On the Complexity of Edge Traversing

CHRISTOS H. PAPADIMITRIOU

Princeton University, Princeton, New Jersey

ABSTRACT It is shown that the Chinese Postman Problem, although tractable in the totally directed and the totally undirected cases, is NP-complete in the mixed case A simpler version of the same problem is shown algorithmically equivalent to the max-flow problem with unit edge capacities

KEY WORDS AND PHRASES mixed graphs, Eulerian graphs, unicursality test, Chinese Postman Problem, NP-complete problems

CR CATEGORIES. 5 25, 5 32, 5 4

1. *Introduction*

Graphs with a distance function defined on their edges have been used extensively to model situations where services have to be delivered at particular places with minimum total traversed distance. When the locations of delivery are the nodes of the graph, the resulting problem is the Traveling Salesman Problem (TSP). This problem appears to be particularly hard, since there is evidence that there is no polynomial time algorithm for its solution, even if the goal is less ambitious than finding the optimal tour ([14, 13]), or if the domain of the problem is substantially restricted [9, 12].

Nevertheless, there are applications, like school bus scheduling and mail delivery, in which all edges of a graph, instead of all nodes, must be visited. In fact, the oldest problem in graph theory [5] is such a problem. Edge oriented problems can generally be reduced to node oriented problems, but this is hardly a gain, since node oriented problems appear to be more difficult [4]. Theorem 1 in the present paper is essentially a reduction in the opposite direction.

The Chinese Postman Problem (CPP) [11, 3] is a generalization of the Eulerian path problem. In the CPP we are allowed to traverse each edge more than once, and our goal is to traverse each edge at least once at a minimal total cost. There are polynomial time algorithms for the CPP on totally directed or totally undirected graphs [4], but not for the mixed case. In Section 3 we show that the mixed graph CPP is NP-complete [9], and consequently it is very unlikely that efficient algorithms for its exact solution exist.

The fact that a problem is NP-complete is usually a good reason to stop all efforts for solving the problem in its full generality. Nevertheless, in practical situations we need to solve the CPP on special classes of graphs (e.g. planar, with small node degrees and equal lengths of edges). It could be the case that these restrictions of the CPP are not NP-complete, and hence there is more hope for the existence of an efficient algorithm. In Section 4 it is shown that the CPP remains NP-complete even

This work was supported by the National Science Foundation under Grant GK-42048 and the US Army Research Office-Durham under Grant DAHC04-75-0192

Author's address Department of Electrical Engineering, Princeton University, Brackett Hall, Engineering Quadrangle, Princeton, NJ 08540

if such restrictions are imposed on its domain.

Finally, in Section 5 we give an efficient algorithm for a related but apparently simpler problem, namely the problem of finding the minimum Eulerian mixed graph containing a given mixed graph. The algorithm is essentially a reduction of this problem to the max-flow problem in networks with unit edge capacities [15] We also show that the reduction in the opposite direction is possible, thus establishing that this problem is not easier than the max-flow problem

2. *Definitions*

The set of nonnegative integers is denoted by Z^+. All other sets mentioned in this paper are finite. A set is maximal with respect to some property if it has this property and is not properly included in any set having this property. A set is maximum w.r.t. (with respect to) some property if it is as large as any set having this property. The cardinality of a set S is denoted by $|S|$. A multiset S' defined on S is a function mapping S to $Z^+ - \{0\}$ Multisets can be thought of as sets with repetitions permitted.

A digraph is a pair $D = (V, A)$, where V is a set of nodes and A, a subset of $V \times V$, is a set of directed edges. A graph is a pair $G = (V, E)$, where E is a set of edges, that is, subsets of V of cardinality 2. A mixed graph is a triple $M = (V, E, A)$ such that (V, E) is a graph and (V, A) is a digraph

The degree of a node v, $d(v)$, is the number of edges incident upon v. The indegree (respectively outdegree) of v, $in(v)$ (respectively $out(v)$), is the number of directed edges entering v (respectively leaving v). If $M = (V, E, A)$ is a mixed graph, an orientation e of M is an assignment of directions to E. $M(e)$ is the resulting digraph.

A path in a mixed graph is a sequence of (not necessarily distinct) directed edges and edges $[(v_1, v_2), (v_2, v_3), (v_3, v_4), \cdots, (v_{n-1}, v_n)]$. A path is closed if $v_n = v_1$. An Eulerian path is a closed path containing all directed edges and edges exactly once. A mixed graph is called Eulerian (or unicursal) if it has an Eulerian path.

Two nodes u and v of a mixed graph are connected if there are paths from u to v and from v to u. A connected component of a mixed graph is a maximal set of pairwise connected nodes. A mixed graph is connected if it has exactly one connected component.

A route in a mixed graph is a closed path containing all edges and undirected edges at least once. The cost of a route will be the sum of the costs of all edges and directed edges in the route (with multiple occurrences multiply counted) minus the sum of the costs of all edges and directed edges of the graph. That is, the cost of a route is the sum of the costs of all extra copies of directed and undirected edges contained in the route.

A generalized graph (or digraph, or mixed graph) is one whose edge set is a multiset. Degrees, paths, routes, and unicursality have analogous definitions for generalized graphs

The simple proofs of the following can be found in [7]:

PROPOSITION 1. *A generalized graph is Eulerian iff it is connected and the degrees of all notes are even*

PROPOSITION 2. *A generalized digraph is Eulerian iff it is connected and the indegree of each node equals its outdegree.*

A bipartite graph is a triple $B = (Q, R, P)$, where Q and R are disjoint sets and $(Q \cup R, F)$ is a digraph and F is a subset of $Q \times R$. A matching in B is a node-disjoint subset of F.

The assignment problem is the following: Given a bipartite graph $(Q, R, Q \times R)$ and a cost function c defined on its edge set, find a maximal matching of B with minimum cost.

The Chinese Postman Problem (CPP) is the following: Given a mixed graph $M = (V, E, A)$ and a cost function defined on $E \cup A$, find a route of minimum cost. The language recognition version of the CPP is the following: Given a mixed graph M, a

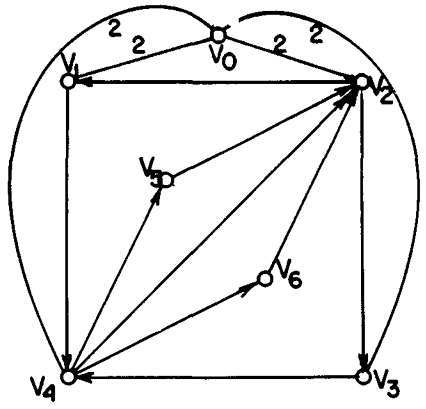

FIG 1

cost function c, and an integer k, does M have a route of cost k or less?

NP is the set of language recognition problems solvable by polynomial time bounded nondeterministic Turing machines. A problem is NP-complete if it is in NP and all problems in NP are reducible to it by some construction requiring only a polynomial amount of steps. The 3-Sat problem (the problem of recognizing whether a Boolean expression in disjunctive normal form with exactly three literals per clause is satisfiable) is known to be NP-complete [9].

3. *NP-Completeness of the Chinese Postman Problem*

The CPP appears to be particularly easy when the underlying graph is either totally undirected or totally directed. In both cases, in order to find the optimal route we first find the nodes failing to satisfy Propositions 1 and 2, respectively, and then connect those nodes in pairs by paths in an optimal way.

In particular, if the graph is totally directed we find the distance, $dist(u, v)$, between each pair of nodes u and v, that is, the length of the shortest path from u to v. The deficiency of a node v is defined as $def(v) = in(v) - out(v)$. We then set up the following assignment problem:

$$B = (Q, R, Q \times R),$$

where

$$Q = \bigcup_{def(v) > 0} \{v^1, v^2, \cdots, v^{def(v)}\},$$

$$R = \bigcup_{def(v) < 0} \{v^1, v^2, \cdots, v^{|def(v)|}\}, \qquad c(u^i, v^j) = dist(u, v).$$

(That is, for each node v with $def(v) > 0$ we have, in Q, $def(v)$ copies of v, and for each node v with $def(v) < 0$ we have, in R, $|def(v)|$ copies of v.)

The optimal route is then easily obtained from the optimal assignment. All the above steps can be carried out in $O(|V|^5)$ time by using the algorithms in [1] and [10]. In the totally undirected case [4], the problem can be reduced to the maximum weighted matching problem for complete graphs, and hence it can be solved in $O(|V|^3)$ steps. It is interesting to notice that there is a straightforward reduction in the opposite direction, and consequently the two problems are of the same complexity.

Yet the general problem appears to be considerably harder—possibly of complexity growing faster than any polynomial—as the following theorem suggests:

THEOREM 1. *The CPP is NP-complete*

For the proof of Theorem 1, the following lemma is needed:

LEMMA. *For the mixed graph of Figure* 1,[1] *the optimal route has cost* 2. *Moreover,*

[1] Unless otherwise shown in a figure, all edges have cost 1

in any optimal route either both edges (v_0, v_1), (v_0, v_3) enter v_0 and both edges (v_0, v_2), (v_0, v_4) leave v_0, or vice versa.

PROOF OF THE LEMMA. Obviously, for a route to be optimal, two of the edges (v_0, v_1), (v_0, v_2), (v_0, v_3), (v_0, v_4) must leave v_0 and the remaining two must enter v_0 (otherwise at least two extra copies of some of these four edges would have to be contained in the route, resulting to a suboptimal route). Hence there are six possible orientations of the undirected edges. The proof follows from a case-by-case analysis. The six feasible orientations are shown in Table I together with the resulting deficiencies of the nodes and the costs of the corresponding optimal routes (obtained by the algorithm mentioned in the beginning of this section) for each orientation. The straightforward but tedious verification is omitted. □

TABLE I

Edges entering v_0	(v_0, v_1) (v_0, v_2)	(v_0, v_4) (v_0, v_4)	(v_0, v_2) (v_0, v_1)	(v_0, v_3) (v_0, v_3)	(v_0, v_2) (v_0, v_1)	(v_0, v_2) (v_0, v_4)
v_1	−1	1	−1	1	−1	1
v_2	0	2	2	0	2	0
v_3	1	−1	1	−1	−1	1
v_4	0	−2	−2	0	0	−2
Optimal cost	3	4	4	3	2	2

The mixed graph C of Figure 2(a) will be abbreviated henceforth by the schematic simplification shown in Figure 2(b). It follows directly from the lemma that, if a mixed graph M contains m copies of the graph C, then the optimal route of this graph has cost at least $2m$. This lower bound can be achieved if, in all m copies of C, either both edges (u_1, v_2), (u_1', v_4) enter C and both (u_2, v_1), (u_2', v_3) leave C, or vice versa, and no other edge of M having nonzero cost is traversed more than once by the optimal route. Intuitively, C will be used as a "comparator," forcing pairs of undirected edges to agree in their orientation in any optimal route.

We now define the fanout graph of capacity $n > 0$, $F(n)$, as the mixed graph shown in Figure 3. It is obvious that if $F(n)$ is a part of a graph M and $F(n)$ contributes only $2(n - 1)$ to the cost of the optimal route of M, then the presence of the $n - 1$ copies of the graph C will force the edges (s_1, t_1), (s_2, t_2), $\cdots$, (s_n, t_n) to either all enter $F(n)$ or all leave $F(n)$; the reverse holds for the edges (f, r_1), (f, r_2), $\cdots$, (f, r_n).

PROOF OF THEOREM 1: By reducing the 3-Sat problem to the CPP. Let D be an instance of the 3-Sat problem with variables, $x_1, x_2, \cdots, x_n$ and clauses $L_1, L_2, \cdots, L_m$. Let c_j be the total number of occurrences of x_j in D, and d_j the total number of occurrences of $\neg x_j$ in D. Let $g_j = \max(c_j, d_j) > 0$ and $k = 2 \sum_{j=1}^{n} (g_j - 1)$. We will construct a mixed graph M and a cost function over its edge set such that M has a route of cost k or less iff D is satisfiable. We first create a node v, and for each clause L_i of D we create two nodes u_i and u_i' and the directed edges (v, u_i), (v, u_i'), and (u_i', u_i) each with cost 1, and the directed edge (u_i, v) with cost 0.

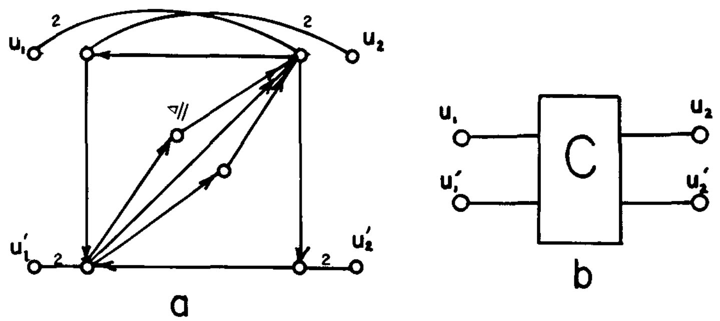

FIG 2

For each variable x_j of D, we create a graph $F(g_j)$. We let a node r_i coincide with u_p for each occurrence of $\neg x_j$ in the clause L_p, and a node t_j coincide with u_p for each occurrence of x_j in u_p. We let the remaining nodes from $\{r_m\}$ and $\{t_m\}$ coincide with v. This completes the construction of the graph M. (An illustration for the formula $D = (x_1 + \neg x_1 + \neg x_2)(x_1 + x_2 + \neg x_2)$ is shown in Figure 4.)

We claim that M has a route of cost at most k iff D is satisfiable. Suppose that D is satisfied by some truth assignment T. If $T(x_j) = 1$, we direct the edges (s_i, t_i), $i = 1, \cdots, g_j$, to leave the copy of $F(g_j)$ corresponding to x_j; otherwise, these edges are directed toward $F(g_j)$. Since D is satisfied by T, this will cause at least one of the three undirected edges incident upon u_i to enter u_i, for each clause L_i of D. Hence, in the resulting orientation of the undirected edges of M, all nodes u_i are guaranteed to have nonnegative deficiencies (possibly positive deficiencies can be satisfied at no cost by the edge (u_i, v)). Consequently the only cost of the optimal route of M will be the cost caused by the $k/2$ copies of C, which by the lemma is k.

Conversely, suppose that M has a route of cost at most k. Since a cost of k is inevitable because of the presence of the $k/2$ copies of C, it follows that in the orientation of M corresponding to the optimal route all nodes u_i have nonnegative deficiencies. This means that at least one undirected edge will be oriented toward u_i, for all clauses L_i of D. Consequently the orientation of the undirected edges in the copies of $F(n)$ corresponding to the variables induces a truth assignment satisfying D. This completes the proof of Theorem 1. (The straightforward verification of the facts that the CPP is in NP and that the reduction of this proof can be done in a polynomial number of steps has been omitted). □

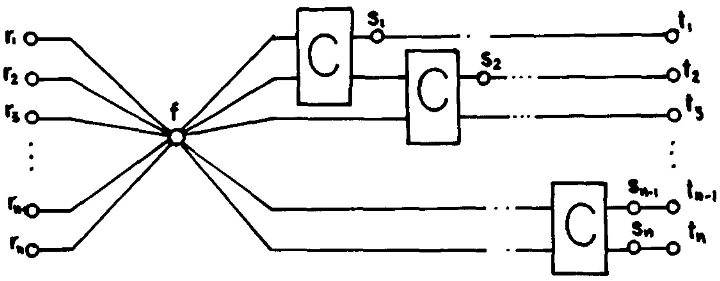

FIG. 3

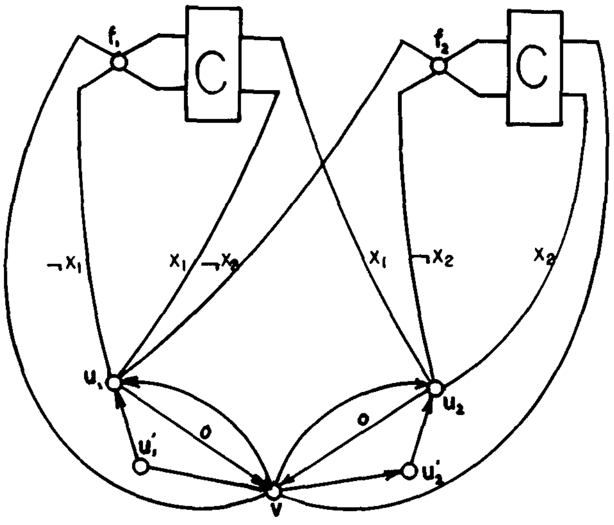

FIG 4

4. *NP-Completeness of a Practical Special Case*

It would be interesting to determine whether some restrictions on the domain of the CPP yield an easier problem, since the CPPs arising in the applications are subject to some "physical" constraints For example, in vehicle scheduling problems the underlying graphs are, naturally enough, planar, whereas in problems related to trash collection the degrees of the nodes very rarely exceed 4. The next theorem states that the CPP remains NP-complete even if such substantial—and "natural"—restrictions are imposed on the underlying graph and the cost function.

First we note that the edges of the graph M constructed in the proof of Theorem 1 have costs either 0 or 1 (edges of cost 2 can be easily replaced by two edges of cost 1). This is already one serious restriction with which the CPP remains NP-complete.

We are now going to prove that the CPP remains NP-complete even if the graph is planar. For this purpose we will remove, one by one, the reasons that cause the nonplanarity of M. The "comparator" graph of Figure 2(a), whose use was essential in the proof of Theorem 1, is itself nonplanar. In Figure 5 we show a planar version of C having precisely the same optimal route properties stated in the lemma; the fact that all costs are essentially multiplied by 2 will cause us to replace k by $2k$ in the construction of M for the proof of Theorem 1. It is the graph in Figure 5 that will be abbreviated henceforth by the simplification of Figure 2(b).

Now suppose that we embed the graph M in the plane. It is easy to see that this can be arranged, so that the edges joining the graphs $F(n)$ with v do not intersect with any other edge. Thus we simply have intersections of pairs of undirected edges joining copies of $F(n)$ with the vertices u_i. In order to remove these intersections, we first change the construction of M as follows: For each variable x_j we now have a copy of $F(2g_j)$ (instead of $F(g_j)$) and instead of one undirected edge joining this graph with u_i for each occurrence of x_j (or $\neg x_j$) in the clause L_i, we now have two undirected edges. We also add, for each node u_i, a third node u_i'' and the directed edges (v, u_i''), (u_i'', u_i) with cost 1, and we increase k from $4\sum(g_j - 1)$ to $4\sum(2g_j - 1)$. No further modifications are needed for the proof of the Theorem 1 to remain valid. In the embedding of M in the plane we let such pairs of undirected edges traverse the distance from $F(n)$ to u_i in parallel. Thus we now have intersections of the form of Figure 6(a) only, in which the edges (a, a') and (b, b') have the same orientation in any optimal route (presumably due to the presence of a copy of the graph C whose endpoints u_1 and u_1'—or u_2 and u_2'—coincide with a and b). It can be easily verified that the subgraph shown in Figure 6(b) can be used to replace any such intersection. k must again be increased by 4 for each such replacement. Since the number of possible intersections in a planar embedding of any graph, given that no two edges intersect twice (a condition easy to guarantee), is bounded by the square of the total number of edges in the graph, the construction described above is of polynomial time complexity.

We now define the total degree of a node v of a mixed graph to be $td(v) = d(v) + in(v) + out(v)$. We will show that the CPP remains NP-complete even if, besides the above restrictions, the total degree of each node is at most 3.

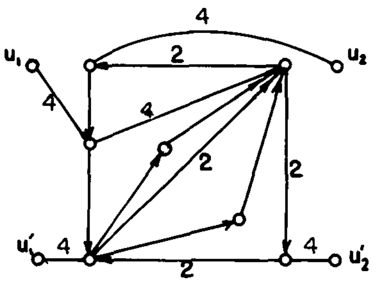

FIG 5

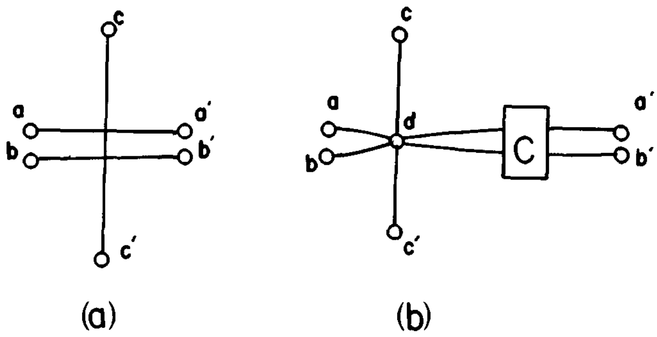

FIG 6

Suppose that a node v of the mixed graph $M = (V, E, A)$ has total degree greater than 3. We choose any two edges incident upon v such that they belong to the same face of the planar embedding of M, and we make them incident, instead of upon v, upon a new mode v'. We also join v and v' with an undirected edge of cost 0. Since the total degree of v has been decreased by 1 and no new nodes with degree 4 or more have been added to the graph, we need to repeat the same construction at most $\sum_{v \in V} td(v) = 2(|E| + |A|)$ times until no node of degree more than 3 exists. The new graph has the same cost of optimal route as the previous one, and its planarity remains unaffected.

We will now remove another unrealistic feature of M: the edges of cost 0. Suppose that our mixed graph is $M = (V, E, A)$ and a 0-1 cost function c is defined on its edges. Let $b = (|E| + |A|)^2$. We define a new cost function c' as follows: For each $e \in A \cup E$ we have

$$c'(e) = \begin{cases} 1 & \text{if } c(e) = 0, \\ b & \text{if } c(e) = 1. \end{cases}$$

We claim that M has a route of cost at most k under cost function c iff it has one of cost at most $(k + 1)b - 1$ under cost function c'. Suppose that M with cost function c has a route of cost k or less. Then there is an optimal route with fewer than b occurrences of edges of cost 0. Consequently this same route has cost at most $(k + 1)b - 1$ under cost function c'. Conversely, suppose that M with cost function c' has a route of cost at most $(k + 1)b - 1$. Then this route contains at most k edges of cost b. Hence the same route will have cost at most k under cost function c. Furthermore we can now replace all edges in M having cost b with b edges with cost 1 forming a chain with $b - 1$ new nodes. The resulting graph has edges of cost uniformly 1.

We summarize the discussion above by the following theorem:

THEOREM 2. *The CPP is NP-complete even if the underlying graph is planar with the total degree of nodes at most* 3 *and the costs of all edges equal to* 1 □

5. *An Efficient Algorithm for a Related Problem*

Let $B = (Q, R, F)$ be a bipartite graph with a positive, integer valued capacity function cp defined on $Q \cup R$. A b-matching [15] of B is a subset T of F such that, for all $v \in Q \cup R$, there are at most $cp(v)$ edges in T incident upon v.

A network with unit capacities [6, 15], $N = (S, F, s, t)$, is a generalized digraph (S, F) with two distinguished nodes s, t, such that $in(s) = out(t) = 0$. A flow in N is a subset f of F such that for each v in $S - \{s, t\}$ $|\{(u, v) \in f\}| = |\{(v, u) \in f\}|$. The value v of a flow is $v(f) = |\{(s, u) \in f\}|$. By using an algorithm by Dinic [2] as modified by Tarjan [15], we can find the maximum flow in N in $O(\min(|S|^{2/3} |F|, |F|^{3/2}))$ time. In this algorithm we start with $f = \varnothing$ and we apply the following iterative step: We first stratify S by $S_0 = \{s\}$, $S_{i+1} = \{v \in S : (u, v) \in F - f \text{ or } (v, u) \in f, u \in S_i, v \notin S_0,$

$\cdots, S_i\}$, $i = 0, \cdots, k-1$, so that $t \in S_k$. We call G_k the vertex induced subgraph of G containing the nodes in $S_1, \cdots, S_k$. We then add to f a maximal set of edge disjoint $s - t$ paths of G_k. We also update G_k by deleting vertices having 0 indegree or outdegree. Each such iteration is called a stage, and can be carried out in $O(|F|)$ time by the use of breadth-first search Note that k is increased after each stage. The algorithm terminates if t is not reachable from s in $(S, F - f)$. The upper bound follows from the fact that there are at most $O(\min(|S|^{2/3}, |F|^{1/2}))$ stages in the algorithm For a more detailed description of the algorithm see [15].

The Minimum Eulerian Graph (MEG) problem is the following: Given a mixed graph $M = (V, E, A)$, find a multiset D of directed edges having minimal cardinality, such that $M' - (V, E, A \cup D)$ is Eulerian We describe for this problem an $O(|A| + \min(|E|^{3/2}, |E||V|^{2/3}))$ algorithm. The algorithm will also provide us with an orientation e of E such that $M'(e)$ is Eulerian. Naturally the algorithm can be used as a unicursality test for mixed graphs. As such, it is an improvement over the test proposed by [6]. Moreover, it can be verified that the assumption that M is a proper (i.e. not generalized) mixed graph is a technical one, and that the assertions about the correctness and time requirements of the algorithm remain valid even if A is a multiset.

The input to the algorithm is a representation of M (Figure 7(a)). D will be given in the output in terms of an array $count(v)$, $v \in V$, denoting the number of directed edges in D that are leaving (respectively entering, when $count(v)$ is negative) v. Also, the optimal orientation e can be given as an array of $|E|$ nodes.

In the algorithm described below (steps 1 and 2), it is assumed that all nodes have even total degree. We then show that a modification of the algorithm works for the general case.

ALGORITHM A

Step 1. We first define the graph $G = (V, E)$ and the labeling function l on V by $l(v) = in(v) - out(v)$. Note that the array $l(v)$ is all the information we need to keep from

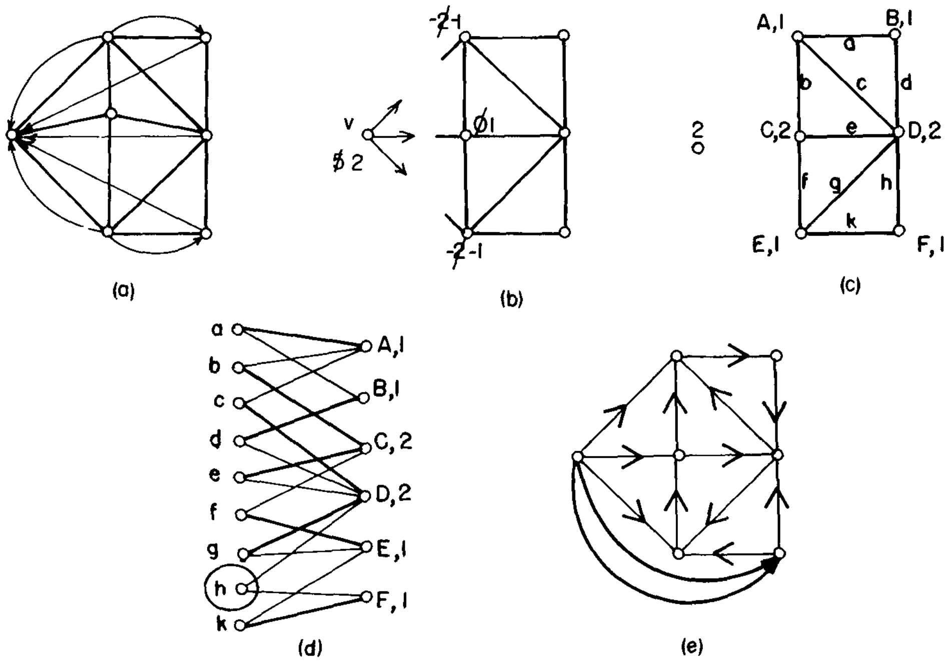

FIG 7

A. If for some $v \in V$, $d(v) \leq l(v)$ (Figure 7(b)), directing all undirected edges incident upon v to leave v is not suboptimal. This can be seen as follows: If in an optimal orientation e some edge (u, v) is directed toward v, then $count(v) \geq l(v) - d(v) + 2 \geq 2$. If $count(u) < 0$, then e is suboptimal, since we can reverse the direction of (u, v) in e, thus decreasing by two the cardinality of D. If $count(u) \geq 0$, we can find another optimal orientation e' such that (u, v) is directed toward u, by reversing the direction of (u, v), decreasing $count(v)$ by 2, and increasing $count(u)$ by 2.

Thus for all v such that $d(v) < l(v)$, we delete all undirected edges incident upon v, we replace $l(v)$ by $l(v) - d(v)$, we increase $l(u)$ by 1 for all u adjacent to v in G, and we update e. Similarly, if $d(v) \leq -l(v)$, we delete all undirected edges incident upon v, we replace $l(v)$ by $l(v) + d(v)$, we decrease by 1 $l(u)$ for all u adjacent to v, and we update e. We terminate this step if no more such v can be found. At this point, $|l(v)| < d(v)$ for all v that are not isolated vertices in G.

Step 2. This step is applied to each connected component $\bar{G} = (\bar{V}, \bar{E})$ of G separately. If $\bar{V}$ has only one node v, we set $count(v) = l(v)$. Otherwise, for each node $v \in \bar{V}$ we set $m(v) = (l(v) + d(v))/2$. Since $l(v) + d(v) = td(v) - 2\,out(v)$ ($td(v)$ is by hypothesis even) and $|l(v)| < d(v)$, it follows that $m(v)$ is an integer and $0 < m(v) < d(v)$ (Figure 7 (c)). The significance of $m(v)$ is that, for each $v \in \bar{V}$, exactly $m(v)$ out of $d(v)$ edges incident upon v must be oriented away from v if v is going to have 0 count. Hence the optimal orientation for this component can be obtained from the maximum b-matching of $B = (\bar{E}, \bar{V}, \bar{F})$, where $F = \{(d, v), (d, u) : d = (v, u) \in \bar{E}\}$ and the capacities of the nodes are $cp(d) = 1$ for all $d \in \bar{E}$ and $cp(v) = m(v)$ for all $v \in \bar{V}$ (Figure 7 (d)). We will investigate th complexity of this step (as well as of the whole algorithm) later. For the moment we note that the maximum b-matching can be obtained as the maximum flow in the network (with unit edge capacities) $N = (S, F', s, t)$, where $S = \bar{E} \cup \bar{V} \cup \{s, t\}$ and F' contains, besides F, all edges (s, d) for $d \in \bar{E}$ and $m(v)$ copies of (v, t), $v \in \bar{V}$. The maximum b-matching T corresponds in a natural way to the optimal orientation of the edges. If for some edge $d = (u, v)$ neither (d, u) nor (d, v) is in T, then we orient it, say, from u to v and decrease the count of u by 2. On the other hand, if for some $k > 0$ only $m(v) - k$ edges of T are incident upon v, we increase the count of v by k.

This completes the description of the algorithm. (The optimal D and e for the mixed graph of Figure 7(a) are shown in Figure 7(e)). □

It turns out that in the analysis of the algorithm described above the solution of the max-flow problem is the most time consuming step. This problem can be solved fairly efficiently by the Dinic-Tarjan algorithm outlined above, and an improvement on the time requirements of this algorithm can be obtained by exploiting the special structure of the problem. The following two lemmas essentially show that the arguments in [15] are also applicable here, despite the fact that the number of nodes of N (step 3) is related to $|\bar{E}|$ rather than $|\bar{V}|$.

LEMMA 1. *If in a subgraph $\bar{N}$ of the network N (defined in the description of step 2) there are n edge-disjoint $s - t$ paths, then the $s - t$ distance is at most* $4|\bar{V}|n^{-1/2} + 1$.

PROOF We construct G_k (see the description of the Dinic-Tarjan algorithm in the beginning of this section) from the given subgraph. For i even and nonzero, S_i will contain nodes of $\bar{G}$; hence the n edge-disjoint paths in $\bar{N}$ correspond in a natural way to n edge-disjoint paths in $\bar{G}$. Since we must have $|S_{2i}|\,|S_{2i+2}| \geq n$ for $i = 1, \cdots, (k+1)/2$, it follows that either S_{2i} or S_{2i+2} has more than $n^{1/2}$ elements, and since $\sum_i |S_{2i}| \leq |\bar{V}|$ the inequality follows. □

LEMMA 2. *The Dinic-Tarjan algorithm applied on N will terminate after at most* $O(|\bar{V}|^{2/3})$ stages.

PROOF Suppose that after $|\bar{V}|^{2/3}$ stages we have a flow f with value n, whereas the optimal flow value is $n + p$. Then in $N' = (S, F' - f)$ we have p edge-disjoint $s - t$

paths, and, by Lemma 1, the $s - t$ distance is at most $4|\bar{V}|p^{1/2} + 1$, although we know that it is at least $|\bar{V}|^{2/3}$. Hence $p \leq 25|\bar{V}|^{2/3}$ and there are at most $O(|\bar{V}|^{2/3})$ stages left. □

THEOREM 3. *Algorithm A correctly computes count(v) and e in* $O(|A| + \min(|E|^{3/2}, |E||V|^{2/3}))$ *time.*

PROOF The proof of the correctness of Algorithm A follows from the discussion above. We can find the connected components of G for step 1 in $O(|E|)$ time, by using the methods of [8]. The rest of step 1, if carefully implemented, requires $O(|E|)$ time since at most $|E|$ edges are deleted, each at most once.

For step 2, we note that the max-flow problem can be solved in $O(\min(|F'|^{3/2}, |F'||S|^{2/3}))$ time by the Dinic-Tarjan algorithm. Since $|F'| = 4|\bar{E}|$ and $|S| = |\bar{V}| + |\bar{E}| + 2$, we have an $O(|E|^{3/2})$ upper bound. In order to establish the $O(|E||V|^{2/3})$ bound, we note that each stage of the algorithm requires $O(|E|)$ time and, by Lemma 2, we have at most $O(|V|^{2/3})$ stages. □

Now suppose that the original mixed graph has some nodes of odd total degree. For such a node v, we let $m(v) = (1 + d(v))/2$ and, in the construction of the network N we create, instead of $m(v)$ copies of the edge (v, t), $m(v) - 1$ such copies together with a chain $(v, v_1), (v_1, v_2), \cdots, (v_{|V|}, t)$, where $v_1, \cdots, v_{|V|}$ are new nodes. Obviously the maximum flow obtained by the Dinic-Tarjan algorithm in such a network will contain the edges in the chain only if no other flow augmentation is possible. It is not hard to see that this condition guarantees optimality for the corresponding solution to the MEG problem, and also that the modification does not affect the asymptotic time requirements of the algorithm.

In the reduction of the MEG to the max-flow problem (step 2), the number of nodes of the resulting network N is $O(|E|)$. Nevertheless, due to the special structure of N, we have shown that the complexity of the MEG shares the same upper bound with the complexity of the max-flow problem. The next theorem implies that an asymptotic improvement on the upper bound $O(\min(|E|^{3/2}, |E||V|^{2/3}))$ for the MEG would result in a similar improvement for the max-flow problem.

THEOREM 4. *Given a network with unit capacities* $N = (S, F, t, s)$, *we can find in linear (w.r.t* $|S \cup F|$) *time a mixed graph* $M = (S, E, A)$ *with* $|E| \leq |F|$ *and* $|A| \leq 2|F|$ *such that the maximal flow for N can be obtained in linear time from the optimal solution (count, e) of the MEG induced by M.*

SKETCH OF PROOF. It is not hard to see that it is not a loss of generality to assume that $out(s) = in(t)$. We construct E and A as follows: We first add $2\,out(s)$ copies of the edge (t, s) in A. For each directed edge (u, v) of F, we add an undirected edge (u, v) in E, and we also add a directed edge (u, v) in A. Let us call a solution $(count, e)$ of the MEG of M proper if $count(v) = 0$ for all $v \in V - \{s, t\}$. We will show that, given any optimal solution $(count, e)$ of the MEG, we can find a proper optimal solution. Suppose that we are given an optimal solution $(count, e)$ of the MEG $M = (S, E, A)$ We form a digraph $G = (S, A')$ with A' a subset of A such that $(u, v) \in A'$ if $(u, v) \in A$ and the edge $\{u, v\} \in E$ is directed (in e) from u to v. Suppose that for some $u \in S - \{s, t\}$, $count(u) = 2k > 0$. Then we find k edge-disjoint paths from s to u in G, reverse the orientation e for edges in these paths, remove these edges from A', set $count(u) = 0$, and increase $count(s)$ by $2k$. Similarly, if $count(u) = -2k < 0$, we find $k\,u - t$ paths, remove these edges, and decrease $count(t)$ by $2k$. If the above steps are carefully implemented (and the breadth-first searches start from u), all vertices with $count \neq 0$ can be taken care of in linear time, since we delete at most $|A'|$ edges. Moreover, an optimal proper solution $(count, e)$ corresponds to a maximal flow f, where $(u, v) \in f$ iff $(u, v) \in F$ and the undirected edge $\{u, v\} \in E$ is oriented in e from u to v. □

ACKNOWLEDGMENTS I wish to thank Professor Orloff for suggesting this problem to me I am grateful to Professor Ullman for persuading me on the validity of (and also suggesting to me the line attack to) Theorem 1. Also many thanks are due to Professor Steiglitz for the careful reading of the manuscript and many helpful remarks, as well as to an anonymous referee for some important suggestions.

REFERENCES

1 AHO, A V , HOPCROFT, J E , AND ULLMAN, J.D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass , 1974
2 DINIC, E A Algorithm for solution of a problem of maximum flow in a network with power estimation *Soviet Math Dokl. 11,* 5 (1970), 1277-1280
3 EDMONDS, J The Chinese Postman Problem *Oper Res 13,* Suppl 1 (1965), B73-B77
4 EDMONDS, J , AND JOHNSON, E L Matching, Euler tours and the Chinese Postman *Math Programming 5,* 1 (1973), 88-124
5 EULER, L Solutio problematis ad geometriam situs pertinentis *Commentarii Academiae Petropolitanae 8* (1736), 128-140
6 FORD JR , L R , AND FULKERSON, D R *Flows in Networks* Princeton U Press, Princeton, N J., 1962
7 HARARY, F *Graph Theory* Addison-Wesley, Reading, Mass , 1970
8 HOPCROFT, J E , AND TARJAN, R E Algorithm 447 Efficient algorithms for graph manipulation *Comm ACM 16,* 6 (1973), 372, 378
9 KARP, R M Reducibility among combinatorial problems In *Complexity of Computer Computations,* R E Miller and J W Thatcher, Eds , Plenum Press, New York, 1972, pp 85-103.
10 KUHN, H W The Hungarian method for the assignment problem *Naval Res Logistics Quart 2* (1955), 83-97
11 MEI-KO, K Graphic programming using odd or even points *Chinese Math 1,* (1962), 237-277
12 PAPADIMITRIOU, C H The Euclidean TSP is NP-complete TR #191, Dep of Elec Eng , Princeton U , Princeton, N J , 1975
13 PAPADIMITRIOU, C H , AND STEIGLITZ, K On the complexity of local search for the Traveling Salesman Problem TR #189, Dep of Elec Eng , Princeton U , Princeton, N J , 1975
14 SAHNI, S , AND GONZALES, T P-complete problems and approximate solutions Proc 15th Annual Symp on Switching and Automata Theory, 1974, pp 28-32 (available from IEEE, New York)
15 TARJAN, R E Testing graph connectivity Proc 6th Sigact Symp on the Theory of Computing, 1974, pp 185-193 (available from ACM, New York)

RECEIVED NOVEMBER 1975, REVISED MARCH 1976