Reprinted from NETWORKS, Volume 12, Number 1, Spring 1982

On the Worst-Case Performance of Some Algorithms for the Asymmetric Traveling Salesman Problem

A. M. Frieze
Department of Computer Science and Statistics, Queen Mary College, London University, London, England

G. Galbiati
Istituto di Matematica, Universita di Pavia, 27100 Pavia, Italy

F. Maffioli
Istituto di Elettrotecnica ed Elettronica & Centro Telecommunicazioni Spazioli of CNR, Politecnico di Milano, 20133, Milano, Italy

We consider the asymmetric traveling salesman problem for which the triangular inequality is satisfied. For various heuristics we construct examples to show that the worst-case ratio of length of tour found to minimum length tour is $\Omega(n)$ for n city problems. We also provide a new $O(\lceil \log_2 n \rceil)$ heuristic.

I. INTRODUCTION

The traveling salesman problem has the following simple description: given a complete digraph on n nodes with an $n \times n$ matrix $||d(i,j)|| \geqslant 0$ giving the lengths of arcs (i, j) find a minimum length circuit (or *tour*) which goes through each node exactly once. The length $d(T)$ of a tour T is given by $d(T) = \Sigma_{(i,j)\in T} d(i,j)$.

The problem has been studied extensively for the past few decades and many algorithms have been proposed for its exact solution.

None however have worst-case time bounds which are polynomial in n. A complexity theory (NP-completeness) initiated by Cook [3] and Karp [16] and extensively covered in Garey and Johnson [11] indicates that an exact algorithm for this problem with a polynomial time bound seems unlikely to exist.

One is therefore also interested in approximate algorithms which take time polynomial in n but which do not guarantee an optimal solution but seem likely to lead to 'good' solutions.

Given such an heuristic method H it is of interest to study its worst-case ratio $R_n(H)$ for n node problems where

$$R_n(H) = \text{Sup}\,(d(T)/d(T^*) : T^* \text{ is a minimum length tour, } T \text{ is a tour generated by } H \text{ under the assumption } d(T^*) > 0)$$

NETWORKS, Vol. 12 (1982) 23-39

CCC 0028-3045/82/010023-17$01.70

One would like to have a polynomial heuristic for which $R_n = 1 + \epsilon$ where ϵ is small and positive. Unfortunately, the problem of finding a tour within any required accuracy is also NP-hard–see Sahni and Gonzales [26].

If however we restrict our attention to problems in which the triangular inequality

$$d(i,j) \leqslant d(i,k) + d(k,j) \tag{1}$$

for all $i, j, k \in N$ holds (or maximum length tours) then the outlook is brighter.

Most attention has been focused on the symmetric case where $d(i,j) = d(j,i)$ holds for all $i, j \in N$. For this problem the best known result is that for Christofides heuristic [1] for which it is known that $R_n = (3m - 1)/2m$, where $m = [n/2]$–see Cornuéjols and Nemhauser [4].

In this article we consider the asymmetric problem in which (1) holds.

Sections II-V deal with variations on a number of heuristics proposed over the last 15 years or so. We find that though quite good empirically these all have worst-case performance $R_n = \Omega(n)$.* This is rather poor considering that (1.1) implies that no tour can be more than n times the length of an optimum tour.

Section VI describes a heuristic based on repeated assignment which in contrast to the above has $R_n \leqslant [\log_2 n]$.

Finally Sec. VII gives some 'data dependent' bounds for which R_n depends on extra assumptions about the $d(i,j)$.

Notation

This notation is used for Sec. II-V where we only exhibit lower bounds for R_n.

For each algorithm dealt with we first define a digraph $\hat{G} = (N, A)$, where $N = \{1, 2, \ldots, n\}$ and $A \subseteq N^2$ together with a function $l: A \to \mathbb{R}$. The arc set A is implicitly defined as the set of (i,j) for which a value $l(i,j)$ is given.

The actual "hard" example satisfying (1) uses the complete digraph on n nodes with arc lengths defined by d where $d(i,j)$ = minimum length of a path from i to j in $\hat{G}$ using arc lengths defined by l. We will always have $d(i,j) = l(i,j)$ for $(i,j) \in A$.

The optimal tour will always be $T^* = (1, 2, \ldots, n, 1)$. The tour found by the particular heuristic under discussion will always be denoted by T. Tours will sometimes be viewed as sets of arcs to help with notation. For a set $S \subseteq N^2$ $d(S) = \Sigma_{(i,j)\in S} d(i,j)$

To avoid listing special cases for $i + 1, i - 1$ we adopt the convention:

$$\text{if } i \leqslant 0 \quad \text{node } i = \text{node } i + n,$$

$$\text{if } i > n \quad \text{node } i = \text{node } i - n.$$

II. GREEDY ALGORITHMS

This important algorithm has been widely studied in relation to matroids and general independence systems–see Edmonds [5], Jenkyns [14], and Korte and Hausmann [18].

*We say $f(n) = \Omega(g(n))$ if there exists $c > 0$ such that $f(n) \geqslant cg(n)$ for all n.
$= O(g(n))$ if there exists $c > 0$ such that $f(n) \leqslant cg(n)$ for all n.

Greedy Algorithm

begin
$T := \phi$;
while T is not a tour *do*
begin
$d(e) = \min (d(f): f \in N^2$ and $T \cup \{f\}$ is contained in at least one tour)
$T := T \cup \{e\}$
end
end

Definition of $\hat{G}$

We assume n is even and $n = 2m$

$$l(i, i+1) = 1, \qquad i \in N - \{m\},$$
$$l(m, m+1) = L, \qquad \text{where } L \text{ is "large"},$$
$$l(i,j) = 1, \qquad m+1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant n \quad i \neq j,$$
$$l(i,j) = 1, \qquad 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m \quad i \neq j.$$

We note the following:

$$d(T^*) = n + L - 1$$
$$\text{If } 1 \leqslant i \leqslant m \text{ and } m+1 \leqslant j \leqslant n \text{ then } d(i,j) \geqslant L \tag{2}$$

Now the greedy algorithm could select $(n - k, k + 1)$ for $0 \leqslant k \leqslant m - 1$ as its first m arcs. It would then be forced to select the arcs satisfying (2). Under these circumstances $d(T) \geqslant n(L+1)/2$. Thus

$$R_n \geqslant n(L+1)/2(n+L-1). \tag{3}$$

Since L can be made arbitrarily large the RHS of (3) can be made arbitrarily close to $n/2$. For the symmetric case we showed in Frieze [8] that

$$\Omega(\log n/\log\log n) \leqslant R_n \leqslant O(\log n).$$

Related Heuristics

k-greedy methods

One might expect some improvement if instead of adding one arc at a time, one added the minimum length set of k arcs at each stage (k fixed), as in Frieze [9] or Hausmann, Jenkyns and Korte [13]. By considering the same example as above we see that for $m > k$ the first $k\lfloor m/k \rfloor$ arcs (i, j) could satisfy $m + 1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Consequently T must then contain $k\lfloor m/k \rfloor$ arcs satisfying (2) and so again R_n is $O(n)$.

Savings method

In the savings method—see Clarke and Wright [2] for details—one chooses a particular node, node 0 and then chooses arcs in descending order of $s(i, j) = d(i, 0) + d(0, j) - d(i, j)$. Append an extra node 0 to the example above with $d(0, i) = d(i, 0) = L$. The savings algorithm acts exactly like the greedy algorithm for the first $n - 1$ arc choices and again R_n is $O(n)$—see also Golden [12]

Loss methods

Webb [28] and Van Der Cruyssen and Rijckaert [27] propose selecting arcs on the basis of what it costs not to select them. Details are contained in the referenced papers but as an example suppose T consists of those arcs selected so far and for some node i_0 T contains an arc (i_0, j) but no arc (j, i_0). Let (j_1, i_0), (j_2, i_0) be the two cheapest available arcs entering i_0 that can be added to T. Then if (j_1, i_0) is not used a "loss" of at least $L(i_0) = d(j_2, i_0) - d(j_1, i_0) \geqslant 0$ is incurred associated with node i_0. This idea is generalized to all nodes that are not incident to 2 arcs of T. The arc chosen is that associated with the greatest loss.

In the example considered above initially all nodes have losses of zero and this remains true if the algorithm selects $(n - k, k + 1)$ for $0 \leqslant k \leqslant m - 4$ as its first $m - 3$ arcs. Consequently the algorithm would then have to choose $m - 3$ arcs satisfying (2) and so again R_n is $O(n)$.

III. NEAREST NEIGHBOR ALGORITHM

This has the flavor of greedy, except that the set of arcs selected at any stage for a simple path. It was first proposed in Karg and Thompson [15].

For a simple path P let $i(P)$, $j(P)$ denote the initial and terminal nodes of P. We allow a singleton node to be a path with no edges.

Nearest Neighbor Algorithm

begin
min := ∞,
for $m = 1, \ldots n$ *do* (m is the initial node from which the next tour is to be generated),
begin $T_m := \phi$; $i(T_m) := j(T_m) := m$;
  while $|T_m| < n - 1$ *do*
  begin $d(e) = \min(d(x, y)$: x is not in T_m and $y = i(T_m)$
    or y is not in T_m and $x = j(T_m))$
    $T_m := T_m \cup \{e\}$,
  end
$T_m := T_m \cup \{(j(T_m), i(T_m))\}$;
  if $d(T_m) < \min$ *then* $T := T_m$; min := $d(T_m)$
  end
end

The following example serves *a fortiori* whenever a single starting node is used.

Definition of $\hat{G}$

$$
\begin{aligned}
&l(i, i+1) = 1, && i \in N - \{2\},\\
&l(2, 1) = 1,\\
&l(2, 3) = n - 1,\\
&l(i, 2) = 1, && i \in N - \{2\},\\
&l(3, i) = 1, && i \in N - \{3\},\\
&l(i, i-1) = n, && i \in N - \{3\},\\
&l(i, i-2) = n, && i \in N - \{3, 4\}.
\end{aligned}
$$

We note the following:

$$d(T^*) = 2n - 2.$$

For $m \notin \{1, 2, 3\}$ the algorithm could select $(3, m)$, $(m, 2)$, $(2, 1)$ as its first 3 arcs. It could then proceed to produce

$$T_m = (3, m, 2, 1, n, n-1, \ldots, m+1, m-1, \ldots, 4, 3)$$

and

$$d(T_m) = n(n-3) + 3.$$

For $m \in \{1, 2, 3\}$ we could have

$$T_m = (3, 4, 2, 1, n-1, n-2, \ldots, 5, 3)$$

and again $d(T_m) = n(n-3) + 3$.
Thus

$$
\begin{aligned}
R_n &\geqslant (n(n-3) + 3)/(2n - 2)\\
&\approx n/2, \qquad \text{for large } n.
\end{aligned}
$$

In the symmetric case Rosencrantz, Stearns, and Lewis [25] showed that $R_n = O(\log n)$.

A k-greedy version

It is plausible to consider expanding the current path by k nodes at a time as cheaply as possible. A bad example is obtained by adding the following arcs to $\hat{G}$ as defined above: We assume $k \geqslant 5$ initially

$$l(i, i+2) = 1, \qquad i \in N - \{2\},$$

$$l(i, i-1) = 1, \qquad i = 1, n, n-1, \ldots, n-k+5,$$
$$l(n-k+4, n-2k+4) = n,$$
$$l(n-pk+3, n-(p+2)k+3) = n, \qquad p = 1, \ldots, \lfloor n/k \rfloor - 2.$$

Using these lengths the k-greedy version starting at node $m \notin \{1, 2, 3\}$ could choose arcs

$$(3, m), (m, 2), (2, 1), (1, n), \ldots, (n-k+5, n-k+4)$$

and then

$$(n-k+4, n-2k+4), (n-2k+4, n-2k+5), \ldots, (n-k+2, n-k+3)$$

and then

$$(n-k+3, n-3k+3), (n-3k+3, n-3k+4), \ldots, (n-2k+2, n-2k+3)$$

and so on. The arcs $(i, i+2)$ of length 1 are used to "skip over" node m. For nodes $m \in \{1, 2, 3\}$ the tour produced could be that for node 4.

The length of the tours T_m generated can be seen to satisfy

$$d(T_m) \geqslant k + (n+k-1)\lfloor (n-k)/k \rfloor$$

and so for k fixed $R_n = \Omega(n)$.

If $2 \leqslant k \leqslant 4$ we proceed in a similar manner to above using straightforward modifications, to get the tours going the "wrong way round."

IV. CHEAPEST INSERTION ALGORITHM

In this algorithm we start with a degenerate subtour consisting of a single node. Then at a general stage we have a subtour T through the nodes $C \subseteq N$.

We generalize the idea of building a tour by inserting a node $k \notin C$ between consecutive nodes (i, j) of T as cheaply as possible as proposed by Nicholson [22].

Where

$e = (i, j) \in T$ and $S \subseteq N - C$ let $\Delta(e, S) =$ (length of shortest path that starts at i, visits each node of S exactly once and terminates at j) $- d(i, j)$

= minimum cost of "inserting" S between i and j.

We further define for $S \subseteq N - C$.

$\Delta(S) = \min (\Sigma_{e \in T} \Delta(e, S_e) := \{S_e\}$ is a partition of S into $|T|$ subsets)

= minimum cost of inserting S into the tour.

Now let k be a fixed positive integer

k-Insertion Algorithm

begin
min := ∞
for $m = 1, \ldots, n$ *do*
begin
$T_m := \phi$; $C_m := \{m\}$
while $C_m \neq N$ *do*
begin

Step A
- Let $k_1 = \min(k, |N - C_m|)$;
- Let $\Delta(S^*) = \min(\Delta(S): S \subseteq N - C_m$ and $|S| = k_1)$;
- $C_m := C_m \cup S^*$;
- amend T_m as implied by calculation of $\Delta(S^*)$

end
if min $> d(T_m)$ *then* min := $d(T_m)$; $T := T_m$
end
end

We note that the algorithm runs in $O(n^{2k+2})$ time.

Definition of $\hat{G}$

$$l(i, i+1) = 1, \qquad \text{for } i \in N,$$
$$l(i, i-1) = h = (n-k-1)/(k+1) \qquad \text{for } i \in N.$$

We note that

$$d(T^*) = n.$$

We will show that for $n \geqslant 3k+1$ we could have

$$d(T) > k\lfloor (n-k-1)/k \rfloor (h+1),$$
$$\approx n^2/k.$$

This shows that for fixed k, R_n is $\Omega(n)$ after showing that in producing T_1 we could have $T_1 = (1, m, m-1, \ldots, 3, 2, 1)$ after p executions of Step A where $m = pk + 1 \leqslant n - k$. The result for $T_2, \ldots, T_n$ follows by symmetry.

We first show that $H_0 = (1, k+1, k, \ldots, 3, 2, 1)$ is a minimum length tour through $[k+1]$ ($[m] = \{1, 2, \ldots, m\}$).

To see this we note that for arc (i, j)

$$d(i,j) = \min(j - i, (n - (j - i))h), \qquad \text{if } j > i,$$
$$= \min((i - j)h, n - (i - j)), \qquad \text{if } i > j.$$

Now for $n \geq 2k$ we have $d(H_0) = k(h+1)$. Next let $H = (i_0 = 1, i_1, \ldots, i_k, i_{k+1} = 1)$ be any tour through $[k+1]$. Let $m_t = i_t - i_{t-1}$ for $1 \leq t \leq k+1$ and let $I^+ = \{t: m_t > 0\}$ and $I^- = \{t: m_t < 0\}$. We note that there exists p such that $i_p \geq k+1$. It follows then that

$$\sum_{t \in I^+} m_t \geq k \text{ and hence } \sum_{t \in I^-} (-m_t) \geq k,$$

as

$$\sum_{t=1}^{k+1} m_t = 0.$$

It now follows that

$$\begin{aligned} d(H) &= \sum_{t \in I^+} \min(m_t, (n - m_t)h) + \sum_{t \in I^-} \min(-m_t h, n + m_t), \\ &\geq \min(k, (n-k)h) + \min(kh, n-k), \\ &= k + kh \\ &= d(H_0). \end{aligned}$$

The inequality follows from the fact that the minimum of a concave function over a convex polyhedron occurs at a vertex.

Assume now inductively that after p iterations of Step A we have $T_1 = (1, m, m-1, \ldots, 3, 2, 1)$ where $m = pk + 1 \leq n - 2k$.

If we insert $m+1, \ldots, m+k$ between nodes $1, m$ so that T_1 becomes $(1, m+k, \ldots, m+1, m, \ldots, 1)$ the increase in length is $k(h+1)$. Now let $S \subseteq N - [m]$, where $|S| = q \leq k$. We consider inserting S into T_1.

Case 1.

$$\Delta((r, r-1), S) \geq q(h+1), \quad r \in \{2, \ldots, m\}. \tag{4}$$

Suppose $(r = i_0, i_1, \ldots, i_q, i_{q+1} = r-1)$ is a path from r to $r-1$ through S and let $m_t = i_t - i_{t-1}$ for $1 \leq t \leq q+1$. As there exists $i_p \geq r+q$ we have

$$\sum_{t \in I^+} m_t \geq q \text{ and hence } \sum_{t \in I^-} (-m_t) \geq q+1$$

and hence

$$h + \Delta((r, r-1), S) \geq \min(q, (n-q)h) + \min((q+1)h, n-q-1) = q + (q+1)h$$

which implies (4).

Case 2.

$$\Delta((1, m), S) \geq q(h + 1) \tag{5}$$

Define m_t as before. We see that there exists $i_p \geq m + q$ and so

$$\sum_{t \in I^+} m_t \geq m + q - 1 \text{ and } \sum_{t \in I^-} (-m_t) \geq q$$

and so

$$\begin{aligned} m - 1 + \Delta &\geq \min (m + q - 1, (n - (m + q - 1))\, h) + \min (qh, n - q) \\ &= m + q - 1 + qh \end{aligned}$$

which implies (5).

It follows that $\Delta(S) \geq k(h + 1)$ for $|S| = k$ and the inductive step is complete. For the symmetric case Rosencrantz, Stearns, and Lewis [25] showed that for $k = 1$, $R_n = 2 - 1/n$.

V. INTERCHANGE ALGORITHMS

The most successful heuristics to date are those based on trying to improve a given tour by deleting some arcs and replacing them by others. This has been proposed by several authors, e.g., Lin [19], Lin and Kernighan [20], Nicholson [22], and Reiter and Sherman [24].

Given a tour T a proper k-swap T' is any tour such that $|T \cap T'| \geq n - k$. k being some positive integer.

For the symmetric case it is quite satisfactory to define *k-optimality* by

$$T \text{ is } k\text{-optimal if } d(T) = \min (d(T'): T' \text{ is a proper } k\text{-swap of } T).$$

For the nonsymmetric case there are problems as exemplified by the fact that T is its own unique 2-swap and so 2-optimality is trivial.

A more sensible definition of a k-swap must allow some subpaths of T to be traversed in the opposite direction to that of T.

We thus define for tour T rev $(T) = \{(j, i): (i, j) \in T\}$ and a k-swap T' to be a tour such that $|T' \cap (T \cup \text{rev } (T))| \geq n - k$, and $\text{NHD}(T) = \{T' : T' \text{ is a } k\text{-swap of } T\}$. A tour T is k-optimal if $d(T) = \min (d(T') : T' \in \text{NHD}(T))$.

To find a k-optimal tour we start with an arbitrary tour T and search NHD(T) for a better tour. If a better is found we repeat the process with this tour otherwise we have a k-optimal tour.

As the verification that a tour is indeed k-optimal requires at the very least $O(n^k)$ time a *necessary* condition for this process to be polynomial is that k be fixed independent of n.

We show that under these circumstances R_n is again $O(n)$.

Definition of $\hat{G}$

We assume n is odd, $n = 2m + 1$ and $n \geqslant 2k^2 + 1$

$$\begin{aligned} l(i, i+1) &= 1, && \text{for } i \in N, \\ l(i, i+m) &= h = (n-k)/2k, && \text{for } i \in N, \\ l(i, i-m) &= kh, && \text{for } i \in N, \end{aligned}$$

Note that node $i + m$ is node $i + m \bmod n$ by our convention. With this definition of l the shortest distances d satisfy

$$\begin{aligned} d(i,j) &= \min(j-i, (n-j+i)h), && 1 \leqslant j-i \leqslant m, \\ &= \min(j-i, 2(n-j+i)h, (2j-2i-n)kh), && m+1 \leqslant j-i < n, \\ &= \min(n-i+j, 2(i-j)h, (n-2i+2j)kh), && 1 \leqslant i-j \leqslant m, \\ &= \min(n-i+j, (2i-2j-n)h), && m+1 \leqslant i-j < n. \end{aligned}$$

Now $d(T^*) = n$ and if

$$T = (1, m+1, n, m, 2m, \ldots, m+2, 1) = \{(i, i+m) : i \in N\}$$

then $d(T) = nh$ and because T is k-optimal $R_n \geqslant h = \Omega(n)$ for k fixed.

We must now show that T is indeed k-optimal. Let $X \subseteq T$ with $|X| = p \leqslant k$ and $Y \subseteq N^2$ satisfy $|Y| = p$, $Y \cap T = \phi$ and $T' = (T - X) \cup Y \in \mathrm{NHD}(T)$.

Now $d(T') = d(T) - ph + d(Y)$ and so if $Y \cap \mathrm{rev}(T) \neq \phi$ $d(Y) \geqslant kh$ implying $d(T') \geqslant d(T)$. We can thus restrict ourselves to proper k-swaps.

Next define for $i, j \in N$ $t(i, j)$ = the number of arcs in the subpath from i to j in T e.g. $t(1, m+1) = 1$ and $t(1, 2) = n - 2$.

Note that by symmetry $t(i, j) = t(i', j') \to d(i, j) = d(i', j')$. Suppose now that $T' = (T - X) \cup Y$ is a proper k-swap of T and $Y = \{(i_q, j_q) : 1 \leqslant q \leqslant p\}$.

Now as T' is a tour we have

$$\sum_{(i,j) \in T'} t(i,j) = an$$

for some integer $a > 0$ and as T' uses $n - p$ arcs of T we have

$$\sum_{q=1}^{p} t(i_q, j_q) = bn + p, \qquad b = a - 1. \tag{6}$$

We can clearly restrict our attention to the case where $d(i_q, j_q) < ph$ for $1 \leqslant q \leqslant p$. It follows that

(i) $1 \leqslant t(i_q, j_q) \leqslant p - 1$ and $d(i_q, j_q) = t(i_q, j_q)\, h$ or

(ii) $n - 2(ph - 1) \leqslant t(i_q, j_q) \leqslant n - 2$ and $d(i_q, j_q) = (n - t(i_q, j_q))/2$ ($t(i_q, j_q)$ is even in this case).

Partition $[p]$ into Q_1, Q_2 such that $q \in Q_1$ (resp. Q_2) if Case 1 (resp. Case 2) holds for q.

Then from (6) we have

$$\sum_{q \in Q} t(i_q, j_q) = (b - |Q_2|)\, n + p + \sum_{q \in Q_2} (n - t(i_q, j_q)). \tag{7}$$

We can now further restrict ourselves to the case where

$$\sum_{q \in Q_2} d(i_q, j_q) \leqslant ph - 1$$

which implies that

$$\sum_{q \in Q_2} (n - t(i_q, j_q)) \leqslant 2(ph - 1).$$

Using this in (7) together with the fact that $p + 2(ph - 1) < n$ we find that $b - |Q_2| \geqslant 0$ and hence that

$$\sum_{q \in Q_1} t(i_q, j_q) \geqslant p.$$

Thus

$$\sum_{q \in Q_1} d(i_q, j_q) \geqslant ph \text{ and so } d(T') \geqslant d(T)$$

and T is k-optimal.

In the symmetric case Rosencrantz, Stearns, and Lewis [25] describe an example of an $n/4$-optimal tour which is $2 - 1/n$ times the length of the optimum tour.

For maximum length tours Fisher, Nemhauser, and Wolsey [7] show that in the symmetric case a 2-optimal tour is at least one-half the length of the optimum tour but that in the nonsymmetric case 2-optimal tours can be a mere $4/n$ times the length of the maximum tour.

Papadimitrou and Steiglitz [23] consider interchange algorithms when (1) does not hold and give particularly nasty examples.

VI. REPEATED ASSIGNMENT HEURISTIC

In order to write the algorithm we need 2 additional procedures.

Procedure ASSIGN (C, D) solves the assignment problem defined on G by the cost matrix D (where $d_{ij} = \infty$ to exclude loops) i.e. it finds a subset S of N^2 or minimum cost such that in the graph $G' = (V, S)$ every vertex has in degree and out degree equal to 1. S defines, in general, a set of disjoint tours $P_1 \ldots, P_K$ and it is a solution of the directed non symmetric T.S.P. if $K = 1$.

Let T be a subset of A such that the graph $G'' = (V, T)$ is connected and for every node $v \in V$,

(i) in degree (v) = out degree $(v) = k(v)$.

(ii) the deletion of node v from G'' leaves $k(v)$ connected components.

Observe that these properties imply that for each connected component C_i obtained by deleting any node v there are nodes $u_i, w_i \in C_i$ such that (u_i, v) and (v, w_i) are in T (Fig. 1).

The following procedure modifies T until a tour of G is obtained.

Procedure TOUR (G, T).
  while there exists a vertex $v \in V$ with $k(v) > 1$ *do*
    begin
      add arc (u_1, w_2);
      delete arcs (u_1, v) and (v, w_2)
    end
  end

Notice that after execution of every add and delete operations the graph $G''(V, T)$ is connected and maintains the initial properties and therefore if $k(v) > 1$ it is always possible to find node u_1, w_1, u_2, w_2 as in Fig. 1.

Let G_0 be the given graph, D_0 be the given cost matrix and T_0 be the tour to be found.

The algorithm follows.

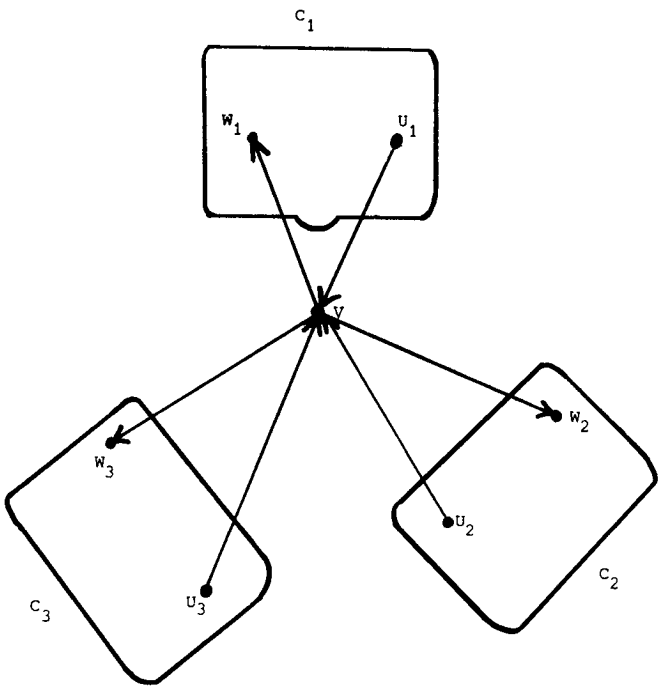

FIG. 1

Procedure DTSP (T_0, G_0, D_0).

```
begin
  T ← φ;
  D ← C_0;
  G ← G_0;
  k ← 2
  while k ≠ 1 do
    begin
      {P_1, P_2, ..., P_h} ← ASSIGN(G, C);
       V ← φ;
       for i = 1 until h do
       begin
         choose any node v_i of P_i;
         V ← V ∪ {v_i};
         T ← T ∪ {P_i}
       end
      Let G be the complete subgraph of G_0 induced by V
      and D be the induced cost matrix of G;
        k ← h
    end
      T_0 ← TOUR (G_0, T)
end
```

Let $\overline{T}$ be the set of arcs of G_0 to which TOUR is applied. It is straightforward to see that $\overline{G} = (V, \overline{T})$ is connected and has properties (i) and (ii).

Notice that ASSIGN is applied no more than $[\log_2 n]$ times and that the cost of the solution of every assignment problem is always not greater than the cost of the optimum tour T^* of G_0. It follows that the cost of the set of arcs $\overline{T}$ is bounded from above by $[\log_2 n]$ times the costs of T^* and so $R_n \leqslant [\log_2 n]$.

For what concerns the complexity of the algorithm, we know that the assignment problem can be solved in time $O(n^3)$. Since the algorithm applies ASSIGN on graphs whose number of vertices is at worst only halved at each iteration the total computing time spent on solving the assignment problems remains of the same order.

Since the complexity of TOUR is $O(n)$, the overall complexity of the algorithm is $O(n^3)$.

This algorithm can be easily generalized for solving the *multisalesman problem* with $R_n \leqslant [\log_2 n]$.

This problem asks for a minimum cost spanning connected subgraph of G having in degree and out degree equal to one at every vertex except vertex 1 which is allowed to have equal in and out degrees not exceeding a given constant m.

For the symmetric multisalesman problem Frieze [10] has modified Christofides heuristic to get $R_n \leqslant 3/2$.

VII. DATA DEPENDENT BOUNDS

We give here some upper bounds for algorithms that depend on the values l_{ij}.

(i) Suppose that in addition to (1) we have

$$l_{ij} \leqslant \alpha l_{ji}, \quad \text{for } i,j \in N, \tag{8}$$

where necessarily $\alpha \geqslant 1$.

For a given problem it will be easy to compute the smallest α satisfying (8). (α will be finite unless there exists i,j such that $c_{ij} > 0$ and $c_{ji} = 0$.)

We now give a simple modification of Christofides heuristic such that

$$R_n \leqslant 3\alpha/2 \tag{9}$$

holds.

Modified Christofides Heuristic

begin

for $i,j \in N$ *do* $d_{ij} := \min(l_{ij}, l_{ji})$;

Apply Christofides heuristic to the symmetric problem using arc lengths d_{ij}: i.e., (a) Construct a minimum length spanning tree T, (b) Construct a minimum weight matching M of the odd nodes X of T, (c) The graph defined by $T \cup M$ is eulerian. Orient the edges of $T \cup M$ to conform with the direction of some eulerian tour C. (d) Reduce the directed cycle C to a hamiltonian cycle H by traversing it and deleting repeated nodes.

We note that the time complexity is $O(n^3)$. (Note also the d need not satisfy (1) but this does not matter.)

Now

$$\begin{aligned} l(H) &\leqslant l(C) && \text{by (1)} \\ &\leqslant \alpha d(C) && \text{by (8)} \\ &= \alpha d(T) + \alpha d(M) \end{aligned}$$

But clearly $d(T) \leqslant d(H^*)$ and by considering the "reduction" of H^* to a tour through X we see that $l(H^*) \geqslant 2d(M)$.

We thus deduce that

$$\begin{aligned} l(H) &\leqslant \alpha d(H^*) + \alpha l(H^*)/2, \\ &\leqslant \alpha l(H^*) + \alpha l(H^*)/2 \end{aligned}$$

which implies (9).

We note that [4] shows that (9) is (almost) tight for $\alpha = 1$.

(ii) Suppose now that in addition to (1) we have

$$l_{ki} + l_{kj} \geqslant \beta l_{ij}, \quad \text{for } i,j,k \in N. \tag{10}$$

For a given problem it will again be easy to compute the largest β satisfying 0.1. This will satisfy $\beta \leqslant 2$ which can be deduced by considering all 6 possible inequalities (10) for a given set of nodes $x,y,z \in N$.

We now give a modification of a heuristic described in [8] for which

$$\begin{aligned} R_n &\leqslant 1 + 1/\beta, \quad 1 \leqslant \beta \leqslant 2, \\ &\leqslant 2/\beta, \quad 0 < \beta \leqslant 1. \end{aligned} \tag{11}$$

Branching Heuristic

Begin

Construct a minimum length branching B rooted at node 1 and spanning N, i.e., a minimum length set of $n - 1$ arcs containing no cycle and for which node 1 has in-degree 0 and every other node has in-degree 1. This can be constructed in $O(n^2)$ time–see Edmonds [6];

$$\text{Let } l(k, 1) = \min_{j \neq 1} (l(j, 1));$$

A: $$H := H \cup \{(k, 1)\}$$

At this stage H consists of a directed cycle C plus for each node x in C a (possibly empty) branching B_x rooted at x. These branchings are disjoint and for each node y not in C let $b(y)$ be the unique node in C such that y is a node of $B_{b(y)}$ and let $P(y)$ be the unique path from $b(y)$ to y in $B_{b(y)}$.

For a node x of C let $z(x)$ be the terminal node of the unique arc of C which has x as its initial node.

During subsequent iterations of the algorithm H and C will change and it is convenient to assume that the values of $b(y), P(y), u(x), z(x)$ are updated in concert.

The algorithm continues with

while H is not a hamiltonian cycle *do*

begin

Choose y not in C such that the outdegree of y is zero;

B: $H := (H \cup \{(y, z(b(y)))\}) - \{(b(y), z(b(y)))\}$ [see Fig. 2]

end

end.

One can easily implement the *while* loop so that the overall time complexity is $O(n^2)$. It remains to verify (11). It is clear that if $\hat{H}, \hat{C}$ are the values of H, C immediately after Step A that $l(\hat{H}) \leqslant l(H^*)$.

A single iteration of the *while* loop with a given y and $b = b(y)$, $z = z(b)$ changes

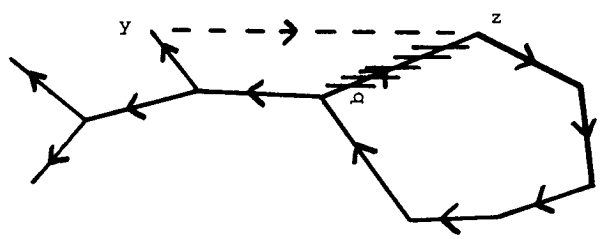

FIG. 2

$l(H)$ by

$$\begin{aligned} & l(y,z) - l(b,z) \\ & \quad \leqslant (l(b,y) + l(b,z))/\beta - l(b,z), \quad \text{by (10)}, \\ & \quad \leqslant l(P(y))/\beta + l(b,z)(1/\beta - 1), \quad \text{by (1)}. \end{aligned}$$

We first consider $\Delta_1 = \Sigma_y l(P(y))$ where the summation is over the y occurring in an execution of the whole algorithm. Now $\Delta_1 = l(\hat{H} - \hat{C})$. This is because the arcs occurring in $P(y)$ are all in H-C prior to execution of Step B of the algorithm and after execution of Step B these arcs will now be in C. Thus the arcs that contribute to Δ_1 are those of $\hat{H} - \hat{C}$.

Next let $\Delta_2 = \Sigma_y l(b, z)$, where the summation is over the same set of y as before. Now an arc can contribute at most once to Δ_2 as after an execution of Step B (b, z) is deleted from H and it will never reappear as both b, z are in C.

Also before execution of Step B the out-degree of b is at least 2. But this implies that (b, z) must be an arc of $\hat{H}$. It cannot be one of the arcs $(\hat{y}, \hat{z})$ added in a previous execution of Step A as this $\hat{y}$ will have out-degree 1 for the remainder of the algorithm. Thus $\Delta_2 \leqslant l(\hat{H})$ and so the length of H on termination of the algorithm is bounded above by

Case 1. $\beta \geqslant 1$. $l(\hat{H}) + \Delta_1/\beta = l(\hat{H}) + l(\hat{H} - \hat{C})/\beta$,

$$\leqslant l(\hat{H}) + l(\hat{H})/\beta.$$

Case 2. $0 < \beta \leqslant 1$. $l(\hat{H}) + \Delta_1/\beta + \Delta_2(1/\beta - 1)$,

$$\begin{aligned} & \leqslant l(\hat{H}) + l(\hat{H} - \hat{C})/\beta + l(\hat{H})(1/\beta - 1), \\ & \leqslant 2l(\hat{H})/\beta. \end{aligned}$$

and (11) follows immediately.

We note that [8] shows that 7.4 is (almost) tight for $\beta = 1$.

VIII. CONCLUSION

We have examined many polynomial time heuristics and found that most have worst-case ratio R_n which is $O(n)$ and only one, the repeated assignment heuristic of Sec. VI which has $R_n \leqslant \lceil \log_2 n \rceil$ is better than this.

Thus the question as to whether there is a polynomial time heuristic for which R_n is bounded by a constant posed in Karp [17] is still open.

References

[1] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem. *Mathematical Programming*. To appear.

[2] G. Clarke, and J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* 12 (1964) 568–581.

[3] S. A. Cook, The complexity of theorem proving procedures. *Proceedings 3rd Annual ACM Symposium on Theory of Computing* (1971) 151–158.

[4] G. Cornuéjols, and G. L. Nemhauser, Tight bounds for Christofides travelling salesman heuristic. *Math. Program.* 14 (1978) 116–121.

[5] J. Edmonds, Matroids and the greedy algorithm. *Math. Program.* 1 (1971) 127–136.

[6] J. Edmonds, *Optimum Branchings, Mathematics of the Decision Sciences Part 1*, American Mathematical Society (1968) 346–364.
[7] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, An analysis of approximations for finding a maximum weight hamiltonian circuit. Unpublished.
[8] A. M. Frieze, Worst-case analysis of algorithms for travelling salesman problems. *Oper. Res. Verfahren* 32 (1979) 93–112.
[9] A. M. Frieze, A generalisation of the greedy algorithm for independence systems. Unpublished.
[10] A. M. Frieze, An extension of Christofides' Heuristic to the k-person travelling salesman problem. Submitted to *Discrete Appl. Math.*
[11] M. R. Garey, and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman City (1979).
[12] B. L. Golden, Evaluating a sequential routing algorithm. *A.I.I.E. Trans.* 9 (1977) 204–208.
[13] D. Hausmann, T. A. Jenkyns, and B. Korte, Worst-case analysis of greedy type algorithms for independence systems. Report No. 7781-OR, University of Bonn (1977).
[14] T. A. Jenkyns, The efficacy of the greedy algorithm. *Proceedings 7th S-E Conference on Combinatorics, Graph Theory and Computing* (1976) 341–350.
[15] L. L. Karg, and G. L. Thompson, A heuristic approach to travelling salesman problems. *Manag. Sci.* 10 (1964) 225–248.
[16] R. M. Karp, Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller, and T. W. Thatcher, Eds. Plenum, New York, (1973) 85–103.
[17] R. M. Karp, The fast approximate solution of hard combinatorial problems. *Proceedings 6th South Eastern Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematical, Winnipeg (1975) 15–31.
[18] B. Korte, and D. Hausmann, An analysis of the greedy heuristic for independence systems. In *Algorithmic Aspects of Combinatorics, Ann. Discrete Math.* 2, B. Alspach, P. Hall, and D. J. Miller, Eds. North Holland, New York (1978) 65–74.
[19] S. Lin, Computer solution of the travelling salesman problem. *Bell System Tech. J.* 44 (1965) 2245–2269.
[20] S. Lin and B. W. Kernighan, An effective heuristic algorithm for the travelling salesman problem. *Oper. Res.* 21 (1973) 498–516.
[21] T. R. Minina, and V. T. Perekrest, On a method of approximating solutions to the travelling salesman problem. *Sov. Math.* 16 (1975) 26–30.
[22] T. A. J. Nicholson, A sequential method for discrete optimisation problems and its application to the assignment, travelling salesman and three-machine scheduling problems. *J. Inst. Math. Appl.* 3 (1967) 362–375.
[23] C. H. Papadimitrou, and K. Steiglitz, Some difficult examples of the travelling salesman problem. *Oper. Res.* 26 (1978) 434–443.
[24] S. Reiter and G. Sherman, Discrete Optimizing. *S.I.A.M. J. Appl. Math.* 13 (1965) 864–889.
[25] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, Approximate algorithms for the travelling salesperson problem. *Proceedings of 15th IEEE Symposium on Switching and Automata Theory* (1974) 33–42.
[26] S. Sahni, and T. Gonzales, *P*-complete approximation problems. *J. ACM* 23 (1976) 555–565.
[27] P. Van Der Cruyssen, and M. J. Rijckaert, Heuristic for the asymmetric travelling salesman problem. *J. Oper. Res. Soc.* 29 (1978) 697–701.
[28] M. H. J. Webb, Some methods of producing approximate solutions to travelling salesman problems with hundreds or thousands of cities. *Oper. Res. Q.* 22 (1971) 49–66.

Received September 8, 1980
Accepted June 27, 1981