## OREFDR STRTE

## UTHERSITT

## EDTाPUTER

DEPARYMENT OF
COMPUIEA SCIENCE
SElEПEE

## IEPRRTITETT

# Tours of <br> Graphs, Digraphs and Sequential Machines 

PAUL CULL
Computer Science Dept.
Oregon State University

TOURS OF
GRAPHS, DIGRAPHS AND SEQUENTIAL MACHINES

Paul Cull
Department of Computer Science Oregon State University
Corvallis, Oregon 97331


#### Abstract

A tour of a graph (digraph, or sequential machine) is a sequence of nodes from the graph such that each node appears at least once and two nodes are adjacent in the sequence only if they are adjacent in the graph. Finding the shortest tour of a graph is known to be an NP-complete problem. Several theorems are given that show that there are classes of graphs in which the shortest tour can be found easily. For more general graphs, we present approximating algorithms for finding short tours. For undirected graphs, the approximating algorithms give tours at worst a constant times the length of the shortest tour. For directed graphs, the size of the calculated tour is bounded by the size of the digraph times the shortest tour. Not only are the bounds worse for the directed case, but the running times of the approximating algorithms are also larger than those for the undirected case.


INDEX TERMS--short tours, Hamiltonian circuits, sequential machines, knight's tour, traveling salesman, approximating algorithms, NP-problems

Many important problems in computer science and engineering can be represented as problems concerning graphs. Thus finding efficient algorithms for graph problems can be a worthwhile activity. Unfortunately, a number of the important graph problems seem to be hard, in the sense that they may have no algorithm whose running time is bounded by a polynomial in the size of the problem. The NP-complete problems seem to be hard in this sense. [5, 11]

If the problems with which we are concerned are assumed to be hard, how shall we look for efficient algorithms? One way is to attack the NP-complete problems and show that any one of them has a polynomial time-bounded algorithm. This would give polynomial algorithms for all NP-problems. But this approach is not likely to be very fruitful. Some of these problems have been discussed for 200 years and no reasonable algorithms have yet been found. In fact, when all the problems were shown to be polynomially related, it probably increased the belief that these problems are in fact hard. One might reasonably assume that a proof that NP-complete problems are hard will shortly be available.

A second approach is to simply accept that these problems are hard and use the various tricks of the trade to try to speed up the algorithms. Unfortunately, in this case the running time will still grow very, very rapidly with the size of the problem. If these problems are of any practical concern it is unlikely that we will want to wait months, years or centuries for a computer to print out the answer.

A more reasonable approach is to change the problem. For example, NPproblems are stated as recognition problems. Does this graph have such and such a property? Many of these problems can be restated to say: Find the shortest (or longest) something in this graph. This does not make the problem any easier, since to find the shortest something would give an answer to the recognition problem. But if these are problems of practical import, then it might not be necessary to actually find the shortest if we can find something that is almost as short, at least if we can find it quickly. This approach may be summed up by saying that a recognition problem is replaced by an optimization problem, and fast approximate solutions are sought to the optimization problem. There is a growing literature on this approach showing that some problems can be approximated arbitrarily closely, some with a constant multiplicative bound, and that some cannot be approximated within a multiplicative bound unless NP-complete problems are easy (not hard). [10, 12, 15]

Another approach is to look for tractable subclasses of hard problems. Do you really need an algorithm that works for all graphs, or only those that arise in practice? Do those that arise in practice have some extra properties that make them easier to deal with? Having tractable subclasses is also useful when you are investigating the behavior of the approximating algorithms mentioned above. Although you may prove a bound on how closely the approximating algorithm approximates the optimum, this is likely to be a worst case bound, so you may like to see how well the approximating algorithm works on problems to which you know the answer.

In this paper, we will discuss tour problems. We will show that even though the problems are NP-complete, several interesting subclasses are tractable. We will also show that there are approximating algorithms for tour problems.

## DEFINITIONS

A directed graph or digraph is a pair (V,E) where $V$ is a finite set of objects called vertices or nodes, and E, the set of edges or arrows, is a subset of $V x V$. If for every edge ( $x, y$ ) there is also the edge $(y, x)$, then the digraph is an undirected graph or simply, a graph. A sequential machine is a finite set of objects called states, together with a finite set of mappings from the state set to itself. The digraph of the sequential machine consists of a vertex set which is the state set of the machine, and an edge set that contains an arrow ( $x, y$ ) if and only if there is a mapping of the machine that takes state $x$ to state $y$.

A tour of a graph or digraph is a sequence of nodes such that each node appears in the sequence at least once, and such that two nodes are adjacent in the sequence only if there is an edge from the first node to the second node. If the tour is allowed to start at some node and finish at a different node, it is an open tour. If the tour is required to start and finish at the same node, it is a closed tour. A tour of a sequential machine is a tour of the machine's digraph.

There are a number of tour problems that are known to be NP-complete:
(1) Hamiltonian problems. Given a graph or digraph, does it have an open or closed tour in which each node appears exactly once. For an open tour, this is called a Hamiltonian path, For a closed tour, this is called a Hamiltonian circuit.
(2) Shortest tour problems. Given a graph or digraph, find the shortest open or closed tour, in the sense that it contains the fewest nodes counting repetitions.
(3) Traveling salesman problems. Given a graph or digraph with positive weights assigned to each edge, find an open or closed tour that is shortest in the sense that the sum of the weights assigned to edges between adjacent nodes in the tour is a minimum.

Clearly, the traveling salesman problems are most general, but each of the other problems generates some interest in its own right.

## HAMILTONIAN PROBLEMS

There are many sufficient conditions for the existence of Hamiltonian circuits in graphs. The simplest of these is that a complete graph, that is, a graph with an edge between every pair of states, has a Hamiltonian circuit. In fact, every permutation of the sequence of nodes is a Hamiltonian circuit of a complete graph. This is the most extreme example of what I would call a density theorem. A density theorem says that if a graph has enough edges, then it has a Hamiltonian circuit. Let $d_{i}$ be the degree of vertex $i$, that is, the number of other vertices that can be reached in one step from vertex i. The following examples of density theorems are from [3].

THEOREM 1: $G$ has a Hamiltonian circuit if:
(1) $d_{i} \geq \pi / 2$
or (2)
$\left(v_{i}, v_{j}\right) \& E \Rightarrow d_{i}+d_{j} \geq n$
or (3) $d_{i} \leq i \Rightarrow i \geq(n-1) / 2$,
and when $n$ is odd, $d_{(n+1) / 2 \geq(n+1) / 2}$
or (4) $d_{i} \leq i, d_{j} \leq j, i \neq j \Rightarrow d_{i}+d_{j} \geq n$
One of the classic Hamiltonian problems is the knight's tour problem, Given a chessboard, is it possible to start a knight in some square and following allowed moves, have it enter each square once and return to the starting square? The knight is the chess piece that usually looks like a horse. If the knight is on square ( $\mathrm{i}, \mathrm{j}$ ), then it is allowed to move to any of the eight squares ( $\mathrm{i} \pm 1, \mathrm{j} \pm 2$ ) and ( $i \pm 2, j \pm 1$ ), if these squares are on the board. Euler investigated this problem almost 200 years ago, and showed that a knight's tour was possible on the standard $8 \times 8$ chessboard and on several other boards. Until last year, I thought that the conditions for a knight's tour on a rectangular board were well known. But a
search of the literature failed to turn up a general proof, se we constructed our own.

One thing that is well known is that a Hamiltonian closed knight's tour is impossible on a rectangular board with an odd number of squares. The proof is rather trivial. Recall that a chessboard has squares of two colors, usually red and black. If the knight is on a square of one color, then it can only move to a square of the other color. If the board has an odd number of squares, then there must be more of one color than of the other. So if a tour starts on a square of the more numerous color, it will end on a square of that color when every square has been visited once. But it is then impossible to return to the starting square in one step since it is of the same color. If the knight starts on a square of the less numerous color, then it will not even be able to visit all the squares before it runs out of squares of the less numerous color. Even though this argument rules out closed Hamiltonian tours on odd boards, it does not rule out open Hamiltonian tours.

We have been able to prove the following theorem.
THEOREM 2: On an $n \times m$ rectangular board with $\min (n, m) \geq 5$, there is an open Hamiltonian knight's tour. If nm is even, then there is a closed Hamiltonian knight's tour.

The proof is too long to include here. Suffice it to say that it is an inductive proof based on breaking the board into $5 \times 5$ boards and some residual boards, and using the tours on these small boards to piece together a tour for the whole board. The inductive base consisted of 37 boards. We refer the interested reader to [6].

Another approach to finding knight's tour is Warnsdorff's rule, which states that the knight should go to the allowed square of lowest degree. The degree of a
square is the number of unvisited squares that can be reached from it in one step. Unfortunately, Warnsdorff's rule does not say what to do when there are several squares of lowest degree. Adopting the strategy that the knight can go to any square of lowest degree may not work. Ball [2] states that there are cases in which using Warnsdorff's rule with this strategy will lead to a dead end even when there is a tour. Pohl [14] has proposed using a higher order Warnsdorff's rule in which ties are broken by looking at the degrees of squares that can be reached in one step from the tied squares. Unfortunately, ties may persist and further tiebreaking may be necessary. Continued use of the tie-breaking procedure should work, but at an expense that may be exponential in the size of the board. Pohl includes an example in which the tie-breaking tree has as many levels as the number of nodes in the graph. On the other hand, he states that first level tie-breaking with arbitrary choice in case of second order ties was always sufficient to find a tour on an $8 \times 8$ chessboard.

If we consider directed graphs, there are not as many known sufficient conditions. The following is a recent result. Let $S(X)$ be the set of nodes reachable in one step from node $X$, i.e., $S(X)=\{Y \mid(X, Y) \varepsilon E\}$. Let $\equiv$ be the equivalence relation defined by $X_{1} \equiv X_{2}$ if and only if $S\left(X_{1}\right)=S\left(X_{2}\right)$. The digraph $D$ is nice if $S\left(X_{1}\right) \cap S\left(X_{2}\right) \neq \emptyset$ implies $X_{1} \equiv X_{2}$, and if there is a number $k$ such that each equivalence class has $k$ members and a number $j$ such that each $S(X)$ has $j$ members. A digraph is strongly connected if for every pair of nodes there is a directed path from the first to the second.

THEOREM 3: If $D$ is a strongly connected digraph and $D$ is nice, then $D$ has a directed Hamiltonian circuit.

PROOF: Since each node is reached from some equivalence class and the next node sets of two equivalence classes are nonoverlapping, we have $j C=|D|$,
where $j$ is the number of next nodes of an equivalence class, $C$ is the number of equivalence classes, and $|D|$ is the number of nodes in $D$. We a1so have that $k C=|D|$, where $k$ is the number of nodes in an equivalence class. Thus $j=k$.

We now set up a one-to-one correspondence between $E$, an equivalence class, and $S(E)$, the next node set of $E$, such that the first member of each pair has an arrow to the second member of its pair. Since every member of $E$ has an arrow to every member of $S(E)$ and $|E|=|S(E)|$, it is trivial to set up the required correspondence. Since equivalence classes are nonoverlapping and exhaust the set, and next node sets of equivalence classes are nonoverlapping and exhaust the set, setting up the one-to-one correspondence between each $E$ and its $S(E)$ gives us a one-to-one correspondence from $D$ to $D$. But every one-to-one correspondence from a set to itself is a permutation and thus decomposes the set into a set of disjoint cycles.

We now connect these cycles together to form a Hamiltonian circuit. Since D is strongly connected, there is either one cycle which is a Hamiltonian circuit, or there are several cycles and there is an arrow from some node on a cycle to some node not on the cycle. Say $v_{11}$ on cycle one has an arrow to $v_{22}$ on cycle two; then there is a node $v_{21}$ (which may be $v_{22}$ ) on cycle two that has an arrow to $v_{22}$. But then $v_{11} \equiv v_{21}$, so they have the same set of next nodes. In particular, $v_{11}$ goes to $v_{12}$ (which may be $v_{11}$ ) on cycle one and so does $v_{21}$. Thus starting at $v_{11}$ we may go to $v_{22}$, go into every node on cycle two, and end in $v_{21}$. But then we can jump to $v_{12}$, enter every node on cycle one, and end in $v_{11}$. Thus from two cycles we have built a larger cycle. By the strong connectedness of $D$, we can continue joining cycles until all nodes are in a single cycle, which is our desired Hamiltonian circuit.

COROLLARY: A strongly connected digraph $D=(V, E)$ has a Hamiltonian circuit if and only if there is a strongly connected subgraph $D^{\prime}=\left(V, E^{\prime}\right)$ with $E^{\prime} \subseteq E$ such that $D^{\prime}$ is nice.

PROOF: If $D^{\prime}$ is nice, then $D^{\prime}$ has a Hamiltonian circuit.
Since the vertex set of $D^{\prime}$ is the vertex set of $D$, and the edges of $D^{\prime}$ are some of the edges of $D, D$ has a Hamiltonian circuit. If $D$ has a Hamiltonian circuit, then let $D^{\prime}$ be ( $V, E^{\prime}$ ), where an edge is in $E^{\prime}$ if and only if it is in the Hamiltonian circuit of D. Applying $\equiv$ to $D^{\prime}$, we obtain equivalence classes and next node sets that have exactly one member. So $D^{\prime}$ is nice.

The theorem can be used to show that several classes of sequential machines have directed Hamiltonian circuits. Let $R$ be a finite ring. Then $X_{t+1}=F\left(X_{t}\right)+B Y_{t}$ is a sequential machine with linear inputs over $R$, where $X$ and $Y$ are $n$-dimensional vectors over $R, B$ is an $n \times n$ matrix over $R$, and $F$ is a function from $R^{n}$ to $R^{n}$. A sequential machine is strongly connected if its digraph is.

THEOREM 4: If $M$ is a strongly connected sequential machine with linear inputs, and $F$ is a one-to-one function, then the digraph of M has a directed Hamiltonian circuit.

PROOF: Clearly the digraph of the machine is strongly connected. If two
states $X_{1}$ and $X_{2}$ have a next state in common, then $F\left(X_{1}\right)+B Y_{1}=F\left(X_{2}\right)+B Y_{2}$. But then $F\left(X_{1}\right)+B Y_{3}=F\left(X_{2}\right)+B\left(Y_{2}-Y_{1}+Y_{3}\right)$, so they have all next states in common. The size of the next state set depends solely on $B$, so all next state sets are the same size. If $F$ is one-to-one, then the number of states in an equivalence class depends solely on B so all equivalence classes are the same size. Thus the digraph of $M$ is nice and by Theorem 3 has a Hamiltonian circuit.

COROLLARY; A strongly connected sequential machine with linear inputs has a Hamiltonian circuit if and only if its digraph can be decomposed into a set of nonoverlapping cycles by ignoring some of the arrows.

If there is a cyclic decomposition we can use it to define a one-to-one $F$ and Theorem 4 applies. The only if part follows since the Hamiltonian circuit is a trivial cyclic decomposition.

If in the previous definition $F\left(X_{t}\right)=A X_{t}+C$, where $A$ is an $n x n$ matrix, and $C$ is an $n$-dimensional vector, we call $M$ an affine sequential machine. If further $C=0$, then $M$ is a linear sequential machine.

THEOREM 5: If $M$ is a strongly connected affine sequential machine, then the digraph of $M$ has a directed Hamiltonian circuit.

PROOF: By the proof of the previous theorem, if two next node sets overlap they are identical, and the sizes of next node sets are identical. Two states are equivalent if and only if there exist $Y_{1}, Y_{2}$ such that $A\left(X_{1}-X_{2}\right)=B\left(Y_{1}-Y_{2}\right)$. Thus each equivalence class is a coset of the equivalence class of 0 . Thus all equivalence classes are the same size. Thus by Theorem 3, M has a Hamiltonian circuit.

COROLLARY: If $M$ is a strongly connected linear sequential machine, then $M$ has a directed Hamiltonian circuit.

SHORTEST TOUR PROBLEMS

Shortest tour problems have the form: Given a graph (or digraph) $G$ and a node $q$, find $T(q)$, the length of the shortest open (or closed) tour starting at node $q$. Or find $T_{\text {min }}$, the minimum of the $T(q)^{\prime} s$, and $T_{\max }$, the maximum of the $T(q)^{\prime} s$. Unfortunately, all these problems are NP-complete, so it is unlikely that there is any reasonable algorithm for computing them. Instead we will look for bounds on
$\mathrm{T}_{\min }$ and $\mathrm{T}_{\max }$. For closed tours $\mathrm{T}_{\max }$ and $\mathrm{T}_{\min }$ are identical, so we call their common value $T_{c}$.

Consider a connected graph. Clearly we can find a spanning tree quickly. In fact [1], we can find a depth first spanning tree in $O(E)$ steps. The spanning tree will have $n-1$ edges. To obtain a closed tour we can start at any node, visit the other nodes, and return by using each edge of the tree twice. Thus, $\mathrm{n} \leq \mathrm{T}_{\mathrm{c}} \leq 2(\mathrm{n}-1)$, where the lower bound comes from the fact that each node must be entered at least once. These bounds are of course attainable. The lower bound is attained by any graph with a Hamiltonian circuit. The upper bound is attained by any graph that is a tree. Another bound is $T_{c} \leq 2 n-k$, where $k$ is the length of a cycle in the graph. To make the bound as good as possible, it would be nice to have the longest cycle. Unfortunately, finding the length of the longest cycle is as hard as finding the shortest tour.

Turning to open tours, it is easy to see that $n-1 \leq T_{\max } \leq 2 n-3$, and that $n-1 \leq T_{\min } \leq 2(n-1)-D \leq 2(n-2)$. The lower bounds come from the fact that every node except the starting node must be entered at least once. These lower bounds are attainable. For $T_{\max }$ the lower bound is attained when there is a Hamiltonian path starting at any node. The lower bound on $T_{\min }$ is attained when the graph has a Hamiltonian path. The upper bound on $T_{\text {min }}$ contains $D$, which should be the maximum path length between any two nodes. Unfortunately, on this interpretation $D$ is hard to calculate. Instead we can take $D$ to be the maximum path length found between any two nodes. This is certainly at least as great as the maximum distance between any two nodes. D must be at least 2 since if $D$ is 1 the graph is completely connected and has a Hamiltonian circuit. Consider the following graph:


The longest open tour starts from node 1. Each edge except one must be traversed twice, giving the upper bound on $T_{\max }$. The shortest tour starts from any node except node 1. Each edge except two must be traversed twice giving the upper bound on $T_{\text {min }}$.

The results on both open and closed tours can be summed up by:

$$
\operatorname{ALG}(\mathrm{G}) \leq 20 \mathrm{PT}(\mathrm{G}),
$$

where by OPT(G) we mean the length of the shortest tour of graph $G$, and by ALG(G) we mean the length of the tour that is found quickly. This is of course a worst case bound and in practice the computed tour may come very close or find the shortest tour. The constant 2 in the above inequality can be reduced to 1.5 by using an algorithm of Christofides [4] which we will describe in the section on traveling salesman problems.

Turning to digraphs, we can also obtain bounds. Let $D$ be the maximum distance between two nodes. Then there is a path of length $D$, on which $D+1$ nodes including the starting node are visited. To visit any other node takes at most D steps. Thus we obtain the following bounds:

$$
\begin{aligned}
& \mathrm{n}-1 \leq \mathrm{T}_{\min } \leq D(\mathrm{n}-\mathrm{D}) \leq\left[^{(\mathrm{n} / 2)^{2}}\right\rfloor \\
& \mathrm{n}-1 \leq \mathrm{T}_{\max } \leq D(\mathrm{n}+1-\mathrm{D})-1 \leq \mathrm{L}^{((\mathrm{n}+1) / 2)^{2}}-1 \\
& \mathrm{n} \leq \mathrm{T}_{\mathrm{c}} \leq D(\mathrm{n}+1-\mathrm{D}) \leq \mathrm{L}^{((\mathrm{n}+1) / 2)^{2}},
\end{aligned}
$$

where the upper bounds are obtained by maximizing the expression involving D by a trivial differentiation, and the lower brackets indicate the greatest integer
function. The upper bounds are attained in the following digraph.


The longest open tour starts at node 1 . The shortest open tour can start at any of the nodes $\left\lfloor^{(n-1) / 2} \beth^{+1}\right.$ through $n$.

Finding an upper bound on an open tour in a digraph is a standard problem in switching theory $[9,13]$. Usually the problem is to show that $(n(n-1)) / 2$ is an upper bound on $T_{\max }$. Our bounds are better and are attainable. The problem of finding a best upper bound on $T_{\max }$ has also been investigated by Dewdney and Szilard [8]. They find by a more complicated procedure the correct upper bound on $T_{\text {max }}$, and conjecture an upper bound on $T_{\text {min }}$ which turns out to be incorrect when n is even.

The bounds on tours of directed graphs may be summarized by:

$$
\operatorname{ALG}(G) \leq \min (D, n / 4, n-D) \operatorname{OPT}(G) \leq(n / 4) \operatorname{OPT}(G),
$$

where we have ignored all but the leading term. Of course, these are worst case bounds so the behavior of the algorithms may be better in practice.

We tested several variants of the algorithms for finding open directed tours on the knight's tour problem on square boards. As we have mentioned, these boards always have open knight's tours. The maximum distance between two squares
on a board is approximately $\sqrt{n}$. So from the above relation we could expect our algorithm to find a tour that was at worst $\sqrt{n}$ times the shortest tour. In fact, our algorithm behaved much better, finding a tour that was at most OPT +3 .

## TRAVELING SALESMAN PROBLEMS

Although traveling salesman problems look difficult, the techniques of the previous section can be used to calculate fast approximate solutions.

In the undirected case, instead of finding a spanning tree, one finds a minimum weight spanning tree. Then since a tour must include a spanning tree, the tour must have a weight at least as great as the sum of the weights of the edges in the minimum weight spanning tree. Of course, to tour the minimum spanning tree, each edge must be traversed at most twice, so we obtain:

$$
\operatorname{OPT}(\mathrm{G}) \leq \mathrm{ALG}(\mathrm{G}) \leq 20 \mathrm{PT}(\mathrm{G}) .
$$

The only difference between this algorithm and the one of the previous section is that this algorithm must form a minimum weight spanning tree. Using standard algorithms [1], this would take $O(E \log E)$. This approach seems to be well known and is mentioned in [12].

An algorithm that produces a traveling salesman's tour that is at worst 1.5 times as long as the shortest traveling salesman's tour has recently been described by Christofides [4]. As above, one finds a minimum weight spanning tree. Since the sum of the degrees of the nodes in a tree is even, there are an even number of nodes with odd degree. One now pairs the odd degree nodes so that the
sum of the weights between members of pairs is minimum. Now considering only the tree edges and the pair edges, each node has even degree, thus there is an Euler circuit which uses each edge exactly once and visits each node at least once. The weight of this tour will be the sum of the weights of the tree edges plus the sum of the weights of the pair edges. Since any tour must have weight at least as great as the minimum weight spanning tree and since the sum of the weights of the pair edges can be at most $1 / 2$ the weight of a tour, we obtain:

$$
\mathrm{OPT}(\mathrm{G}) \leq \operatorname{ALG}(\mathrm{G})<1.5 \mathrm{OPT}(\mathrm{G})
$$

for this algorithm. In worst case, the running time of this algorithm will be dominated by the time to find the minimum pairing of the odd degree nodes which can take $O\left(\mathrm{n}^{3}\right)$. Thus the running time of the algorithm is $O\left(n^{3}\right)$.

This algorithm can also be used to find a short closed tour of an unweighted graph by assigning weight 1 to all edges in the graph and by assigning to each edge not in the graph a weight equal to the minimum distance between the two nodes on the edge. Since computing these weights can be done in $O\left(n^{3}\right)$ time, the algorithm still has $O\left(n^{3}\right)$ running time.

We can also obtain bounds on the directed traveling salesman problem. Assume there are two nodes distance $D$ apart, where $D$ is the maximum distance between any two nodes. (Here we mean directed distance; we do not assume symmetry of distances.) If there are $M$ nodes on this path then there is a closed tour of length at most $D(n+1-M)$. On the other hand, any closed tour is at least as long as $D+(n-M) d$, where $d$ is the minimum distance between any two nodes. Thus:

$$
D+(n-M) d \leq O P T \leq A L G \leq D(n+1-M) .
$$

Since the optimum is clearly at least as great as D, we obtain:

$$
\mathrm{OPT} \leq \mathrm{ALG} \leq(\mathrm{n}+1-\mathrm{M}) \mathrm{OPT} \leq \mathrm{nOPT} .
$$

To calculate a tour for a directed graph, one should first find the distance matrix, whose $i, j$ entry is the directed distance from node $i$ to node $j$. (Even if distances are given between every pair of nodes, the actual distances may be shorter, since we can go from one node to the other by passing through several other nodes.) There are standard methods for computing the distance matrix $[1,8]$ which give as a by-product a list of the nodes visited in going from one node to the other. This information may be used to select the next node to visit. For instance, we might wish to visit next the node on whose path we repeat the fewest already visited nodes, or visit the most unvisited nodes, or perhaps try to do both. Simply calculating, the distance matrix is $O\left(\mathrm{n}^{3}\right)$, and we can build a tour-finding algorithm with this bound. Adding various refinements to try to make the tour shorter can drive the cost of the algorithm to $O\left(n^{5}\right)$.

## CONCLUSION

Although tour problems are very probably hard, solving them in practice might not be hard. The graphs that occur in practice might have special structure, and one of the theorems mentioned in this paper might be used to show that a Hamiltonian path or circuit exists. In problems of the shortest tour or traveling salesman type, there are fast approximating algorithms. For the undirected case, these may be satisfactory since they guarantee a tour no worse than a constant times the shortest tour. In the directed case, the bound only assures that the calculated tour is no longer than $n$ times the shortest tour. Although this bound may not be satisfactory, it is a worst case bound, so calculated tours may be much closer to shortest tours.

The directed case seems to be harder than the undirected case. Not only are the bounds worse, but the computation time for the approximating algorithms for the directed case have longer running times than the approximating algorithms for the undirected case.

## REFERENCES

[ 1] A. Aho, J. Hopcroft, and J. Ullman, The Design and Analysis of Computer Algorithms. Reading, Mass.: Addison-Wesley, 1974.
[ 2] W.W.R. Ba11, Mathematical Recreations and Essays. New York: Macmillan, 1947, 174-184.
[3] J. Bondy, "Cycles in Graphs" in Combinatorial Structures and their Applications. New York: Gordon and Breach, 1970, 15-18.
[ 4] N. Christofides, Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Management Sciences Research Report No. 388. Pittsburgh: Carnegie-Mellon University, 1976.
[5] S.A. Cook, "The Complexity of Theorem Proving Procedures," Proc. 3rd ACM Symposium on Theory of Computing, 151-158, 1971.
[6] P. Cull and J. DeCurtins, "Knight's Tour Revisited," Fibonacci Quarterly, 1978.
[7] N. Deo, Graph Theory with Applications to Engineering and Computer Science. Englewood Cliffs: Prentice-Hall, 1974, 297-301.
[8] A.K. Dewdney and A.L. Szilard, "Tours in Machines and Digraphs," IEEE Trans. Comput., Vol. C-22, pp. 635-638, 1973.
[9] F.G. Hennie, Finite State Models for Logical Machines. New York: Wiley, 1968, 45.
[10] D. Johnson, "Approximate Algorithms for Combinatorial Problems," J. Comput. and Syst. Scis., Vo1. 9, pp. 256-278, 1974.
[11] R.M. Karp, "Reducibility among Combinatorial Problems," in Miller and Thatcher (eds.), Complexity of Computer Computations. New York: Plenum Press, 1972, 85-103.
[12] R.M. Karp, "The Fast Approximate Solution of Hard Combinatorial Problems," Proc. 6th S-E Conf. Combinatorics, Graph Theory, and Computing, 1976, 15-31.
[13] Z. Kohavi, Switching and Finite Automata Theory. New York: McGraw-Hill, 1970, 306.
[14] I. Poh1, "A Method for Finding Hamiltonian Paths and Knight's Tours," CACM, Vol. 10, pp. 446-449, 1967.
[15] S. Sanhi and T. Gonzalez, "P-Complete Problems and Approximate Solutions," JACM, Vo1. 23, pp. 555-565, 1976.

