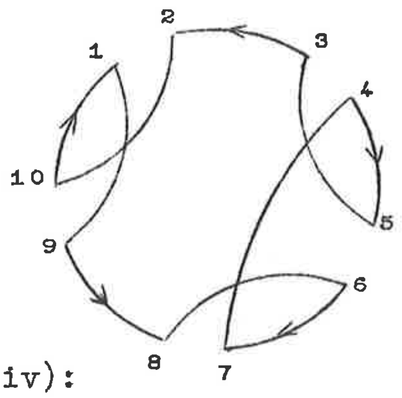
(ii):

(1 9 8 4 5 7 6 3 2 10)



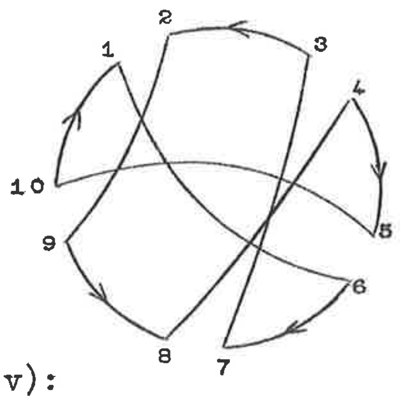
(iii):

(1 9 8 5 4 6 7 3 2 10)



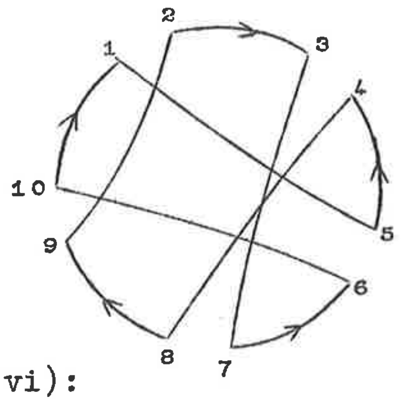
(iv):

(1 9 8 6 7 4 5 3 2 10)



(v):

(1 6 7 3 2 9 8 4 5 10)



(vi):

(1 5 4 8 9 2 3 7 6 10)

<u>Figure 2.2</u> Degenerate strict 5-tests. Each is congruent to 4 other 5-tests.

n \ p	0	1	2	3	4	5
3	1	0	3	4		
4	1	0	6	16	23	
5	1	0	10	40	115	188
6	1	0	15	80	345	1128
7	1	0	21	140	805	3948
8	1	0	28	224	1610	10528
9	1	0	36	336	2898	23688
10	1	0	45	480	4830	47376

TABLE 2.6 Number of nondegenerate p-tours obtained by constructive enumeration, for $3 \leqslant n \leqslant 10$, $0 \leqslant p \leqslant 5$.

n \ p	0	1	2	3	4	5
3	1	0	3	6		
4	1	0	6	24	32	
5	1	0	10	60	160	195
6	1	0	15	120	480	1170
7	1	0	21	210	1120	4095
8	1	0	28	336	2240	10920
9	1	0	36	504	4032	24570
10	1	0	45	720	6720	49140

TABLE 2.7 Number of nondegenerate p-tours obtained from simplified method, using congruence, for $3 \leqslant n \leqslant 10$, $0 \leqslant p \leqslant 5$.

optimality (i.e. n-optimality) may be much less than the astronomical number required in the purely enumerative procedure. The subject of degeneracy of p-tests is yet to be fully investigated.

2.6 Testing for p-Optimality

It is assumed that a tour is tested for p-optimality by comparing its length with that of every q-tour, for $0 \leqslant q \leqslant p$ (or $2 \leqslant q \leqslant p$, since the only 0-tour is the initial tour, and there are no 1-tours).

In this section, the number of distinct trial tours is first obtained from the result of Theorem 2.2. The numbers of trials required in the constructive procedures of sections 2.3 and 2.4 are then obtained and compared with the number of distinct trial tours. Finally, the number of trial tours resulting from the latter two procedures after degenerate p-tests have been omitted, is obtained, and a further comparison made. With the exception of the final results involving degeneracy, which are limited to $p \leqslant 6$, the number of trials in each case can be derived for all n and p from the results of the preceding sections. The results are illustrated by means of numerical values for $n \leqslant 10$, and explicit formulae for $p \leqslant 4$ and all n.

Number of Trial Tours

The number of distinct trial tours (excluding the initial tour) is

$$A(n,p) = \sum_{q=2}^{p} a(n,q) \qquad (2.39)$$

for all $p \geq 2$, where $a(n,q)$ is the number of distinct q-tours derived in section 2.2. Sample values are given for $n \leq 10$ and $p \leq 10$ in Table 2.8. Examples of explicit formulae obtained from (2.39) are

$$A(n,2) = \tfrac{1}{2}n(n-3)$$
$$A(n,3) = \tfrac{1}{2}n(n^2-14n+25)$$
$$A(n,4) = \tfrac{1}{24}n(25n^3-330n^2+1511n-2070) \ .$$

Enumerative Procedures

Similarly, using the constructive procedure of section 2.3 the number of trial tours is

$$C(n,p) = \sum_{q=2}^{p} c(n,q) \qquad (2.40)$$

for $p \geq 2$. Sample values of $C(n,p)$ are given in Table 2.9, and some explicit formulae derived from (2.40) are

$$C(n,2) = \tfrac{1}{2}n(n-1)$$
$$C(n,3) = \tfrac{1}{6}n(4n^2-9n+5)$$
$$C(n,4) = \tfrac{1}{24}n(25n^3-134n^2+239n-130) \ .$$

Using the simplified procedure of section 2.4, which makes use of the congruence of p-tests, the number of

n \ p	2	3	4	5	6	7	8	9	10
3	0	0							
4	2	2	2						
5	5	10	10	11					
6	9	29	44	56	59				
7	14	63	154	266	336	359			
8	20	116	418	1058	1798	2342	2519		
9	27	192	939	3378	8103	14106	18606	20159	
10	35	295	1845	8921	29731	71151	124736	166476	181439

TABLE 2.8 Values of $A(n,p)$ for $3 \leqslant n \leqslant 10$, $2 \leqslant p \leqslant n$.

n \ p	2	3	4	5	6	7
3	3	7				
4	6	22	47			
5	10	50	175	383		
6	15	95	470	1718	3839	
7	21	161	1036	5404	20251	46079
8	28	252	2002	13650	73038	279662
9	36	372	3522	29730	207894	1137702
10	45	525	5775	58191	503601	3602961

TABLE 2.9 Values of $C(n,p)$ for $3 \leq n \leq 10$ and $0 \leq p \leq 7$.

trial tours is

$$E(n,p) = \sum_{q=2}^{p} e(n,q)$$

for $p \geq 2$. Numerical values are given in Table 2.10, and some explicit formulae are

$$E(n,2) = \tfrac{1}{2}n(n-1)$$
$$E(n,3) = \tfrac{1}{2}n(2n^2-5n+3)$$
$$E(n,4) = \tfrac{1}{6}n(8n^3-42n^2+73n-39) \ .$$

Comparison of $A(n,p)$ and $C(n,p)$ shows that the number of repetitions occurring in the purely enumerative procedure is of order n^{p-1}. Since the total number of trial tours is of order n^p, the number of repetitions becomes insignificant for sufficiently large n.

The simplified method results in a somewhat larger number of repetitions. The number $[E(n,p) - A(n,p)]$ of repetitions in this case is of order n^p. However, the effect of this disadvantage is not great in practice. The proportions of repetitions, for $p = 3,4,5,6$ are asymptotically equal to

$$\frac{1}{2} , \frac{11}{25} , \frac{17}{208} , \frac{57}{2121} ,$$

respectively. The proportion therefore appears to decrease quite rapidly as p increases. It is fortunate that the worst cases, $p \leq 4$, happen to be the cases which are

n \ p	2	3	4	5	6
3	3	9			
4	6	30	66		
5	10	70	250	475	
6	15	135	675	2025	4203
7	21	231	1491	6216	21462
8	28	364	2884	15484	76468
9	36	540	5076	33426	216378
10	45	765	8325	65025	522405

TABLE 2.10 Values of $E(n,p)$ for $3 \leqslant n \leqslant 10$ and $o \leqslant p \leqslant 6$.

handled relatively easily by the first method, and also that for $p \geqslant 5$, when the simplified method becomes more important (even essential), the effect of this disadvantage becomes comparatively small.

<u>Effect of Degeneracy</u>

The effect of making use of degeneracy is now investigated for $p \leqslant 6$. If at each stage of the above enumerative procedure the degenerate q-tests are omitted, the numbers of strict q-tests for $q = 4,5,6$ are 23, 188, and at most 1623, respectively, from section 2.5. There is no change in the number of trial tours for $p \leqslant 3$. For $p = 4$ the number of trial tours becomes

$$\frac{1}{24}n(23n^3-122n^2+217n-118) .$$

Numerical values for $n \leqslant 10$ are given in Table 2.11. For all p, the reduction in the number of trial tours is of order n^p. For $p = 4,5,6$ these reductions are asymptotically equal to

$$\frac{2}{25}n^4 , \quad \frac{30}{208}n^5 , \quad \text{at least } \frac{498}{2121}n^6 ,$$

respectively.

In the simplified procedure, the omission of degenerate p-tests has a similar effect. In this case the number of trial tours for $p = 4$ is

$$\frac{1}{6}n(7n^3-38n^2+68n-37) ,$$

n \ p	2	3	4	5
3	3	7		
4	6	22	45	
5	10	50	165	353
6	15	95	440	1568
7	21	161	966	4914
8	28	252	1862	12390
9	36	372	3270	26958
10	45	525	5355	52731

TABLE 2.11 Number of trials to prove p-optimality by constructive enumeration procedure, excluding degenerate q-tours, for $3 \leqslant n \leqslant 10$, $2 \leqslant p \leqslant 5$.

n \ p	2	3	4	5
3	3	9		
4	6	30	62	
5	10	70	230	425
6	15	135	615	1785
7	21	231	1351	5446
8	28	364	2604	13524
9	36	540	4572	29142
10	45	765	7440	56580

TABLE 2.12 As for Table 2.10, for simplified procedure using congruence of p-tests.

and numerical values for $n \leqslant 10$ and $p \leqslant 5$ are given in Table 2.12. For $p \leqslant 3$ there are again no reductions, and for $p = 4,5$, the reductions are asymptotically equal to

$$\frac{1}{9}n^4 \ , \ \frac{6}{45}n^5 \ ,$$

respectively.

From the viewpoint of practical application, the results of this section may be summarized qualitatively as follows. The most efficient method for testing a tour for p-optimality is the method of section 2.3, modified by deleting degenerate p-tests. If, however, the storage and/or programming requirements of this method are prohibitive, the simplified method may be used, with only a slight decrease in computational efficiency.

On the other hand, the theoretical results themselves have some merit, as is explained in the Appendix.

CHAPTER 3

ALGORITHMS FOR THE TRAVELLING SALESMAN PROBLEM

The previously described methods for testing a tour for p-optimality enable the algorithms in this chapter to be described very briefly. Section 3.1 contains a general algorithm for generating p-optimal tours, which forms the basis of the algorithms for the Travelling Salesman Problem. Section 3.2 contains the basic algorithm in which a suitably sized set of 3-optimal tours is generated. Of importance in this algorithm is the estimation of the probability that a given set of 3-optimal tours (with repetitions) contains an optimal tour. In section 3.3 the method of section 3.2 is modified by selectively bypassing sections of the systematic tour-improvement procedure. Computational results indicate that the resulting accelerated algorithm is an efficient method for practical purposes, particularly in networks of more than 20 nodes.

3.1 Algorithm for Generating p-Optimal Tours

In the following algorithm, a systematic tour-improvement procedure is applied to a given tour, resulting in a p-optimal tour.

Suppose the partitions of a given tour into q segments $(2 \leqslant q \leqslant p)$ may be generated in some specified order. The steps of the algorithm are as follows:

1. Apply steps 2 to 7 (inclusive) for $q = 2, .., p$.
2. Generate the first partition of the current tour into q segments.
3. Apply steps 4 to 6 for each strict q-test.
4. Apply the q-test to the segments of the partition.
5. Calculate the length of the resulting tour.
6. If this tour is shorter than the current initial tour, go to 9 ; otherwise continue.
7. If all partitions into q segments have not been generated, generate the next partition and go to 3 ; otherwise continue.
8. Exit: the current tour is now p-optimal.
9. Replace the current tour by the shorter tour and go to 1.

This algorithm makes use of the direct method of section 2.3 to enumerate the p-tours of the current tour at each stage. The simplified method of section 2.4 may be incorporated in the above procedure by making the following two alterations:

(a) In step 3, instead of applying every strict q-test, apply only one representative from each congruence class of strict q-tests.

(b) Instead of generating each partition once only, generate firstly (in some order) those partitions in which a given link is deleted. There are $\binom{n-1}{q-1}$ such partitions for each q. Then repeat this process for each of the $n - 1$ other links. Clearly this amounts to rotating the segment labels in every possible way. The resulting number of partitions (with repetitions) is

$$n\binom{n-1}{q-1} = q\binom{n}{q} .$$

The above two alterations enable the simplified procedure to be obtained by simply altering the sets of partitions and strict q-tests, thus avoiding the need to physically rotate the segment labels.

The above algorithm is quite general, the only limitations to its practical application being the physical size of the set of strict q-tests, and the number of steps required. The above general statement of the algorithm does not specify how the q-tests are to be ordered, or how the tour lengths are to be compared. These questions have an important effect on computational efficiency in practice; however they need to be answered individually for each value of q. They may therefore be regarded as programming problems, and are not discussed any further in this section.

The result of the above procedure is a single p-optimal tour for each given tour. It is of course possible to obtain every p-optimal tour by repeating this procedure for a suitable set of initial tours. If such a set could be found for some p, and all the p-optimal tours generated, then an optimum tour could be found by finding the shortest p-optimal tour by inspection. However, such a set cannot be found at present. (This problem is probably as difficult as that of finding the p-optimal tours themselves.)

An alternative method for generating sets of p-optimal tours consists of applying the above algorithm to a set of randomly generated initial tours. The resulting set of p-optimal tours contains repetitions, and is not necessarily a random sample of the set of all p-optimal tours. However, because the initial tours are randomly generated, and because the p-optimal tours are obtained by applying a large number of small improvements to the tour, it seems reasonable to suppose that the resulting set of p-optimal tours does form a random sample. Computational results (see section 3.2) indicate that this assumption is acceptable, although it gives a somewhat pessimistic estimate of the probability that the shortest of a given set of p-optimal tours is optimal. It is found that shorter p-optimal tours almost invariably occur more frequently than longer p-optimal tours. Details of the estimation of this probability are given in the following section, where this method is discussed in detail for $p = 3$.

3.2 Practical Algorithm for the Travelling Salesman Problem

As a result of experiments using $p = 2,3,4$, Lin ([13], p.2263) concludes that the case $p = 3$ is the most useful for practical purposes, for medium-sized networks (up to 100 nodes). The algorithm for generating 3-optimal tours differs in detail from that of Lin, the main difference being the organization of the steps to make use of the fact that the number of steps required to invert a segment of a tour is proportional to the number of nodes in the segment. This algorithm is now described in detail, omitting programming details such as the use of temporary storage locations and the method used to invert segments.

Let $[d_{ij}]$ be the distance matrix and let a tour be represented by the permutation t, where for each node i, $t(i)$ is the node following i in the tour. The three links deleted at each stage are labelled (i_1,i_2), (j_1,j_2) and (k_1,k_2) as in Fig.3.1, in which the two 3-tests applied in this procedure are illustrated. The steps of the algorithm are as follows: (i_0 is an arbitrary initial value for i_1).

```
1.  $i_1 = i_0$
2.  $i_2 = j_1 = t(i_1)$
    $j_2 = k_1 = t(j_1)$
    $k_2 = t(k_1)$
```

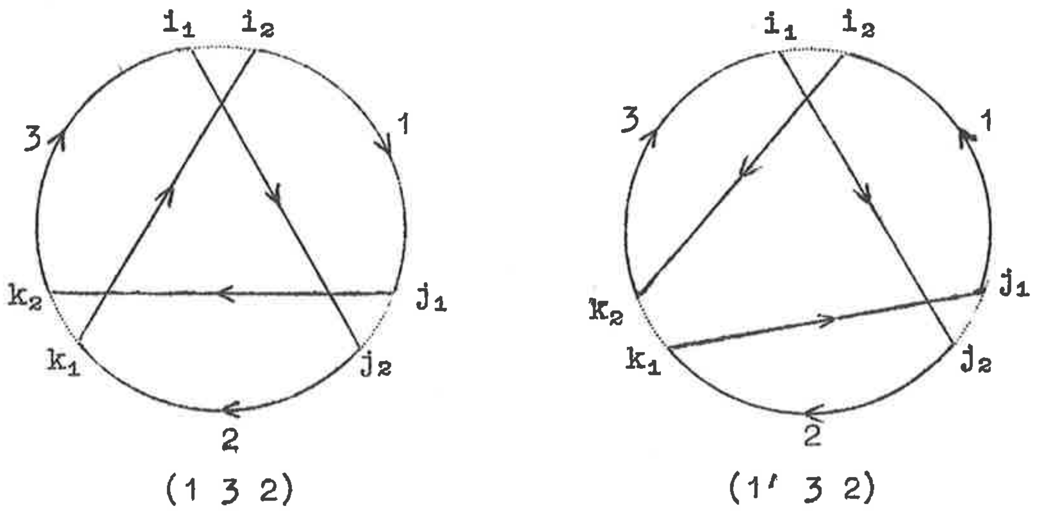


Figure 3.1: The two 3-tests applied in the practical algorithm.

```
3.   $d_1 = d_{i_2k_1} + d_{j_1k_2}$
     $d_2 = d_{i_2k_2} + d_{j_1k_1}$
4.   If $d_2 < d_1$ go to 7.
5.   If $d_{i_1i_2} + d_{j_1j_2} + d_{k_1k_2} \leqslant d_{i_1j_2} + d_1$ ,  go to 9.
6.   $t(i_1) = j_2$
     $t(k_1) = i_2$
     $t(j_1) = k_2$
        $i_0 = j_1$
        o  2.
        $d_{i_1i_2} + d_{j_1j_2} + d_{k_1k_2} \leqslant d_{i_1j_2} + d_2$ ,  go to 9.
        $_1) = j_2$
        $_1) = j_1$
        $_2) = k_2$
       vert segment from $i_2$ to $j_1$ .
        $= i_0 = j_1$
        to  2.
        $= k_2$
        $= t(k_1)$
       $k_1 \neq i_1$ ,  go to 3.
       $_1 = j_2$
       $_2 = k_1 = t(j_1)$
       $_2 = t(k_1)$
     If $k_2 \neq i_1$ ,  go to 3.
     If $i_2 = i_0$ ,  exit.   (Tour t is now 3-optimal).
     $i_1 = i_2$
     Go to 2.
```

In this description the equal sign is used in the sense of replacement. Note the 3 nested loops in the procedure: steps 2 to 12 are carried out for each value of (i_1,i_2) , steps 3 to 10 for each value of (j_1, j_2), and steps 3 to 9 for each value of (k_1, k_2).

It can be seen from Fig.3.1 that in the first iteration for each (j_1, j_2), in which $j_2 = k_1$, the 3-test (1'32) results in a 2-test applied to the links (i_1, i_2) and (k_1, k_2). Thus 2-optimality and 3-optimality are tested in the one routine. Apart from this variation, the above algorithm is that of section 3.1, using the simplified method of generating p-tours. A method for determining the number of 3-optimal tours needing to be generated in order to achieve a given probability of obtaining an optimal tour is now described.

Number of Trial Tours

Suppose firstly that the probability of obtaining every 3-optimal tour is to be greater than some given level (e.g. ·001). Suppose that the 3-optimal tours occur with equal probability, and that at some stage, ν distinct tours have been obtained in μ trials. The probability that there exists a further 3-optimal tour is smallest if the number of 3-optimal tours is $\nu + 1$, and is equal to

$$(\nu+1)\left(1 - \frac{1}{\nu+1}\right)^{\mu} = (\nu+1)\left(\frac{\nu}{\nu+1}\right)^{\mu} .$$

This is the probability that one of the $\nu + 1$ 3-optimal tours is not obtained in any of the μ trials. The number of trials required to reduce this probability below ·001, for example, is given by

$$\mu \log_{10} \nu - (\mu-1)\log_{10}(\nu+1) \leqslant -3$$

which gives

$$\mu \geqslant \frac{3 + \log(\nu+1)}{\log(\nu+1)-\log \nu} \ . \qquad (3.1)$$

Some particular values of the smallest integers $\mu(\nu)$ with this property are:

$\mu(1) = 11$	$\mu(5) = 48$
$\mu(2) = 20$	$\mu(6) = 58$
$\mu(3) = 29$	$\mu(7) = 68$
$\mu(4) = 39$	$\mu(8) = 78$.

These numbers may be incorporated in a routine for attempting to generate the set of all 3-optimal tours by specifying that if ν tours have been found, then the total number of trials is to be $\mu(\nu)$.

If only an optimal tour is required, then (3.1) becomes simply

$$\mu \geqslant \frac{3}{\log(\nu+1)-\log \nu} \qquad (3.2)$$

for a tolerance level of ·001, and for example,

$\mu(1) = 10$	$\mu(5) = 38$
$\mu(2) = 17$	$\mu(6) = 45$
$\mu(3) = 24$	$\mu(7) = 52$
$\mu(4) = 31$	$\mu(8) = 59$.

These numbers are used in a routine for attempting to generate an optimal tour in the same way as for generating the set of all 3-optimal tours.

Computational Results

Average computation times for networks of various sizes are shown in the following table. These results were obtained using a Control Data 6400 computer. Because of the probabilistic nature of the method, computation times may be expected to vary from one network to another. However, variations of more than 20% for sets of 11 or more tours are uncommon.

Number of nodes	Av. time per tour	Number of nodes	Av. time per tour
8	22 msec.	20	620 msec.
10	50 "	22	854 "
12	99 "	24	1·20 sec.
13	132 "	30	2·41 "
14	159 "	36	5·01 "
15	223 "	42	8·16 "
16	277 "	48	11·3 "
18	432 "		

The programme used to obtain these results is written in FORTRAN, apart from the procedures for generating pseudo-random initial tours and for inverting segments of a tour. A programme written entirely in assembly language should effect substantial improvements to these computation times.

The following table shows results obtained from the method for generating all 3-optimal tours for samples of randomly generated two-dimensional Euclidean networks.

Number of nodes	Number of Networks	Number of distinct 3-optimal tours			
		1	2	3	4
13	200	184	11	5	0
15	165	123	30	10	2
17	78	52	17	8	1
20	40	23	11	4	2

This table shows the numbers of randomly generated networks containing 1,2,3, ... distinct 3-optimal tours. Samples obtained for larger networks were too small to be of value. This table enables a priori estimates of the probable number of steps required in the above algorithm for generating optimal tours. For example, the percentages of 15-node networks containing 1,2,3 and 4 3-optimal tours are, respectively, 74·5%, 18·2% 6·1% and 1·2%. Using (3.2), the numbers of trials in these cases are 10, 17, 24 and 31, giving total computation times of 2·23, 3·69, 5·35 and 6·91 seconds. The probabilities of obtaining these computation times for a given random network are, from above, 0·745, 0·182, 0·061 and 0·012 respectively. The above table can be extended to include larger networks, although the amount of computation time required increases very rapidly for increasingly large networks.

3.3 Accelerated Algorithm

The accelerated algorithm is a variant of the algorithm described in the preceding section. The algorithm for generating 3-optimal tours is modified by introducing a rule whereby the innermost loop is omitted, or skipped, under certain conditions. Tours resulting from this algorithm are not necessarily 3-optimal; however if a suitable rule is chosen, the probability of resulting tours being 3-optimal can be very high.

There are many possible ways in which steps may be skipped. For example, Lin [13] "flags" each link which occurs in a 3-optimal tour, and in his procedure, if the link (i_1, i_2) is flagged, both the inner loops are skipped. This results in a great decrease in computation time. However, the skipping procedure is so severe that there is little chance of the resulting tour being 3-optimal. Lin therefore applies his standard 3-optimal algorithm to the resulting tour. A further disadvantage is the fact that no skipping is possible until at least one 3-optimal tour has been found. In the following method the first difficulty is overcome by using a less severe skipping procedure, and the second is overcome by flagging certain short links a priori.

Firstly, a link (i,j) is flagged if it is one of the shortest 3 links attached to node i or j. (The number 3 is not essential to this procedure; however it is found that for randomly generated networks in 2 dimensions, this gives the best results. For specialized applications a different number may be preferable.)

Secondly, in the algorithm of section 3.2, the innermost loop (in which (k_1,k_2) is incremented) is skipped if the link (i_1,j_2) is not flagged (see Fig. 3.1). This choice of rule may be justified as follows. Firstly, link (i_1,j_2) is the only inserted link common to both 3-tests for every choice of (k_1,k_2). Secondly, in the initial stages of the procedure, when the current tour still contains a large number of relatively long links, only substantial improvements are made, thus saving, possibly, a number of steps in which only small improvements are made. Also, near the end of the procedure, when the tour contains a large number of relatively short links, it appears unlikely that the tour will be improved by inserting an unflagged link. Finally, the repetition due to the fact that the first p-test is congruent to itself ensures that this p-test is applied to every partition unless all three of the inserted links are unflagged.

In spite of the above reasons in favour of the above skipping rule, any of a number of other rules could prove in practice to be just as effective. Generally a rule consists of: Skip the innermost loop and/or the second loop if certain of the deleted links are flagged and/or certain of the inserted links are unflagged. The many resulting rules differ in their relative effects at the start and finish of the procedure. Some combinations may be dismissed at the outset as absurd (e.g. "skip both loops if any one of the deleted links is flagged." Using this rule it is impossible to obtain a tour containing a large proportion of flagged links, unless there happens to be a large number of steps in which 3 unflagged links are replaced by 3 flagged links.) However there remain several rules which appear feasible, and the only way to compare these rules is to compare experimental results.

From computations carried out on networks of 48 nodes, it appears that the above rule is the most effective in practice, combining computational speed with a high probability that the resulting tour is 3-optimal. A second rule also gave good results in that the computation was three times as fast; however the proportion of resulting 3-optimal tours was about one third that of the above method. This rule is: Skip the innermost loop if both (i_1, i_2) and (j_1, j_2) are flagged. The added computational speed appears to result from the fact that this rule causes much of the lengthy final stages to be skipped. This may also be the source of the decreased probability of 3-optimality. This second rule may be of value for very large problems in practice, in cases where computational speed is of primary importance, and where the need is only for a tour which is reasonably short.

Average computation times for the accelerated algorithm (using the former rule for the skipping procedure) are as follows:

Number of nodes	Av. time per tour	Number of nodes	Av. time per tour
10	25 msec.	35	830 msec.
15	85 "	40	1·1 sec.
20	190 "	45	1·2 "
25	330 "	50	1·4 "
30	550 "		

Much wider variations in computational speed were obtained with this method. In some cases times varied between 0·5 and 1·8 times the average.

In the accelerated algorithm, no alteration is made to the procedure for determining the number of trials required in order to achieve a given probability of obtaining an optimum. This is exactly as described in section 3.2.

Note that as the number of nodes increases, the ratio of the computation times decreases sharply. This decrease more than offsets the increase in the number of trials resulting from the presence of tours which are not 3-optimal. For example, in the 48-node network of Held and Karp [10], the average time per tour is 0·81 seconds, while the increase in the number of trial tours is less than 30%. (The low computation time in this case is probably due to the presence of a large number of flagged links in the optimal tour).

The accelerated algorithm in this section serves a twofold purpose for practical applications. Firstly, it supplies a method of obtaining individual short tours in an extremely short time, even for large networks, for applications in which computation time is of primary importance. Secondly, it provides an improved method for obtaining optimal or near-optimal tours, for applications in which the length of the resulting tour is of primary importance. It is particularly applicable to large networks.

CHAPTER 4

DISCUSSION

This thesis contains contributions, of both a theoretical and a practical nature, to the subject of p-optimality of tours in a network, and its application to the Travelling Salesman Problem.

The main theoretical results are the combinatorial results proved in sections 2.1, 2.2 and 2.3, in particular Theorems 2.1 and 2.2, and Lemma 2.8. Apart from their application in section 2.6 to the determination of the number of trials required to test the p-optimality of a tour, these results are believed by the Author to be of some merit in themselves, in the field of combinatorial analysis. The general combinatorial significance of these results is demonstrated by the general statements in the Appendix. To the Author's knowledge, there has previously been no investigation into the permutations and combinations of points on a circle in which a fixed number k of adjacent pairs of points remain adjacent, except for the special case $k = 0$. (For $k = 0$, see Kaplansky [11] for a result on combinations, and Riordan [18] for results on permutations).

Throughout the text, a number of unsolved problems are mentioned. Firstly, in section 2.4 it is noted that there is no theoretical method for determining the number of congruence classes of strict p-tests. Secondly, present results concerning the number of degenerate strict p-tests are limited to $p \leqslant 5$. Also, the number of distinct p-tours resulting from nondegenerate strict p-tests is not known. (This suggests the extension of the idea of degeneracy to p-tours instead of p-tests.) Problems involving the lengths of tours in a network are likely to be even more difficult, as results will generally depend on some property of the distance matrix. There is for example the determination of p' in a given network (or possibly bounds on p', in general) such that p'-optimality of a tour implies optimality. Also, (see section 3.1) there is no way of avoiding repetitions when generating a set of p-optimal tours in a network. Finally, there is of course the problem of generating p-optimal tours for large p. The solution of this problem depends on the determination of a method for generating p-tours which does not depend largely on enumeration. Some, at least, of these problems should provide fruitful areas for further research.

The main results of a practical nature are the improved estimate of the probability that a given set of p-optimal tours contains an optimum (see section 3.1 and 3.2), and the accelerated algorithm of section 3.3. This algorithm enables good approximate solutions to Travelling Salesman problems to be obtained for networks containing up to 145 nodes (using a computer with a 32K core). Thus, together with Lin's method, the above method handles larger problems than other known methods. For problems containing less than 50 nodes the computational efficiency of the accelerated algorithm allows a very high probability of obtaining an optimum tour to be reached in a comparatively short time.

Despite the extremely good practical results obtained with the accelerated algorithm, it remains true that optimality cannot be proved using this method. However, as solutions to the above (and other) problems are found, better methods of applying p-optimality to the Travelling Salesman Problem may be developed, methods which may eventually lead to a method of obtaining optimal solutions.

APPENDIX

THE COMBINATORIAL THEOREMS OF CHAPTER 2

With the exception of the result of section 2.1, the combinatorial results of chapter 2 are stated and proved using the notation of tours in a network. The results may however be restated in such a way as to illuminate their combinatorial nature.

Theorem 2.2 may be restated as follows. Let $\alpha(n,r)$ be the number of ways of permuting n points on a circle in such a way that exactly r adjacent pairs of points remain adjacent. Then

$$\alpha(n,r) = a(n,n-r) ,$$

where $a(n,p)$ is given by equation (2.27). Alternatively $\alpha(n,r)$ is the number of cyclic permutations on $\{1,2, \ldots, n\}$ in which there occur exactly r pairs of the form $i,i+1$ or $i,i-1$ (mod n). This is a generalization of a result due to Riordan [18], who derived a recurrence formula for the case $r = 0$.

The main result of section 2.3, namely Lemma 2.8, may be restated as follows, using the obvious notation of 'dominoes' for convenience. Suppose p dominoes are labelled $(1,2)$, $(2,3)$,...$(p-1,p)$, $(p,1)$. The number of ways of arranging these dominoes on the circumference of a circle such that no two equal numbers are adjacent is

$2v(p)$, where $v(p)$ is given by equations (2.33) and (2.34). This result may also be stated as a variation on the well-known 'Problème des Ménages' (see Riordan [17], chapter 8): p married couples at a party have been dancing, no man dancing with his wife. The number of ways of seating the dancing partners at a circular table in such a way that no man is seated next to his wife is $2v(p)$. Note that there is no restriction here that men and women should sit in alternate positions. If this restriction is added, then it can be shown that the number of ways becomes, for $p \geqslant 1$,

$$w(p) = 2 \sum_{k=0}^{p} (-1)^{p-k} \binom{p}{k} x(k) ,$$

where

$$x(0) = 1$$
$$x(k) = (k-1)!$$

(c.f. equation (2.33)). Also, the number of p-cycles on $\{1,2, \ldots, p\}$ containing no pairs $(i,i+1)$ (mod p) is $\frac{1}{2}w(p)$.

It is interesting to note that the result of Kaplansky [11], of which Theorem 2.1 is a generalization, may also be generalized in two further ways (see Lagrange [12] and Abrahamson [1]), yet these two generalizations do not appear to be related to the result of Theorem 2.1 in any other way.

The following is an incidental result which follows from Theorem 2.2: The number of ways of placing n numbers i, $2 \leqslant i \leqslant n-2$, on a circle such that

(a) the sum of the n numbers is a multiple of n, and

(b) the sum of any contiguous subset of less than n of the numbers is not a multiple of n,

is $a(n,n)$, given by (2.27). This result is obtained by considering the differences (mod n) of the node numbers in strict n-tests on a network of n nodes.

A further topic which is of some interest in connection with the above problems is that of the generating functions associated with the above numbers. Note that for all the above problems, explicit solutions are obtained, as well as recurrence formulae.

REFERENCES

1. M. Abrahamson, "Explicit Expressions for a Class of Permutation Problems", Canad. Math. Bull. 7 (1964) 345-350.

2. E.L. Arnoff and S.S. Sengupta, "The Travelling Salesman Problem", Progress in Operations Research, Vol.I, (R.L. Ackoff, ed.), Wiley, New York, 1961.

3. R. Bellman, "Dynamic Programming Treatment of the Travelling Salesman Problem", J.ACM, 9 (1962) 61-63.

4. M. Bellmore and G. Nemhauser, "An Analysis of Algorithms for the Travelling Salesman Problem", unpublished paper; abstract in ORSA Bull. 14 (1966) supp.2.

5. F. Bock, "Mathematical Programming Solution of Travelling Salesman Examples", Recent Advances in Mathematical Programming, (R.L. Graves and P. Wolfe, ed.), McGraw-Hill, New York, 1963.

6. G.A. Croes, "A Method for Solving Travelling Salesman Problems", Opns Res. 6 (1958) 791-812.

7. G. Dantzig, D. Fulkerson and S. Johnson, "Solution of a Large Scale Travelling Salesman Problem", Opns Res. 2 (1954) 393-410.

8. M. Flood, "The Travelling Salesman Problem", Opns Res. 4 (1956) 61-75.

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

10. M. Held and R. Karp, "A Dynamic Programming Approach to Sequencing Problems", J.SIAM. 10 (1962) 196-210.

11. I. Kaplansky, "Solution of the 'Problème des Ménages'", Bull. A.M.S. 49 (1943) 784-785.

12. R. Lagrange, "Sur les Combinaisons d'Objets Numérotés", Bull. des Sci. Mathématiques (2) 87 (1963), 1 ière partie, p.29-42.

13. S. Lin, "Computer Solutions of the Travelling Salesman Problem", Bell System Tech. J. 44 (1965) 2245-2269.

14. J.D.C. Little, K.G. Murty, D.W. Sweeney, C. Karel, "An Algorithm for the Travelling Salesman Problem", Opns Res. 11 (1963) 972-989.

15. G.T. Martin, "Solving the Travelling Salesman Problem by Integer Linear Programming", CEIR, New York, 1966.

16. K. Menger, "Botenproblem", Ergebnisse eines Mathematischen Kolloquiums, Heft 2 (K. Menger, ed.) B.G. Teubner, 1932.

17. J. Riordan, "An Introduction to Combinatorial Analysis", Wiley, New York, 1958.

18. J. Riordan, "A Recurrence for Permutations Without Rising or Falling Successions", Ann. Math. Statist. 36 (1965) 708-710.

19. S.M. Roberts and B. Flores, "An Engineering Approach to the Travelling Salesman Problem", Man. Sci. 13 (1966) 269-288.

20. D. Shapiro, "Algorithms for the Solution of the Optimal Cost Travelling Salesman Problem", Sc.D. Thesis, Washington University, St. Louis, 1966.

21. J. Staudhammer and M. Ash, "A Sufficiency Solution of the Travelling Salesman Problem", System Development Corp., Santa Monica, Calif., 1966.

22. D.W. Sweeney, "The Exploration of a New Algorithm for Solving the Travelling Salesman Problem", M.S. Thesis, M.I.T., 1963.

23. S. Reiter and G. Sherman, "Discrete Optimizing ", J. SIAM. 13 (1965) P64-889.