Worst-case Analysis of Algorithms for Travelling Salesman Problems

A. M. Frieze, London

ABSTRACT

This paper is principally concerned with analysing some heuristics for the symmetric travelling salesman problem. In particular we consider the ratio R of the length of a tour generated by each algorithm to that of a minimum tour.

The first algorithm generates a tour by adding the shortest available edge. We demonstrate an upper bound for R which is $O(\log_2 n)$ where n is the number of cities. We also give a sequence of problems for which R grows as $O(\log_2 n/\log_2 \log_2 n)$.

The next algorithm proceeds by altering a minimum spanning 1-tree until it is a tour. We prove that $R \leq 2$ for this method and give a sequence of problems for which $R=2-3/n$.

A modification leads to $R<(2-k/n)$ in $O(n^{3+k})$ time for $1 \leq k < n-1$. Applying this idea to the maximum length tour problem gives $R \geq (2/3+(k-3)/3n)$ in $O(n^{4+k})$ time for symmetric problems and $R \geq (1/2+(k-2)/2n)$ for non-symmetric problems

INTRODUCTION

The travelling salesman problem has been studied for a long time. There are several versions of this problem and in this paper we mainly consider the following: let G=(N,E) be a complete undirected graph on n nodes. Associated with an edge e=(i,j) we have a non-negative length denoted by l(e) or l(i,j) which is assumed to satisfy the triangular inequality

(1) $l(i,j)+l(j,k) \geq l(i,k)$

for any nodes i,j,k.

A tour T is a cycle in G which goes through each node exactly once. The length of T is the sum of the lengths of the edges in T. The problem is to find the minimum length tour.

Known algorithms for solving this problem can take time exponential in n. The problem is in fact NP-hard in the sense that if a polynomial time algorithm exists for its solution then a polynomial time algorithm exists for a wide class of difficult combinatorial problems - see Cook [3] and Karp [12]. As a consequence it is generally accepted that no such algorithm is likely to exist. This has lead to an interest in approximate algorithms which take time polynomial in n but which do not guarantee an optimal solution but are likely to lead to good solutions. Given such an algorithm it is of interest to bound the ratio of the obtained tour length to optimal tour length. This was done for several algorithms by Rosenkrantz, Stearns and Lewis [15] where a best bound of 2 was found for their nearest insertion algorithm. Christofides [1] describes an algorithm for which this bound is 3/2- see also Cornuéjols and Nemhauser [4].

The first part of this paper analyses a greedy algorithm and shows it has a worst case performance which can become arbitrarily bad as n increases. This is similar to that found for the nearest neighbour algorithm in [15].

We then provide a 'hard' problem for the Clarke and Wright [2] savings algorithm.

We then analyse an algorithm which begins with a minimum spanning 1-tree and alters it until it is a tour. We show that this tour is no more than twice the length of the minimum tour.

We then describe a modification of this method which produces in $O(n^{3+k})$ time a tour whose length is no more than (2-k/n) times the minimum length of a tour for $1 \leq k \leq n-2$.

These ideas are then applied to the problem of finding a maximum length tour. For the symmetric problem we describe an algorithm which produces a tour of length at least (2/3+(k-3)/3n) times the length of the longest tour. The time complexity of the algorithm is $O(n^{4+k})$ for $1 \leq k \leq n-2$.

For unsymmetric problems the ratio is (1/2+(k-2)/2n) with the same time complexity.

A GREEDY ALGORITHM

We first introduce some notation:
For a finite set A we let $|A|$ = the number of elements in A. For a real number x $\lfloor x \rfloor$ is the largest integer$\leq x$ and $\lceil x \rceil$ is the smallest integer$\geq x$.
The union of two disjoint sets A,B will be written A+B and for $s \notin A$ A+s denotes A+{s}.
For a set $A \subseteq E$ we let $l(A)=\sum_{e \in A} l(e)$

A tour T can be viewed as a set of edges as well as a path. We define a family F of subsets of E as follows: $S \in F$ if and only if there is some tour T such that $S \subseteq T$. The pair (E,F) forms what is called an independence system. Sets in F are called independent sets and a maximal independent set is called a basis.
The travelling salesman problem is then to find a minimum length basis. A natural approach to this problem is the greedy method.
(Initialise) S:=$\emptyset$
(General Step) While S is not maximal S:=S+e* where $l(e^*)=\min(l(e):e \notin S$ and $S+e \in F)$.
On termination S is the tour output by the algorithm.
The term greedy was coined by Edmonds [6] in relation to matroids. Analyses of the greedy algorithm for general independence systems are given in Korte and Hausmann [13] and Jenkyns [11].
If a set of edges A is in F then it can be considered to make a set of node disjoint paths PATHS(A) and p=p(A) will denote the number of paths. Some paths will consist of single edges and will be called edge paths. Let the number of such paths be q=q(A) and let the number of non-edge paths be r=r(A) where r=p-q. We partition A into A_1, A_2 where A_1 is the set of edges making up the non-edge paths. Finally let I=I(A) be the set of nodes lying on a non-edge path but not an end point of it's path.

We now prove a lemma

Lemma 1

If $A,B \varepsilon F$ and $|B|>2|A| - p(A)$ then there exists $e \varepsilon B-A$ such that $A+e \varepsilon F$.

Proof

Suppose $e=(i,j) \varepsilon B-A$ and $A+e \notin F$ then we can deduce that either (i) $\{i,j\} \cap I \neq \emptyset$ or (ii) i,j are the end points of some non-edge path in PATHS(A).

Since $B \varepsilon F$ case (i) can occur at most twice with the same i. Further if $i \varepsilon I$ and $(i,k) \varepsilon B \cap A_1$ for some k then case(i) can occur at most once with this i. Case (ii) can occur at most r times and so if no $e \varepsilon B-A$ satisfies $A+e \varepsilon F$ then

$$(2)\quad |B-A| \leq 2|I| + r - |B \cap A_1|$$

Now one can see that $|A| = |I| + p$ and so (2) can be expressed

$$(3)\quad |B-A| \leq 2|A| - q - p - |B \cap A_1|$$

Now $|A_2| = q$ and so $|B \cap A_2| \leq q$ and so

$$\begin{aligned} |B| &= |B-A| + |B \cap A_1| + |B \cap A_2| \\ &\leq 2|A| - q - p + q \\ &= 2|A| - p \end{aligned}$$

and hence the result. ∎

In the theorem below we let GREEDY denote the length of some tour found by the algorithm and let OPTIMAL denote the minimum length of a tour.

Theorem 1

$$\text{GREEDY} \leq ((25/12)+(1+\lceil \log_2((n-8)/2)/\log_2(5/4) \rceil)/5)\text{OPTIMAL}$$

Proof

Suppose that the greedy algorithm selects edges $T = \{e_1,\ldots e_n\}$ in this order and let $A_k=\{e_1,\ldots e_k\}$. Next let $e_1^*,\ldots e_n^*$ be the edges in some optimal tour where $l(e_k^*) \leq l(e_{k+1}^*)$ and define $A_k^*=\{e_1^*,\ldots e_k^*\}$. Define also $d(k)=l(e_k)$ and $d^*(k)=l(e_k^*)$.

It follows from lemma 1 with $A=A_k$ and $B=A^*_{2k+1-p(A_k)}$ and the definition of the greedy algorithm that

(4) $d^*(2k+1-p(A_k)) \geq d(k+1)$

Since $p(A_k) \geq 1$ for $k > 0$ we can deduce from (4) that $d^*(2k) \geq d(k+1)$ and as $d^*(2k+1) \geq d^*(2k)$ we can combine these latter inequalities into

(5) $d^*(j) \geq d(\lfloor j/2 \rfloor + 1)$ for $j=1,\ldots n$.

Summing these inequalities gives

$OPTIMAL \geq d(1) + 2(d(2) + \ldots + d(\lceil n/2 \rceil)) + f$

where $f = 0$ if n is odd and $f = d(\lfloor n/2 \rfloor + 1)$ if n is even.

It follows in either case that

(6) $OPTIMAL \geq 2(d(1) + \ldots + d(\lfloor n/2 \rfloor))$

We now have to construct similar inequalities for the remaining edges of the greedy tour. To do this we define for $m>0$ G_m to be the complete graph with nodes $N-I(A_m)$ and edges E_m. Edge lengths are defined by restricting l to E_m. Let F_m denote the family of subsets of tours of G_m. It follows from the triangular inequality (1) that the minimum length of a tour in G_m is $\leq$ OPTIMAL.

Next re-define $e_1^*,\ldots e_s^*$ to be the edges of an optimal tour T_m^* in G_m where $s = |N-I(A_m)|$ and where $l(e_k^*) \leq l(e_{k+1}^*)$.

Define $\tilde{A}_m \subseteq E_m$ to consist of the edge paths of $PATHS(A_m)$ plus the edges joining the endpoints of its non-edge paths and let $\hat{A}_k = \tilde{A}_m + \{e_{m+1},\ldots,e_k\}$.

The edges $e_{m+1},\ldots e_n$ when added to $\tilde{A}_m$ give a tour in G_m which starting from $\tilde{A}_m$ could have been produced by the greedy algorithm.

We now demonstrate a lower bound on the total length of those edges in T_m^* which do not belong to A_m.

Let $e^*_{i_t}$ be the t'th shortest edge of T_m^* which is not in A_m and let $d^*(t)$ be re-defined to be $l(e^*_{i_t})$. For a given $k \geq m$ define $w=w(k)=2|\hat{A}_k|+1-q(\hat{A}_k)-p(\hat{A}_k)$. Then with $A = \hat{A}_k$ and $B = \{e_{i_1}^*,\ldots e_{i_w}^*\}$ we deduce from (3) that

there exists $e \in B-A$ such that $A+e \in F_m$. (A lower bound for k such that B exists is $m+\lfloor (n-m-1)/5 \rfloor$ as shown in (8)).
The definition of the greedy algorithm then implies

(7) $d^*(2\,|\hat{A}_k| + 1 - q(\hat{A}_k) - p(\hat{A}_k)) \geq d(k+1)$

Putting $t=k-m$ we see that since $\tilde{A}_m$ consists of node-disjoint edges that

(i) $p(\hat{A}_k) \geq |\tilde{A}_m| - t$

(ii) $q(\hat{A}_k) \geq |\tilde{A}_m| - 2t$

and as (iii) $|\hat{A}_k| = |\tilde{A}_m| + t$

we see that

$2\,|\hat{A}_k| + 1 - q(\hat{A}_k) - p(\hat{A}_k) \leq 5t+1$

and hence from (7) and the monotonicity of d^* that

(8) $d^*(5t+1) \geq d(m+t+1)$

and then that

(9) $d^*(t) \geq d(m+\lfloor (t-1)/5 \rfloor + 1)$ for $1 \leq t \leq n-m$

Now $|T_m^* - \tilde{A}_m| \geq |T_m^*| - |\tilde{A}_m|$

$= n - |I(A_m)| - |PATHS(A_m)|$

$= n - m$

Thus we can deduce that for $n-m \geq 5$

(10) $OPTIMAL \geq d^*(1) + \ldots + d^*(n-m)$

$\geq 5(d(m+1) + \ldots + d(m+\lfloor (n-m)/5 \rfloor))$

and that for $4 \geq n-m \geq 2$

(11) $OPTIMAL \geq (n-m)(d(m+1))$

and that finally

(12) $OPTIMAL \geq 2d(n)$

as no edge can be longer than half the length of any tour from (1).
We complete the theorem by a partitioning of T and a use of (6),(10),(11),(12).
We define the following sequence $u_1, u_2, \ldots$ of positive integers by $u_1 = \lfloor n/2 \rfloor$ and

$u_{k+1} = u_k + \lfloor (n - u_k)/5 \rfloor$ for $k \geq 1$

Now one can show that for $n \geq 9$ there exists an integer t such that $u = n-5$ (for $n \leq 8$ we can take $t=0$ in (13)) $n \leq 8$ we can take $s=0$ in (13)). If we partition T into $\{e_1,\ldots e_{u_1}\},\{e_{u_1+1},\ldots e_{u_2}\},\ldots \{e_{u_t+1},\ldots e_{u_{t+1}}\}$, $\{e_{n-3},\ldots e_n\}$ then one can see that the total length of the edges in the first set is $\leq$ OPTIMAL/2 and in each of the next t sets is $\leq$ OPTIMAL/5 and in the last set is $\leq (1/4 + 1/3 + 1/2 + 1/2)$OPTIMAL $=(19/12)$ OPTIMAL.
Hence we have

(13) GREEDY $\leq (t/5 + (25/12))$OPTIMAL

It remains to get an upper bound for t in terms of n.
If we define real numbers $v_1,v_2,\ldots$ by $v_1=u_1$ and

(14) $v_{k+1} = v_k + (n - v_k)/5 - 4/5$

then a straightforward induction shows that $v_k \leq u_k$ for $k \geq 1$. The solution of recurrence relation (14) is

$v_k = (n - 4) - (4/5)^{k-1}(n-4-n/2)$

Now $v_k \leq n-5 \rightarrow (4/5)^{k-1}(n/2 -4) \geq 1$

$\rightarrow k \leq 1 + \log_2((n-8)/2)/\log_2(5/4)$

It follows that we can take $t=1+\lceil \log_2((n-8)/2)/\log_2(5/4)\rceil$ in (13) and hence the theorem. ∎

We now describe a sequence of graphs for which the greedy algorithm could behave badly. We have not been able to find graphs giving the bound in theorem 1. We have only been able to make the ratio GREEDY/OPTIMAL grow as $\log_2 n/\log_2\log_2 n$. Further research may close this gap.

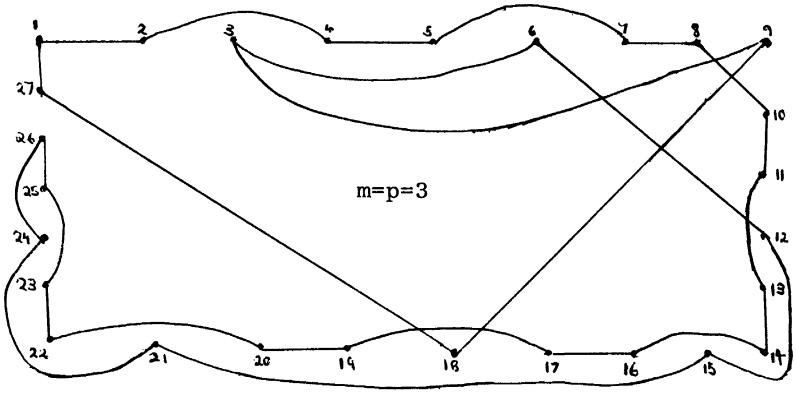
Notation

For $m \geq 1$ and $p \geq 3$ let $G=G(m,p)$ be the complete graph with $n=p^m$ nodes. Let N denote the set of nodes of G and E denote its edges.

Let $B_k=\{j \in N: j=ap^k$ where $p \nmid a\}$ $\quad k=0,1,\ldots m$

Assume that elements in B_k have been ordered so that for k even B_k is ordered by ascending numerical value and for k odd B_k is ordered by descending numerical value.

Next let $H_2=(i_1,\ldots,i_n,i_1)=(B_0,\ldots B_m,i_1=1)$. We will define edge lengths for G so that H_2 is a possible greedy tour with $l(H_2)>n\log_2 n/3\log_2\log_2 n$ and $H_1=(1,2,\ldots n,1)$ is a minimum length tour of length n.



Next define $E_k=\{(i_t,i_{t+1}):i_t \in B_k\}$ for $0\leq k\leq m$ and $F_k=\{(i,j)\in E: i=ap^k$ and $j=i+p^k\}$ for $0<k<m$. F_0 is defined to be H_1.

We define next $d:E\to R$ the set of real numbers by

$d(i,j)=(p-1)^k$ if $(i,j)\in E_k\cup F_k$ $\quad 0\leq k<m$

$= \infty$ otherwise

For a path P in G we define $s(P)=\min(q:P\cap(E_q\cup F_q)\neq\emptyset)$. We now prove

Lemma 2

Let $i\in B_{k_1}, j\in B_{k_2}$ with $k_1\leq k_2$. If P is a shortest path from i to j when d defines edge lengths then $s(P)=k_1$.

Proof

By considering the edge containing i we see that $r=s(P)\leq k_1$. If $r<k_1$ let P_1 be a sub-path of P whose edges belong to $E_r\cup F_r$ and which is not contained in any other such sub-path. Let $x_t=a_t p^{j_t}\in B_{j_t}$ for t=1,2 be the endpoints of P_1 and assume $j_1\leq j_2$ and note that $r<j_1$. It follows that

$a_2p^{j_2}=a_1p^{j_1}+bp^r$ for some integer b and further that $b=cp^{j_1-r}$ for some integer c.

We next deduce that $l(P_1)>|b(1-1/p)(p-1)^r|$. This is because it needs at least p-1 edges in $E_r \cup F_r$ to go from a node x to $x+p^{r+1}$ in G and we can use p-1 edges only if $x \in B_r$. Thus

$$l(P_1)>|cp^{j_1-r-1}(p-1)^{r+1}|$$
$$\geq |c(p-1)^{j_1}|$$

But this means we can shorten P by replacing P_1 by |c| edges in F_{j_1}.

The edge lengths l(i,j) for our travelling salesman problem are defined by l(i,j)=the length of a shortest path from i to j when edge lengths are defined by d.

An immediate corollary of lemma 2 is that if $(i,j) \in E_k \cup F_k$ for some k then l(i,j)=d(i,j).

With this definition of l we have $l(H_1)=n$ and

$$l(H_2)=1+\sum_{k=0}^{m-1}(p^{m-k}-p^{m-k-1})(p-1)^k$$
$$=1+p^m\sum_{k=0}^{m-1}(1-1/p)^{k+1}$$
$$=1+p^{m+1}(1-(1-1/p)^m)(1-1/p)$$

We show next that the edges of H_2 can be chosen by the greedy algorithm in the order $E_m, E_0, \ldots E_{m-1}$.

The greedy algorithm can certainly select edges in E_m, E_0 as they are of length 1. Suppose then that the edges in $E_m, E_0, \ldots E_{k-1}$ have been selected for $k>1$. One then notes that if $(i,j) \notin E_k$ and (i,j) can be added to those edges so far selected without creating subtours or nodes of degree 3 then $i \in B_{k_1}, j \in B_{k_2}$ where $k \leq k_1, k_2$. Lemma 2 implies that $l(i,j) \geq (p-1)^k$ and so the greedy algorithm could select all the edges in E_k next.

We thus have

Theorem 2

There exists a sequence of graphs G_m with $n=m^m$ nodes such that for $m \geq 3$ the ratio GREEDY/OPTIMAL can be worse than

$\log_2 n/3\log_2\log_2 n$.

Proof

Let $G_m=G(m,m)$. We have $l(H_1)=n$ and

(15) $l(H_2) > nm(1-(1-1/m)^m)(1-1/m)$

Now $(1-1/m)^m<1/2$ for $m>2$ and so for $m\geq 3$ we deduce from (15) that $l(H_2)>nm/3$

Now $\log_2 n=m\log_2 m$ and $\log_2\log_2 n=\log_2 m+\log_2\log_2 m>\log_2 m$.

Thus $\log_2 n/\log_2\log_2 n<m$ and hence

$l(H_2)>(\log_2 n/3\log_2\log_2 n)l(H_1)$ ■

One might expect some improvement if we consider k-greedy algorithms -see Frieze [7] and Hausmann and Korte [10] -where one chooses the k best edges to add at each stage. However we see that for large enough m, H_2 in G_m of theorem 2 can be selected by a k-greedy algorithm and so no improvement can be expected in the worst case.

SAVINGS ALGORITHM

The savings algorithm (for the vehicle scheduling problem) was described by Clarke and Wright [2]. We will use an idea of Golden [8] to produce a sequence of graphs for which the savings algorithm can produce a tour of length SAVINGS which satisfies

(16) SAVINGS/OPTIMAL $> \log_2 n/3\log_2\log_2 n$

for large enough n where n is the number of nodes.

In the savings algorithm we pick one node, node 0 and for each edge $(i,j)\neq 0$ we define the savings s(i,j) by

$$s(i,j) = l(0,i) + l(0,j) - l(i,j)$$

The savings algorithm chooses edges not containing 0 in decreasing order of savings until a path is produced which goes exactly once through each node other than node 0. (The version where the edges chosen have always to form a path was analysed by Golden [8]). The tour is completed by joining node 0 to each endpoint.

From G_m of Theorem 2 we produce $\hat{G}_m$ by adding an extra node 0 and defining $l(0,j)=(m-1)^{m-1}$ for each node j of G_m. One can see that for any edge (i,j) of G_m we have $s(i,j)=2(m-1)^{m-1} - l(i,j)$ and that the triangular inequality still holds.

One can see that in this graph

$$OPTIMAL = 2(m-1)^{m-1} + m^m - 1$$

and that the savings algorithm could produce the tour $(0,\hat{H}_2,0)$ where $\hat{H}_2$ is H_2 minus the edge $((m-1)^{m-1},m^m)$. In this case

$$SAVINGS = (m-1)^{m-1} + l(H_2)$$

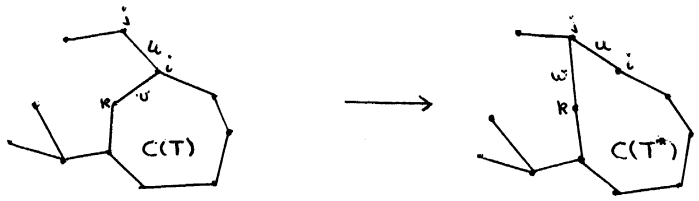
Inequality (16) then follows as in Theorem 2.

TREE ALTERATION ALGORITHM

The concept of a 1-tree was introduced by Held and Karp [9] and assuming the nodes of G are {1,2,...n} it consists of a spanning tree on the nodes 2,...n plus two edges containing node 1.

We now describe a simple method for altering a minimum length 1-tree which results in a tour of length ALTER where ALTER$\leq$2*OPTIMAL.

So let G be a given complete graph on n nodes with non-negative edge lengths satisfying (1). A 1-tree T can be viewed as a spanning tree of G plus one extra edge. Define a treecycle to be a spanning tree of G plus one extra edge. Thus a treecycle T contains a unique cycle C(T). Next let u=(i,j) be an edge such that i is in C(T) and j is not. Let v=(i,k) where k is a neighbour of i in C(T) and let w=(j,k). Note that w$\notin$T and T*=T+w-v is a treecycle and that u$\in$C(T*).



Note also that C(T*) contains one more node than C(T) and that l(T*)=l(T)+l(w)-l(v)

$\leq$l(T)+l(u) from (1)

Consider then the following tree alteration algorithm:

(a) Find a minimum length 1-tree T_1.

p:=1.

(b) If T_p is a travelling salesman tour stop, else choose $u_p=(i_p,j_p)\ \varepsilon\ T_p$ where i_p is in $C(T_p)$ and j_p is not. Let k_p be a neighbour of i_p in T_p and let $v_p=(i_p,k_p)$ and $w_p=(j_p,k_p)$.

(c) $T_{p+1}=T_p+w_p-v_p$

p:=p+1

go to (b).

Theorem 3

If ALTER=$l(T_p)$ where T_p is the tour produced by the algorithm then ALTER$\leq$2×OPTIMAL.

Proof

From the discussion preceding the algorithm we have

$$l(T_p)\leq l(T_1)+\sum_{q=1}^{p} l(u_q)$$

Now it is straightforward to show that $u_1,\ldots u_p$ are distinct edges of $T_1-C(T_1)$.

It follows therefore that

(17) ALTER$\leq 2\times l(T_1) - l(C(T_1))$

$\leq$2×OPTIMAL

Now finding a minimum spanning 1-tree in a complete graph has complexity $O(n^2)$ if Dijkstra's method [5] is used to find the tree on 2,...n. The procedure for altering the treecycle to produce a tour can also be done in $O(n^2)$ time and so the complexity of the algorithm is $O(n^2)$.

We now describe a graph with n nodes and edge lengths such that it is possible to have ALTER=(2-3/n) × OPTIMAL.

Let N={1,2,...n} and define edge sets

$S_1=\{(i,n):1\leq i<n\}$

$S_2=\{(i,i+2):2\leq i\leq n-3\}$

$S_3=\{(1,2),(3,n-a)\}$ where a=1 if n is odd

=2 if n is even

Then define l by

$l(i,j)=1$ if $(i,j)\epsilon S_1+S_2+S_3$

$=2$ otherwise

Theorem 4

Let G=(N,E) be a complete graph on n nodes and let edge lengths l be as defined above. Then using the tree alteration algorithm we could obtain ALTER=(2-3/n)OPTIMAL.

Proof

The tour H_1 where

$H_1=(n,1,2,4,6,\ldots n-a,3,5,7,\ldots n+a-3,n)$

is of length n and so is optimal.

Next let $T_1=S_1+(1,2)$. Now T_1 is a 1-tree and $l(T_1)=n$ and so it is a minimum length 1-tree.

We next note that $l(i,i+1)=2$ for $2\leq i\leq n-2$ and so if $H_2=(1,2,\ldots n,1)$ then $l(H_2)=2n-3$. It remains to show that H_2 could be chosen by the algorithm.

If $T_k=\{(i,i+1):1\leq i\leq k\}+\{(n,i):i=1 \text{ or } k+1\leq i\leq n-1\}$ for $1\leq k\leq n-2$ then T_k is a treecycle with $C(T_k)=(1,\ldots k+1,n,1)$ and $T_k=T_{k-1}+(k,k+1)-(k,n)$. Consider $(k+1,n)\epsilon T_{k-1}$. Now node n is in $C(T_{k-1})$ and node k+1 is not. Node k is adjacent to n in $C(T_{k-1})$. Thus the algorithm could produce $T_1,T_2,\ldots T_{n-2}$ in this sequence. Now $T_{n-2}=H_2$ and the result follows.

■

A MODIFICATION OF THE TREE ALTERATION ALGORITHM

We describe in this section a modification of the tree alteration algorithm which for each k $1 \leq k \leq n-2$ produces a tour T_k of length $ALTER_k$ which satisfies

(18) $ALTER_k \leq (2 - k/n)$ OPTIMAL

The time complexity of the algorithm is $0(n^{3+k})$. For fixed k define

$F_k = \{S \in F : p(S)=1$ and $|S|=k\}$

The notation used is that for lemma 1. Thus $S \in F_k$ if the edges in S form an open path with k edges. We will also refer to path S.

For $S \in F_k$ we let I(S)=the set of internal nodes of the path S.

Next let $T^* = (u_1, \ldots u_n)$ be an optimal tour where the edges $u_1, \ldots u_n$ are in 'tour order'. For $1 \leq t \leq n$ define

$S_t = \{u_t, u_{t\oplus 1}, \ldots u_{t\oplus(k-1)}\}$ where

$t \oplus j = t+j$ if $t+j \leq k$

$= t+j-k$ otherwise

Thus $S_t \in F_k$ for $1 \leq t \leq n$.

The proposed algorithm is as follows:

For each $S \in F_k$ do (a),(b) below:

(a) Find a minimum length spanning tree TREE(S) in the graph G(S) which is the complete graph with node set N-I(S) minus the edge e(S)=(i(S),j(S)) where i(S),j(S) are the endpoints of S.

(b) The graph TREE(S)+S is a treecycle TC(S). Using the technique described in the previous section transform TC(S) into a tour T(S).

Next define T_k by

$l(T_k) = \min(l(T(S)) : S \in F_k)$.

The time complexity of the algorithm is $\leq 0(n^{3+k})$ because $|F_k| = (n!/(n-k-1)!)/2$ for $k<n-1$.

Theorem 5

If $ALTER_k = l(T_k)$ then (18) holds.

Proof

We note first that

$ALTER_k \leq \min(l(T(S_j)):1\leq j\leq n)$

Now $l(TC(S_j)) = l(TREE(S_j) + l(S_j)$

$\leq$ OPTIMAL

as T^*-S_j is a spanning tree of $G(S_j)$.

Thus from (17) we have

$$l(T(S_j)) \leq 2l(TC(S_j)) - l(C(TC(S_j)))$$
$$\leq 2 \text{ OPTIMAL} - l(S_j)$$

as $S_j \subseteq C(TC(S_j))$.

Now as $l(S_1)+\ldots+l(S_n)=k$ x OPTIMAL there exists t such that

$l(S_t) \geq (k/n)$OPTIMAL. Thus

$$ALTER_k \leq l(T(S_t))$$
$$\leq 2 \text{ OPTIMAL} - (k/n)\text{OPTIMAL}$$

■

MAXIMUM LENGTH TOURS

We finally consider the problem of finding a maximum length tour. We assume $l(e)\geq 0$ as before but the triangular inequality (1) is no longer needed.

Let T^* be a maximum length tour and for $1\leq k\leq n-2$ let F_k and $S_1,\ldots S_n$ be as defined in the previous section.

We consider first the symmetric case:

Algorithm k

For each $S\varepsilon F_k$ do (a),(b),(c):

(a) Find a maximum length degree constrained subgraph D(S) of G(S). The degree constraints are: degree i(S),j(S)=1, and the degree of any other node=2.

(b) D(S) consists of a set of node disjoint cycles plus a path from i(S) to j(S).

Create the set of edges $D_1(S)$ by removing the shortest edge from each cycle of D(S).

(c) Let f be the shortest edge of S.Let $D_2(S)=D_1(S)+(S-f)$.

Now $D_2(S)\varepsilon F$ and can be extended to a tour T(S) in any reasonable manner

Let $DEGREE_k = l(T_k) = \max(l(T(S)) : S \in F_k)$
This algorithm is a modification of an algorithm described in Fisher, Nemhauser and Wolsey [13].
Note that D(S) can be found in $O(n^3)$ time using the matching algorithm described in Lawler [14].

Theorem 6

(19) $DEGREE_k \geq (2/3 + k/3n - 1/n)OPTIMAL$

Proof

$DEGREE_k \geq \max(l(T(S_j)) : j=1,\ldots n)$.

Fix j and note that

(20) $l(T(S_j)) \geq l(D_1(S_j) + (1-1/k)l(S_j)$

Each cycle in $D(S_j)$ has at least 3 edges and so

(21 $l(D_1(S_j)) \geq (2/3)l(D(S_j))$

We also have

(22) $l(D(S_j)) \geq OPTIMAL - l(S_j)$

as T^*-S_j satisfies the degree constraints in (a).

From (20),(21),(22) we get

$l(T(S_j)) \geq (2/3)OPTIMAL + (1/3 - 1/k)l(S_j)$

Choosing j so that $l(S_j) \geq (k/n)OPTIMAL$ gives (19). ■

The unsymmetric case is similar except that G is now a digraph. The matching problem in (1) is replaced by an assignment problem with variables x(i,j) for $i,j \in N-I(S)$. Assuming S is directed from i(S) to j(S) we impose $x(i(S),j(S))=1$. Cycles can now have 2 edges and so if $ASSIGN_k$ is the length of the tour produced

$ASSIGN_k \geq (1/2 + k/2n - 1/n)OPTIMAL$

REFERENCES

[1] N.Christofides,"Worst-case analysis of a new heuristic for the travelling salesman problem",Management Science Research Report No.388 Carnegie-Mellon University (1976).

[2] G.Clarke and J.Wright,"Scheduling of vehicles from a central depot to a number of delivery points",Operations Research 12(1964) 568-581.

[3] S.A. Cook,"The complexity of theorem proving procedures", Proceedings of the Third ACM Symposium on the Theory of Computing(1971) 151-158.

[4] G.Cornuéjols and G.L. Nemhauser,"Tight bounds for Christofides' travelling salesman heuristic", Mathematical Programming 14(1978) 116-121.

[5] E.W.Dijkstra,"A note on two problems in connection with graphs",Numerische Mathematik 1(1959) 269-271.

[6] J.Edmonds,"Matroids and the greedy algorithm", Mathematical Programming 1(1971) 127-136.

[7] A.M. Frieze,"Algorithms for generalisations of matroids", submitted to Mathematical Programming.

[8] B.L. Golden,"Evaluating a sequential vehicle routing algorithm", AIIE Transactions 9(1977) 204-208.

[9] M.Held and R.M.Karp,"The travelling salesman problem and minimum spanning trees: part II",Mathematical Programming 1(1971) 6-25.

[10] D.Hausmann and B.Korte,"K-greedy algorithms for independence systems",Report No. 7764-OR,Institut Fur Okonometrie and Operations Research, Universitat Bonn (1977).

[11] T.A. Jenkyns,"The efficacy of the greedy algorithm", Proceedings 7^{th} S-E Conference Combinatorics,Graph Theory and Computing (1976)341-350.

[12] R.M.Karp,"Reducibility among combinatorial problems", in Complexity of Computer Computations,R.E.Miller and J.W.Thatcher(Eds.)Plenum Press,New York(1972)85-104.

|13| M.L. Fisher, G.L. Nemhauser and L.A. Wolsey, "An analysis of approximations for finding a maximum weight hamiltonian circuit".
|14| E.L. Lawler, "Combinatorial optimisation:networks and matroids", Holt,Rinehart and Winston (1976)
|15| D.J. Rosenkrantz, R.E. Stearn and P.M. Lewis, "Approximate algorithms for the travelling salesman problem", Proceedings of 15th IEEE Symposium on Switching and Automata Theory (1974)33-42.

Address:
A.M. Frieze
Queen Mary College
London University
Mile End Road
London E1
England