ARC ROUTING PROBLEMS, PART I: THE CHINESE POSTMAN PROBLEM

H. A. EISELT
University of New Brunswick, Fredericton, New Brunswick, Canada

MICHEL GENDREAU and GILBERT LAPORTE
Centre de recherche sur les transports, Université de Montréal, Montréal, Québec, Canada

(Received January 1994; revision received March 1994; accepted June 1994)

Arc routing problems arise in several areas of distribution management and have long been the object of study by mathematicians and operations researchers. In the first of a two-part survey, the Chinese postman problem (**CPP**) is considered. The main algorithmic results for the **CPP** are reviewed in five main sections: the undirected **CPP**, the directed **CPP**, the windy postman problem, the mixed **CPP**, and the hierarchical **CPP**.

In arc routing problems (**ARPs**), the aim is to determine a least-cost traversal of a specified arc subset of a graph, with or without constraints. Such problems occur in a variety of practical contexts and have long been the object of attention by mathematicians and operations researchers. Perhaps the earliest documented reference to **ARPs** is the famous Königsberg bridge problem. The question is to determine whether there exists a closed walk traversing *exactly once* each of seven bridges on the Pregel river in Königsberg, now called Kaliningrad (Figure 1). The problem was solved by the Swiss mathematician Leonhard Euler who found conditions for the existence of a closed walk and showed there was none in this particular case (Euler 1736). Euler was concerned almost exclusively with the existence of a closed walk. The question of determining such a walk was addressed and solved more than one century later by Hierholzer (1873). Interesting accounts of this problem, including facsimiles of Euler's manuscripts can be found in Euler (1953) and in Korte (1989). English translations of the original Euler and Hierholzer articles are provided in Fleischner (1990).

Another well-known closely related problem is the so-called Chinese postman problem posed by Meigu Guan (or Kwan Mei-Ko), a mathematician at the Shangtun Normal College who spent some time as a post office worker during the Chinese cultural revolution (Korte). In contrast to the Königsberg bridge problem, which is only concerned with the existence and determination of a closed walk, here the question is to deal with situations where such a solution does not exist. In this case, a natural question is to determine a minimum length walk covering each segment *at least once*. The problem is simply stated by Guan (1962): "A mailman has to cover his assigned segment before returning to the post office. The problem is to find the shortest walking distance for the mailman."

Arc routing problems arise in a number of practical contexts, such as mail delivery, garbage collection, snow removal, and school bus routing. Billions of dollars are spent each year by governments and private enterprise on these operations. Important sums of money are also wasted because of poor planning. Operations researchers have for a long time studied the structure of these problems and proposed workable solutions, sometimes yielding sizeable savings. Some of the most important work has been summarized in early surveys (Benavent et al. 1983, Bodin et al. 1983, Guan 1984a), in some books (Busacker and Saaty 1965, Kaufmann 1967, Harary 1969, Liebling 1970, Christofides 1975, Larson and Odoni 1981, Lawler et al. 1985, Fleischner 1990, 1991, Evans and Minieka 1992), and in Win's thesis (1987). However, this literature is quite scattered and disorganized. The objective of this two-part survey is to present an integrated and updated overview of the most relevant operations research literature on arc routing. In addition to providing an extensive bibliographic coverage, we will describe the relationships between the various problems, comment on their relative difficulty, outline some of the best algorithms, and describe applications.

It is convenient to define **ARPs** as a special case of the class of general routing problems (**GRPs**) studied by Orloff (1974) and by Male, Liebman and Orloff (1977).

Subject classification: Transportation, vehicle routing: Chinese postman problem.
Area of review: SURVEY, EXPOSITORY & TUTORIAL.

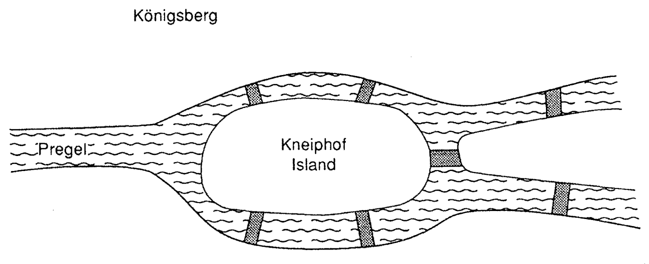

Figure 1. The seven bridges of Königsberg.

Let $G = (V, A)$ be a connected graph without loops, where $V = \{v_1, \ldots, v_n\}$ is the *vertex set* (or *node set*), and $A = \{(v_i, v_j): v_i, v_j \in V \text{ and } i \neq j\}$ is the *arc set*. (In some problems, G is a *multigraph*, i.e., some arcs (v_i, v_j) may be replicated.) With every arc (v_i, v_j) is associated a nonnegative *cost*, *distance* or *length* c_{ij}; assume that $c_{ij} = \infty$ if (v_i, v_j) is not defined. The matrix $C = (c_{ij})$ is symmetric if and only if $c_{ij} = c_{ji}$ for all i, j. When C is symmetric, it is common to associate an *edge* (or undirected arc) with every vertex pair. Hence, depending on whether A is a set of arcs or edges, the associated **ARP** will be termed *directed* or *undirected*. Some graphs that include directed and undirected arcs are called *mixed* graphs. The cost matrix satisfies the triangle inequality if and only if $c_{ik} + c_{kj} \leq c_{ij}$ for all i, j, k. In **GRP**s, one seeks a minimum cost circuit that includes a subset $Q \subseteq V$ of *required vertices* and a subset $R \subseteq A$ of *required arcs*, but may involve other vertices and arcs if necessary. Two important **ARP**s can be derived from **GRP**s (see Lenstra and Rinnooy Kan 1976): 1) the rural postman problem (**RPP**) for $Q = \varnothing$ and some prespecified R; 2) the Chinese postman problem (**CPP**) for $Q = \varnothing$ and $R = A$. The **RPP** is commonly associated with mail delivery in rural areas: There are a number of villages whose set R of streets has to be serviced by a postman, and a set $A \backslash R$ of links between villages that do not have to be served, but may be used for traveling between villages. In contrast, the **CPP** is more likely to arise in urban settings. Most arc routing applications are of the **RPP** type. The first part of this survey of **ARP**s is devoted to the **CPP**, while the second part (to appear in the next issue of this journal) is about the **RPP**.

The property of *unicursality* is central to the study of the **CPP**. A connected graph is said to be *unicursal* or *Eulerian* if there exists a closed walk in G containing each arc exactly once and each vertex at least once. A fundamental result is the statement of necessary and sufficient conditions for a connected graph to be unicursal (Ford and Fulkerson 1962, pp. 70–71):

1. If G is *undirected*, every vertex must have an even degree, i.e., an even number of incident edges. This condition was first proved by Euler (1736).
2. If G is *directed*, the number of arcs entering and leaving each vertex must be equal.
3. If G is *mixed*, every vertex must be incident to an even number of directed and undirected arcs; moreover, for every $S \subseteq V$, the difference between the number of directed arcs from S to $V \backslash S$ and the number of directed arcs from $V \backslash S$ to S must be less than or equal to the number of undirected arcs joining S and $V \backslash S$. This condition is sometimes called the "balanced set condition" (Nobert and Picard 1991).

Algorithms for the **CPP** contain two distinct stages. The first is to determine a minimum cost augmentation of the graph, i.e., a least-cost set of arcs or edges that will make the graph unicursal. Then an actual traversal of the augmented graph can always be determined in polynomial time. When G is completely undirected or completely directed, the augmentation problem is easy and well solved. In the first case, a unicursal graph can be derived from G in polynomial time by solving a matching problem (Edmonds and Johnson 1973); in the second case, the minimum cost augmentation is obtained by solving a minimum cost flow problem (Edmonds and Johnson). Other types of **CPP** are not so easy. If G is a mixed graph, the least-cost augmentation problem is NP-hard, even if G is planar or if all c_{ij}'s are equal to the same value (Papadimitriou 1976). A problem closely related to the mixed **CPP** is the windy postman problem (**WPP**) first introduced by Minieka (1979). Here, G is an undirected graph, although the cost of traversing an edge depends on the direction of travel. The **WPP** consists of determining a least-cost traversal of all edges of G. Brucker (1981) and Guan (1984b) have shown that the **WPP** is NP-hard, but the problem is solvable in polynomial time if G is Eulerian (Win 1989). Every mixed **CPP** can also be transformed into a **WPP**: If (v_i, v_j) is an edge of cost c, then the cost of traversing (v_i, v_j) in either direction is c; if (v_i, v_j) is an arc of cost c, then in the **WPP** the cost of traversing (v_i, v_j) is c if travel goes from v_i to v_j, and ∞ otherwise (Win 1989). Heuristic algorithms for the mixed **CPP** and the **WPP** consist of achieving a low-cost augmentation of the graph to satisfy the unicursality conditions. The best exact algorithms use an integer linear programming formulation of the problem and solve it by a branch-and-cut procedure. While this approach is relatively successful, it does not seem to perform as well as similar algorithms developed for the traveling salesman problem (**TSP**) (for example, see Padberg and Rinaldi 1991 or Grötschel and Holland 1991).

Another important **CPP** extension is the hierarchical **CPP** where a precedence relation is defined on A, and the order in which the arcs are serviced must respect this relation. As shown by Dror, Stern and Trudeau (1987), this problem is NP-hard but can sometimes be solved in polynomial time. This problem has received less attention than other types of **CPP**. A dynamic programming algorithm has been proposed for its solution.

Whereas the Fleischner books (1990, 1991) provide an exhaustive treatment of the **CPP** from a graph theoretical perspective, our treatment of the subject is more algorithmic. A number of interesting applications are described in Section 1. The remaining sections are devoted, respectively, to the following cases: the undirected **CPP**, the directed **CPP**, the **WPP**, the mixed **CPP**, and the hierarchical **CPP**. The conclusions follows in Section 7.

1. APPLICATIONS

In most arc routing applications, it is not necessary to service all arcs of the graph and these are therefore modeled as **RPPs**. In addition, several side constraints often come into play. However, the **CPP** often arises as a subproblem in these applications and this can be exploited in the design of algorithms. There are also problems that can be modeled as pure **CPPs**.

Miliotis, Laporte and Nobert (1981) consider the problem of determining a shortest complete cycle in G, i.e., a cycle passing through each vertex at *least once*. It is well known (see, e.g., Hardgrave and Nemhauser 1962) that this problem can immediately be transformed into a **TSP** with costs c'_{ij} equal to the length of a shortest path between v_i and v_j. However, an alternative approach is possible.

In the undirected case, define a restricted edge set A' by removing from A all edges (v_i, v_j) which are not themselves a shortest chain between v_i and v_j. Also define for each $(v_i, v_j) \in A'(i < j)$ an integer variable x_{ij} equal to the number of times (v_i, v_j) is used in the solution. Then the problem is as follows.

Problem UCC

$$\text{Minimize} \sum_{(v_i,v_j)\in A'} c_{ij}x_{ij} \qquad (1)$$

subject to

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} - 2y_k = 2 \quad (k = 1, \ldots, n) \qquad (2)$$

$$\sum_{\substack{v_i\in S, v_j\notin S \\ \text{or } v_i\notin S, v_j\in S}} x_{ij} \geq 2(S \subseteq V;\ 2 \leq |S| \leq n-2) \qquad (3)$$

$$x_{ij} \in \{0, 1, 2\} \quad (v_i, v_j) \in A') \qquad (4)$$

$$y_i \geq 0 \text{ and integer} \quad (v_i \in V). \qquad (5)$$

The solution to **UCC** defines a connected graph in which each vertex has an even degree; this graph is therefore *unicursal*. A shortest complete cycle on G is then easy to identify by solving a **CPP**. The directed case is similar. Computational results indicate that this approach may be preferable to transforming the problem into a **TSP**.

An important practical problem that can be modeled as a **CPP** is the topological testing of some computer systems at different levels (Malek, Mourad and Pandya 1989). These systems are described by graphs in which the vertices correspond to processors, switches or registers, depending on the specific level at which testing is to be conducted, and arcs represent data flow or data lines between vertices. An Eulerian circuit can then be used to send a testing packet traversing all arcs and vertices in minimal time.

Barahona (1990) describes several problems displaying a mathematical structure similar to that of the **CPP**. One of these problems is the determination of exact ground (minimal-energy) states for some planar spin glass systems (Barahona et al. 1982). A spin glass system can be modeled by an undirected graph in which each vertex corresponds to a spin and weighted edges indicate the interactions between spins (the weight of an edge represents the level of positive or negative interaction between the spins that it links). The ground-state problem consists of assigning $+1$ or -1 orientations to the spins with the objective of minimizing the total energy of the system, given by the opposite of the sum over all edges of the product of the spin orientations and the interaction weights. The optimal solution to this problem can be obtained by finding a minimum weight cut in the graph, partitioning up spins ($+1$ orientation) from down spins (-1 orientation). For planar spin glass systems, this minimum weight cut can be determined by solving an **ARP** on the planar dual graph of the graph describing the system. In this dual graph, the cost assigned to an edge is the absolute value of the interaction weight of the unique edge of the original graph crossed; the desired solution then corresponds to a minimum cost cycle cover of the dual edges associated with negative weight edges in the original graph.

A second problem described by Barahona is the via minimization problem arising in the design of VLSI circuits. In this problem, wires that connect the chip components must be assigned to one of the two layers of the chips. Wires are not allowed to cross one another on the same layer, but may go from one layer to the other through vias (connections between layers). The objective is to minimize the number of vias. To solve this problem, one first constructs a planar graph representation of the chip in which vertices correspond to wire end points and crossings and edges correspond to sections of wire between these. The solution is obtained by determining the smallest number of edges that must be broken into two pieces to make the graph bipartite (each occurrence corresponds to a via).

2. THE UNDIRECTED CHINESE POSTMAN PROBLEM

The first known result concerning the undirected **CPP** is due to Euler (1736) who showed that a connected undirected graph $G = (V, E)$ is unicursal if and only if all its vertices have even degree. Guan (1962) observed that G always has an even number of odd-degree vertices and that a unicursal graph G' can be derived from G by adding edges to link odd-degree vertices, i.e., some edges are replicated by introducing copies with the same cost. Guan proved that a necessary and sufficient condition for the optimality of a Eulerian tour on G' is that it has no redundancy, i.e., it does not contain more than two edges linking any vertex pair, and the length of added edges on every cycle does not exceed half the length of the cycle.

A natural way to formulate the undirected **CPP** as an integer linear program (**ILP**) is to seek a least-cost augmentation of G into G' such that all vertices of G' have an even degree. Define x_{ij}, the number of copies of $(v_i, v_j)(i < j)$ required to augment G. Let $\delta(i)$ be the set of edges incident to v_i, and let $T \subseteq V$ be the set of odd-degree vertices of V. Then the formulation is as follows.

Problem UCPP1

$$\text{Minimize} \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} \tag{6}$$

subject to

$$\sum_{(v_i,v_j)\in\delta(i)} x_{ij} \equiv \begin{cases} 1 \pmod 2 & \text{if } v_i \in T \\ 0 \pmod 2 & \text{if } v_i \in V\setminus T \end{cases} \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad ((v_i, v_j) \in A). \tag{8}$$

This model can be solved as a perfect matching problem on $K_{|T|} = (T, A_T)$, where $A_T = \{(v_i, v_j): v_i, v_j \in T, i < j\}$, and the cost of $(v_i, v_j) \in A_T$ is the cost of a shortest path between v_i and v_j on G (see, e.g., Edmonds 1965a, Busacker and Saaty 1965, Christofides 1973). Graph G' is then obtained from G by introducing the shortest paths corresponding to the optimal matching solution. Lawler (1976) provides an $O(|V|^3)$ time algorithm for the matching problem, but lower complexity algorithms can be derived by suitably exploiting data structures (see, e.g., Galil, Micali and Gabow 1986, Derigs and Metz 1991).

As Barahona notes (1990), this approach does not depend on the density of arcs in G. An equivalent formulation that exploits the sparsity of G has been proposed in a fundamental article by Edmonds and Johnson. For this, define for any nonempty proper subset of V the set $E(S) = \{(v_i, v_j): v_i \in S, v_j \in V\setminus S \text{ or } v_i \in V\setminus S, v_j \in S\}$. Then the problem is as given below.

Problem UCPP2

$$\text{Minimize} \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} \tag{9}$$

subject to

$$\sum_{(v_i,v_j)\in E(S)} x_{ij} \geqslant 1 \quad (S \subset V,\ S \text{ odd}) \tag{10}$$

$$x_{ij} \geqslant 0 \quad ((v_i, v_j) \in A) \tag{11}$$

$$x_{ij} \text{ integer} \quad ((v_{ij}, v_j) \in A). \tag{12}$$

In this formulation, constraints 10 are referred to as *blossom inequalities*. These are defined for every *odd set S*, i.e., for every proper subset of V containing an odd number of odd vertices. The polyhedron of solutions to (10) and (11) is equal to the convex hull of solutions to **UCPP2** (Edmonds and Johnson). Edmonds and Johnson solve **UCPP2** by means of an adaptation of Edmonds' blossom algorithm (1965b) for matching problems.

Once an Eulerian graph has been obtained, an Eulerian cycle must be determined on G'. Defining a *bridge* as an edge whose removal disconnects graph G, we can now describe Fleury's method as reported in Kaufmann (p. 309).

Fleury's Algorithm for Determining an Eulerian Cycle in an Undirected Eulerian Graph

STEP 1. Starting from an arbitrary vertex v_i traverse an edge (v_i, v_j) that is not a bridge and erase edge (v_i, v_j).

STEP 2. Set $v_i := v_j$ and repeat Step 1 starting from the other extremity of the deleted edge or stop if all edges have been deleted.

In spite of its apparent simplicity, this procedure may be time consuming because of the difficulty of determining at each step whether the edge considered for deletion is a bridge. To counter this, Edmonds and Johnson propose alternative methods of $O(|V|)$ time complexity. We describe their "end-pairing" algorithm.

End-Pairing Algorithm for Determining an Eulerian Cycle in an Undirected Eulerian Graph

STEP 1. Trace a simple tour that may not contain all vertices. If all edges have been included in the tour, stop.

STEP 2. Consider any vertex v on the tour incident to an edge not on the tour. Form a second tour from v not overlapping the first one.

STEP 3. Let e_1, e_2 be the two edges incident to v on the first tour, and let e_3, e_4 be the two edges incident to v on the second tour. Merge the two tours into a single tour: For example, starting at v with edge e_3, follow the second tour until it meets v by way of e_4, then continue on the first tour starting with e_2 until v is reached again by e_1. If all edges have been traversed, stop. Otherwise go to Step 2.

Additional algorithms for determining Eulerian cycles in graphs are described in Fleischner (1991).

A number of variants of the undirected **CPP** have been studied. Frederickson, Hecht and Kim (1978) consider the m-vehicle version of the problem where the objective is to minimize the length of the longest route. These authors show that this problem is NP-hard by reduction of the k-partition problem (Sahni and Gonzalez 1976). A heuristic with a worst-case performance ratio of $2 - 1/m$ is also provided. A problem closely related to the **CPP** is that of covering a graph by undirected cycles. The objective is to cover all edges of the graph by simple cycles so that the total length of the cover is minimized. This problem arises in the design of irrigation systems (Cross 1936) and in the analysis of electrical circuits (Itai and Rodeh

1978). Itai et al. (1981) state that the complexity of determining a minimal length cover in a graph is not known, but they conjecture that this problem is NP-hard. They also prove that minimizing cover length and solving the **CPP** are not equivalent. For example, they show that the Petersen graph (Figure 2) with unit edge lengths has an optimal **CPP** solution equal to 20 and a minimal cover solution of length 21. However, Kesel'man (1987) proved that in planar graphs the two problems are equivalent.

3. THE DIRECTED CHINESE POSTMAN PROBLEM

In the directed case, determining a minimum cost Eulerian graph from G can be achieved by solving a minimum cost flow problem where the flow on each arc has to be at least 1 (Orloff 1974). Note, however, that contrary to the undirected case, this problem may not have a solution even if G is connected. For a solution to exist, the graph must be strongly connected, i.e., there must be a directed path between every pair of nodes. Edmonds and Johnson (1973), Orloff (1974) and Beltrami and Bodin (1974) show how a least-cost Eulerian graph can be constructed by solving a transportation problem. Let I be the set of vertices v_i for which the number of incoming arcs exceeds the number of outgoing arcs by s_i and J, the set of vertices v_j for which the number of outgoing arcs exceeds the number of incoming arcs by d_j. Thus, s_i can be interpreted as a supply, and d_j as a demand. In addition, let c_{ij} be the length of a shortest path from v_i to v_j. The problem is then as follows.

Problem DCPP

$$\text{Minimize} \sum_{v_i \in I} \sum_{v_j \in J} c_{ij} x_{ij} \tag{13}$$

subject to

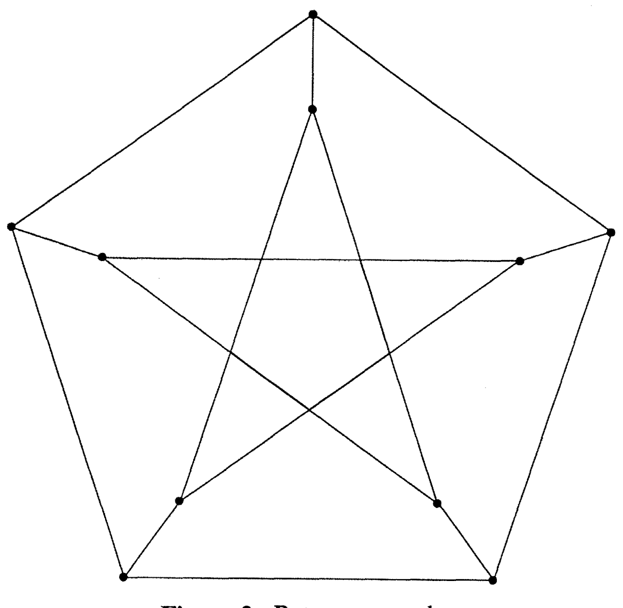

Figure 2. Petersen graph.

$$\sum_{v_j \in J} x_{ij} = s_i \quad (v_i \in I) \tag{14}$$

$$\sum_{v_i \in I} x_{ij} = d_j \quad (v_j \in J) \tag{15}$$

$$x_{ij} \geqslant 0 \quad (v_i \in I,\ v_j \in J). \tag{16}$$

The optimal x_{ij} values represent the number of extra times each arc has to be traversed. An equivalent procedure is also suggested by Lin and Zhao (1988). Once a unicursal graph has been determined, an Eulerian circuit can be drawn by simply adapting Fleury's algorithm for undirected graphs (Christofides 1975, p. 202). Alternatively, one can use the following procedure suggested by van Aardenne-Ehrenfest and de Bruijn (1951) and also described in Edmonds and Johnson.

The van Aardenne-Ehrenfest and de Bruijn Algorithm for Determining an Eulerian Circuit in a Directed Graph

STEP 1. Construct a spanning arborescence rooted at any vertex v_r.

STEP 2. Label all arcs as follows: Order and label the arcs outgoing from v_r in an arbitrary fashion; order and label the arcs out of any other vertex consecutively in an arbitrary fashion, so long as the last arc is the arc used in the arborescence.

STEP 3. Obtain an Euler tour by first following the lowest labeled arc emanating from an arbitrary vertex; whenever a vertex is entered, it is left through the arc not yet traversed having the lowest label. The procedure ends with an Euler circuit when all arcs have been covered.

As in the undirected case, there exist a number of covering problems associated with directed graphs, such as covering the graph with star trees, simple paths or circuits (see, e.g., Busacker and Saaty, and Christofides). An interesting problem consists of determining a minimum cost Eulerian subgraph in a given graph G. This problem was addressed by Richey, Parker and Rardin (1985) and Richey and Parker (1991) who have shown it to be NP-hard, even if G has a bounded vertex degree, but the problem can be solved in $O(|V|^3)$ time on so-called series-parallel graphs.

4. THE WINDY POSTMAN PROBLEM

As indicated in the Introduction, the **WPP** is NP-hard and contains the undirected, directed and mixed **CPPs** as special cases. In some cases, however, the problem can be solved in polynomial time. One sufficient condition, stated by Guan (1984b), is that the two orientations of every cycle of G should have the same length. Then, a shortest **WPP** tour on G is easy to determine as follows: Derive from G an undirected graph G' with edge costs $c'_{ij} = (c_{ij} + c_{ji})/2$, solve a **CPP**, and determine an Euler tour on G'. This tour is a shortest windy postman tour. This algorithm is polynomial, but requires checking whether all cycles have the same cost in both directions. Another sufficient condition for polynomial solvability is for G to be Eulerian. For this case, Win (1987, 1989) has proposed a polynomial-time algorithm based on a transformation of the Edmonds and Johnson algorithm for the mixed **CPP**. He also observed that the transformation of the **WPP** into a minimum cost integer flow problem with gains (Maurras 1972) suggested by Minieka (1979) does not yield a polynomial-time algorithm even for Eulerian graphs.

The polyhedral structure of the **WPP** has been studied by Win (1987, 1989) and by Grötschel and Win (1988, 1992). Let $G = (V, A)$ be the underlying graph and define $\delta(i)$ as the set of edges incident to vertex v_i and $E(S)$ as the set of edges (v_i, v_j) with $v_i \in S$, $v_j \notin S$, where $S \subset V$. Also let x_{ij} be an integer variable indicating how many times edge (v_i, v_j) is traversed from v_i to v_j in the optimal **WPP** solution. The problem is then as given next.

Problem WPP

$$\text{Minimize} \sum_{(v_i,v_j)\in A} (c_{ij}x_{ij} + c_{ji}x_{ji}) \tag{17}$$

subject to

$$x_{ij} + x_{ji} \geqslant 1 \quad ((v_i, v_j) \in A) \tag{18}$$

$$\sum_{(v_i,v_j)\in\delta(i)} (x_{ij} - x_{ji}) = 0 \quad (v_i \in V) \tag{19}$$

$$x_{ij}, x_{ji} \geqslant 0 \quad ((v_i, v_j) \in A) \tag{20}$$

$$x_{ij}, x_{ji} \text{ integer} \quad ((v_i, v_j) \in A). \tag{21}$$

Let P be the polyhedron of the vectors $x = (x_{ij}, x_{ji})$ satisfying (18), (19), and (20). Then G is Eulerian if and only if every vertex of P is integral; furthermore, the components of the vertices of P are always 0, 1/2, or a positive integer. Also, if G is Eulerian, then (18)–(20) give a complete description of the convex hull of the incidence vectors of the windy postman tours on G, denoted by **WP**(G) (Win 1987, 1989). Gendreau, Laporte and Zhao (1990) have shown that a previous result proven by Minieka (1979) for the mixed **CPP** applies to the **WPP**: In any optimal solution to **WPP**, one of the following three cases occurs for any edge (v_i, v_j): either $x_{ij} = 0$ and $x_{ji} \geqslant 1$, or $x_{ji} = 0$ and $x_{ij} \geqslant 1$, or $x_{ij} = x_{ji} = 1$. This suggests a natural three-way branching scheme. Grötschel and Win (1988) have shown that each of inequalities (18) and (20) defines a facet of **WP**(G) if and only if (v_i, v_j) is not a bridge of G. Furthermore for any odd $|E(S)|$, the following *odd cut inequalities* are valid with respect to **WP**(G), and define facet of **WP**(G) if and only if the subgraphs of G induced by S and $V\setminus S$ are connected:

$$\sum_{(v_i,v_j)\in E(S)} (x_{ij} + x_{ji}) \geqslant |E(S)| + 1 \quad (S \subset V) \tag{22}$$

$$\sum_{v_i\in S, v_j\notin S} x_{ij} \geqslant \tfrac{1}{2}(|E(S)| + 1) \quad (S \subset V) \tag{23}$$

and

$$\sum_{v_i\in S, v_j\notin S} x_{ji} \geqslant \tfrac{1}{2}(|E(S)| + 1) \quad (S \subset V). \tag{24}$$

Additional valid inequalities are also derived in Grötschel and Win (1988). Grötschel and Win (1992) observe that although the LP model defined by (17)–(20) and (22)–(24) contains an exponential number of constraints, it can be solved in polynomial time in the input length of the given **WPP**, by making use of the Padberg and Rao (1982) procedure to satisfy the odd-cut constraints. Grötschel and Win have devised a cutting plane algorithm to solve the LP, and have applied it to the solution of **WPP** instances ranging from 52 to 264 vertices, and having between 78 and 489 edges. The LP solution provided an optimal **WPP** solution for 31 problems out of 36. When the above LP fails to provide an integer solution, feasible **WPP** tours can be derived by appropriately rounding up the fractional variables, and then possibly setting some variables to zero (Grötschel and Win 1992). Similar strategies have been proposed by Win (1987, 1989) and by Gendreau, Laporte and Zhao (1990), starting from the solution of the LP defined by (17)–(20).

A problem closely related to the **WPP** is the *minimum cost Eulerian orientation problem* (**MCEOP**). Here $G = (V, A)$ is an undirected Eulerian graph and there are two costs c_{ij} and c_{ji} associated with each edge (v_i, v_j). An orientation of G is obtained by replacing edge (v_i, v_j) either by arc (v_i, v_j), or by arc (v_i, v_j). The **MCEOP** consists of determining the minimum cost Eulerian orientation of G. This problem differs from the **WPP** in that each edge may be traversed only once in the optimal solution. Guan and Pulleyblank (1985) proposed an $O(|V||A|^2)$ time algorithm for this problem. These authors also show how a class of sparse and Hamiltonian graphs can be derived from the set of **MCEOP** solutions. Win (1987, 1989) provides a complete description of the Eulerian orientation polytope of G. He also shows how some results developed for the **MCEOP** can be used to solve a restriction of the **WPP** in which each edge (v_i, v_j) can be traversed at most d_{ij} times (this problem reduces to the **MCEOP** if $d_{ij} = 1$ for all (v_i, v_j)). In particular, if all d_{ij}'s are odd, the problem can be solved in polynomial time and there exists a complete description of the polytope of feasible solutions containing $O(|A|^2)$ constraints.

5. THE MIXED CHINESE POSTMAN PROBLEM

For notational convenience, we will assume in this section that the mixed **CPP** is defined on a strongly connected graph $G = (V, A \cup E)$, where A is an arc set, E is an edge set, and the cost matrix $C = (c_{ij})$ is associated with $A \cup E$. Solving the **CPP** amounts to finding a minimum-cost augmentation of G by suitably replicating some arcs and edges, to satisfy the necessary and sufficient conditions for unicursality stated in the Introduction, and then determining an Eulerian traversal of the augmented graph.

In what follows, a graph is called *even* if the total number of arcs and edges incident to each of its vertices is even; it is *symmetric* if for each vertex the number of incoming arcs is equal to the number of outgoing arcs; a graph is *balanced* if the balanced set conditions are satisfied. Recall that a mixed connected graph is Eulerian if and only if it is even and balanced. If a graph is even and symmetric, then it is balanced, but symmetry is not a necessary condition for unicursality (see Figure 3). If it is known that G is Eulerian, one is left with the problem of determining an Eulerian circuit in G. This can be achieved in three phases: 1) assign directions to some edges so that G becomes symmetric; 2) assign directions to the remaining edges; 3) determine an actual traversal of G.

To derive a symmetric graph from G, one can apply the following procedure proposed by Ford and Fulkerson (p. 60).

Ford and Fulkerson's Procedure for Transforming a Mixed Graph Into a Symmetric Graph

STEP 1. Replace each edge of G with a pair of oppositely directed arcs, thus obtaining a directed graph $G' = (V, A')$. Assign to each arc of $A' \cap A$ a lower bound of 1 and to each arc of $A' \backslash A$ a lower bound of 0. Also assign to each arc of A' an upper bound of 1.

STEP 2. Using a network flow algorithm, determine a feasible circulation in G'. Let x_{ij} be the flow on arc (v_i, v_j).

STEP 3. Orient some edges of G as follows: If $(v_i, v_j) \in E$, $x_{ij} = 1$ and $x_{ji} = 0$, orient (v_i, v_j) from v_i to v_j.

Now that the graph is symmetric, the remaining edges can be oriented by means of the following $O(|A \cup E|)$ time procedure.

Procedure for Completely Orienting a Symmetric Graph

STEP 1. If all edges are directed, stop.

STEP 2. Let v be a vertex with at least one incident undirected edge (v, w). Set $v_1 := v$ and $v_2 := w$.

STEP 3. Orient (v_1, v_2) from v_1 to v_2. If $v_2 = v$, go to Step 1.

STEP 4. Set $v_1 := v_2$ and identify an edge (v_1, v_2) incident to v_1. Go to Step 3.

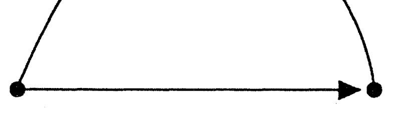
Figure 3. Example of a graph that is not symmetric but Eulerian.

Once a completely directed graph has been obtained, the van Aardenne-Ehrenfest and de Bruijn algorithm described in Section 3 can be applied to determine an Eulerian traversal of G.

As noted by Ford and Fulkerson, their maximal flow algorithm can be applied to an even graph not known a priori to be Eulerian: The algorithm will either produce a feasible flow or fail, in which case the graph is not Eulerian. If G is not Eulerian, a graph that satisfies the sufficient conditions for unicursality can be determined by augmenting G, i.e., by replicating a sufficient number of edges and arcs of G so that the resulting graph is Eulerian. To solve the mixed **CPP** on G, one must determine a minimum-cost augmentation. Note that this is not always feasible because some graphs cannot be made Eulerian (Figure 4).

Several authors have used integer linear programming to determine the minimum cost augmentation of a mixed graph, but computational results are sketchy and comparisons between the various approaches are hard to make. An example is provided by Grötschel and Win (1992): The ILP formulations these authors suggest for the **WPP** can be applied to the **CPP** through simple modifications. This formulation determines a minimal-cost augmentation of the graph while assigning a direction to every edge. The resulting graph is directed and symmetric because of constraints (19): It is therefore Eulerian. The role of the odd-cut inequalities is to strengthen the linear relaxation. Applying a constraint relaxation approach to this formulation, Grötschel and Win solved to optimality and without any branching, nine mixed **CPPs** out of nine that were attempted, in which $52 \leq |V| \leq 172$, $37 \leq |E| \leq 154$ and $31 \leq |A| \leq 116$.

The Grötschel and Win formulation without the odd-cut inequalities is identical to the Kappauf and Koehler (1979) formulation, apart from the variables' names. These authors did not, however, provide any computational results. Another formulation belonging to the same family was proposed by Christofides et al. (1984). Let $A_k^+ = \{(v_i, v_j) \in A : v_i = v_k\}$, $A_k^- = \{(v_i, v_j) \in A : v_j = v_k\}$ and V_k, the set of all vertices linked to v_k by an edge. Let x_{ij} be the number of extra times arc (v_i, v_j) is traversed in the optimal solution, and let y_{ij} be the total number of times edge (v_i, v_j) is traversed from v_i to v_j.

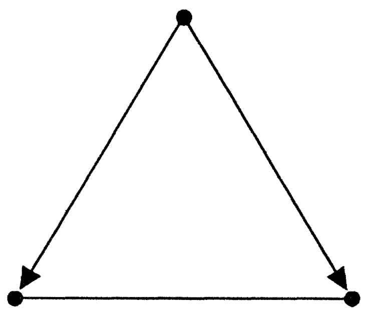
Figure 4. Example of a graph that cannot be made Eulerian.

Also let p_k be a binary constant equal to 1 if and only if the degree of vertex v_k is odd, and let z_k be an integer variable. The formulation is as follows.

Problem MCPP1

$$\text{Minimize} \sum_{(v_i,v_j)\in A} c_{ij}(1+x_{ij}) + \sum_{(v_i,v_j)\in E} c_{ij}(y_{ij}+y_{ji}) \tag{25}$$

subject to

$$\sum_{(v_i,v_j)\in A_k^+} (1+x_{ij}) - \sum_{(v_i,v_j)\in A_k^-} (1+x_{ij}) + \sum_{v_j\in V_k} y_{kj} - \sum_{v_j\in V_k} y_{jk} = 0 \quad (v_k \in V) \tag{26}$$

$$\sum_{(v_i,v_j)\in A_k^+} x_{ij} + \sum_{(v_i,v_j)\in A_k^-} x_{ij} + \sum_{v_j\in V_k} (y_{kj}+y_{jk}-1) = 2z_k + p_k \quad (v_k \in V) \tag{27}$$

$$y_{ij} + y_{ji} \geq 1 \quad ((v_i, v_j) \in E) \tag{28}$$

$$z_k, x_{ij}, y_{ij}, y_{ji} \geq 0 \text{ and integer.} \tag{29}$$

The problem is solved by means of an enumerative algorithm in which two different lower bounds are computed at each node of the search tree. The first bound is obtained by relaxing constraints (26) in a Lagrangian manner and solving the minimum-cost perfect matching. The second is obtained by the Lagrangian relaxation of constraints (28), and solving a minimum-cost flow problem. Using this approach, the authors solved to optimality 34 randomly generated problems with $7 \leq |V| \leq 50$, $3 \leq |A| \leq 85$ and $4 \leq |E| \leq 39$. The maximum CPU time observed was in the region of 500 seconds on a UNIVAC 1100/60.

In the formulation proposed by Nobert and Picard, there is only one variable y_{ij} associated with every edge of E. The ILP solution does not therefore specify edge directions. Constraints are imposed to ensure that the augmented graph satisfies the Ford and Fulkerson necessary and sufficient conditions for unicursality, i.e., the graph must be even and balanced. To present the formulation, define for any proper subset S of V the sets

$$A^+(S) = \{(v_i, v_j) \in A : v_i \in S,\ v_j \in V\backslash S\},$$

$$A^-(S) = \{(v_i, v_j) \in A : v_i \in V\backslash S,\ v_j \in S\},$$

$$E(S) = \{(v_i, v_j) \in E : v_i \in S,\ v_j \in V\backslash S \text{ or } v_i \in V\backslash S,\ v_j \in S\},$$

and let $u(S) = |A^+(S)| - |A^-(S)| - |E(S)|$. Thus, if $S = \{v_k\}$, then $A^+(S) = A_k^+$, $A^-(S) = A_k^-$, and $E(S) = E_k$. Constants p_k and variables z_k are defined as above, and only one variable y_{ij} is defined for each $(v_i, v_j) \in E$. Variable y_{ij} now represents the number of copies of edge (v_i, v_j) that must be added to the graph to make it Eulerian. The formulation is given next.

Problem MCPP2

$$\text{Minimize} \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} + \sum_{(v_i,v_j)\in E} c_{ij}y_{ij} \tag{30}$$

subject to

$$\sum_{(v_i,v_j)\in A} x_{ij} + \sum_{(v_i,v_j)\in E} y_{ij} = 2z_k + p_k \quad (v_k \in V) \tag{31}$$

$$-\sum_{(v_i,v_j)\in A^+(S)} x_{ij} + \sum_{(v_i,v_j)\in A^-(S)} x_{ij} + \sum_{(v_i,v_j)\in E(S)} y_{ij} \geq u(S) \quad (S \subset V,\ S \neq \varnothing) \tag{32}$$

$$z_k, x_{ij}, y_{ij} \geq 0 \text{ and integer.} \tag{33}$$

In this formulation, constraints (32) force all nonempty proper subsets S of V to be balanced: This is done by imposing that a sufficient number of arcs and edges will be introduced to compensate for the imbalance $u(S)$ of S. In addition to these constraints, the authors introduce a generalized form of blossom inequalities (see (10)):

$$\sum_{(v_i,v_j)\in A^+(S)} x_{ij} + \sum_{(v_i,v_j)\in A^-(S)} x_{ij} + \sum_{(v_i,v_j)\in E(S)} y_{ij} \geq 1 \quad (S \subset V,\ S \text{ odd}). \tag{34}$$

Again, these constraints are redundant, but help strengthen the LP relaxation. Nobert and Picard describe a procedure to identify the most unbalanced sets, based on previous algorithms by Picard and Ratliff (1975) and by Picard and Queyranne (1980). A constraint relaxation approach is used to solve **MCPP**: Initially, the program includes all nonnegativity constraints, balanced set constraints (32) corresponding to unbalanced vertices and to most of the unbalanced sets S of G, and generalized blossom inequalities (34) associated with odd vertices. In the course of the algorithm, additional balanced set and generalized blossom inequalities are generated as they are found to be violated. When this is no longer possible, a number of Gomory cuts are added to help gain integrality. The procedure may terminate at an integer or at a noninteger solution. If the solution is integer and satisfies all constraints, a minimum-cost Eulerian graph has been identified. Otherwise, a branching process has to be initiated. The algorithm was applied to a large number of randomly generated **MCPP**s with $16 \leq |V| \leq 225$, $2 \leq |A| \leq 5{,}569$ and $15 \leq |E| \leq 4{,}455$. Out of 440 problems that were attempted, 313 were solved to optimality without branching. The number of constraints generated during the course of the algorithm was usually of the order of $|V|$.

Finally, Minieka (1979) suggested that the mixed **CPP** can be formulated and solved as a network flow problem with gains (see Maurras). He did not, however, provide a formulation or an algorithm.

Heuristics for the mixed **CPP** have been suggested by Edmonds and Johnson (1973), and have been improved by Frederickson (1979), and by Christofides et al. (1984).

These heuristics seek good solutions satisfying the necessary and sufficient conditions for unicursality. Frederickson suggests two heuristics, **MIXED1** and **MIXED2**, each of time complexity $O(\max\{|V|^3, |A|(\max\{|A|, |E|\})^2\})$ and having a worst-case ratio of 2. If the two algorithms are applied in succession to the same problem, the worst-case ratio of this mixed heuristic goes down to 5/3. No results are reported on the computational behavior of these heuristics. We now sketch these two algorithms. Detailed procedures are provided in the original article.

Heuristic Mixed1 for the Mixed CPP

STEP 1. (**EVENDEGREE**) In a graph $G = (V, A \cup E)$, identify odd-degree vertices. Find all shortest paths between odd vertices, ignoring arc directions, and use these to determine minimum-cost matching of odd vertices. Augment the original graph by including all edges and arcs used for the matching solution.

STEP 2. (**INOUTDEGREE**) Using a minimum-cost flow algorithm, make the graph symmetric. Let $G' = (V, A' \cup E')$ be the resulting graph.

STEP 3. (**EVENPARITY**) Identify the set of odd vertices in G'. Identify cycles consisting of alternating paths (irrespective of arc directions) in $A'\backslash A$ and E', with every path anchored at each end by an odd vertex. As a cycle is covered, its arcs are either replicated or deleted, and its edges are directed, so that the resulting graph remains symmetric and becomes even.

As observed by Frederickson, Step 3 is made necessary in part by the fact that an even graph may not be preserved by Step 2. To see this, consider Figure 5 borrowed from Frederickson, showing: a) an even but asymmetric graph obtained at the end of Step 1, b) a symmetric but noneven graph at the end of Step 2, c) procedure **EVENPARITY** being applied to the graph after Step 2, and d) the final even, directed, and symmetric graph.

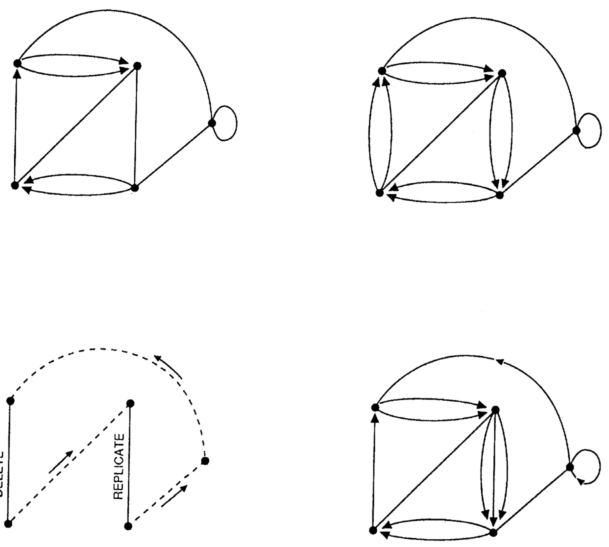

Figure 5. Solution produced by the **MIXED1** heuristic.

Heuristic Mixed2 for the Mixed CPP

STEP 1. (**INOUTDEGREE**) See Step 2 in heuristic **MIXED1**. Let $G' = (V, A' \cup E')$ be the resulting graph.

STEP 2. (**LARGECYCLES**) Identify the vertices of odd degrees in $G'' = (V, E')$. Find all shortest paths between these odd vertices over the graph $\bar{G} = (V, E)$. Perform a minimum-cost matching of the odd vertices of $\bar{G}$ using shortest path distances. Insert the edges used in the matching into E'. Find a traversal of the arcs and directed edges in A' and undirected edges in E'.

The algorithm of Christofides et al. is equivalent to **MIXED2** and will not be described here. It is worth noting, however, that when applied to 34 instances with $7 \leq |V| \leq 50$, $3 \leq |A| \leq 85$ and $4 \leq |E| \leq 39$, this heuristic produced solution values that were on the average 3% above the optimal value, the worst deviation being 17%. Finally, these authors have suggested the following improvement procedure applicable to any feasible solution: If the cost c_{ij} of an arc exceeds the length of a shortest path from v_j to v_i, then it is advantageous to replace (v_i, v_j) by that path in order to reduce the number of variables. This idea is also used by Gendreau, Laporte and Zhao in the context of the **WPP**.

6. THE HIERARCHICAL POSTMAN PROBLEM

The hierarchical postman problem (**HPP**) is defined on a directed or on an undirected graph $G = (V, A)$, where V contains a source s and a sink t. The arcs of A not incident to s or t are partitioned into $\{A_1, \ldots, A_k\}$, and an order relation $<$ is imposed on the elements of the partition. The **HPP** consists of determining a least-cost traversal of G starting at s, ending at t, and *servicing* the arcs of the partition in such a way that if $A_p < A_q$, then all arcs of A_p are serviced before any arc of A_q. However, arcs of A_q may be *traversed* before some arcs of A_p. This problem arises naturally in snow plowing operations where the sets A correspond to streets with different priority levels (Stricker 1970, Lemieux and Campagna 1984, Alfa and Liu 1988, Haslam and Wright 1991). Other applications are waste collection (Bodin and Kursh 1978) and flame cutting (Manber and Israni 1984).

Consider the subgraphs $G_p = (V_p, A_p)$ induced by the sets A_p. If each such subgraph is connected and the order relation is complete, i.e., a total order $A_1 < A_2 < \cdots < A_k$ is specified, the problem can be solved in $O(k|V|^5)$ time, as shown by Dror, Stern and Trudeau. For this, construct a directed acyclic graph $\bar{G} = (\bar{V}, \bar{A})$ as follows. Define $\bar{V}_i$ as the set containing one copy of each vertex incident to an arc of A_i, and let $\bar{V} = \{s, t\} \cup \bar{V}_1 \cup \ldots \cup \bar{V}_k$. The arc set $\bar{A}$ is made up of all arcs from s to the vertices of $\bar{V}_1$ and from the vertices of $\bar{V}_k$ to t, and of all arcs from the vertices of $\bar{V}_p$ to the vertices of $\bar{V}_{p+1}(p = 1, \ldots, k - 1)$. The cost c_{uv} of an arc (u, v) of $\bar{A}$ is determined as follows. If $u = s$ and $v \in \bar{V}_1$, then c_{uv} is the value of a shortest path (if G is directed) or chain (if G is undirected) from u to v; if $u \in \bar{V}_p$ and $v \in \bar{V}_{p+1}$ for $p = 1, \ldots, k$ and $\bar{V}_{k+1} = \{t\}$, then c_{uv} is the value of a minimum cost Eulerian path or chain starting at u, ending at v, and covering all arcs of A_p. The **HPP** is then solved by determining a shortest path form s to t in $\bar{G}$. To illustrate, consider the example depicted in Figure 6. Here the arcs from s and to t are shown in bold lines. The remaining arcs are partitioned as follows: $A_1 = \{(1, 2), (1, 5), (4, 5), (4, 7), (5, 7), (7, 8)\}$ in full light lines, $A_2 = \{(2, 5), (2, 6)\}$ in dashed lines, and $A_3 = \{(3, 5), (5, 8), (6, 8), (8, 9)\}$ in dotted lines. Hence, $\bar{V}_1 = \{1, 2, 4, 5, 7, 8\}$, $\bar{V}_2 = \{2, 5, 6\}$, $\bar{V}_3 = \{3, 5, 6, 8, 9\}$. In all arcs from s to t are indicated. All arcs from $\bar{V}_1$ to $\bar{V}_2$ and from $\bar{V}_2$ to $\bar{V}_3$ are defined, but these are not shown explicitly.

The complexity of this procedure is determined by the computation of the arc costs in $\bar{G}$. Since any vertex of $\bar{V}$ can belong to up to k vertex sets V_p, there are $O(k|V|^2)$ such arcs. Algorithms exist to compute each arc cost in $O(|V|^3)$ time, whether G is undirected or directed. In the undirected case, a better time complexity can be achieved by using a faster matching algorithm.

When subgraphs G_p are not connected or if the order relation is only partial, the problem becomes NP-hard. This can be shown by transformation of the recognition version of the **HPP** from the Hamiltonian path problem, known to be strongly NP-complete (Garey and Johnson 1979). Such transformations are provided by Dror, Stern

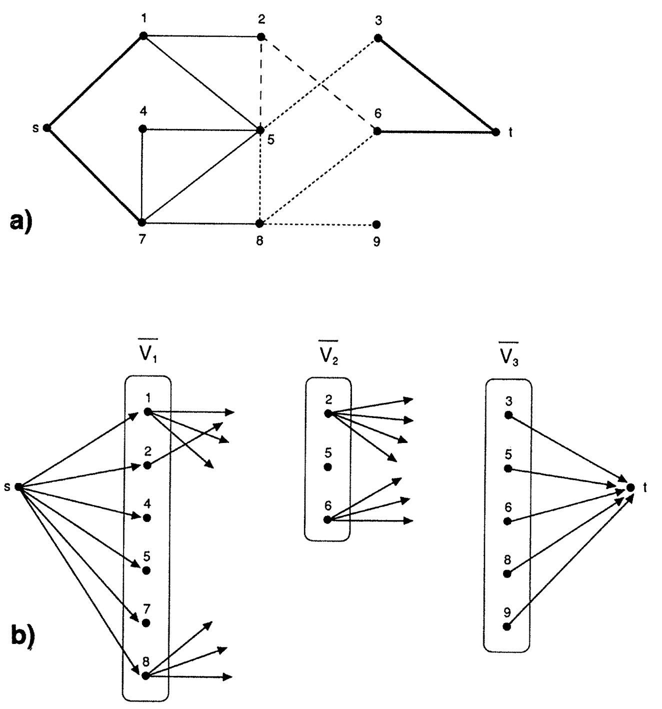

Figure 6. Construction of $\bar{G}$ from G: a) Graph G; b) Graph $\bar{G}$.

and Trudeau in the undirected case, and by Gélinas (1992) in the directed case.

Gélinas describes an exact enumerative algorithm for undirected **HPP**s with specified starting and ending vertices, in which the order relation is general, and all subgraphs G_p are strongly connected. The problem is solved

Table I
Summary of the Main Algorithmic Results for the **CPP**

Problem	Exact Algorithms	Heuristic Algorithms
	Constructing a Unicursal Graph	
Undirected **CPP**	Polynomial. Matching-based algorithm (Edmonds and Johnson 1973).	—
Directed **CPP**	Polynomial. Flow algorithm (Edmonds and Johnson 1973, Orloff 1974, Beltrami and Bodin 1974).	—
Windy **CPP**	NP-hard. Polynomial if graph is Eulerian (Win 1987). A cutting plane algorithm solves 31 instances out of 36: $52 \leq \lvert V\rvert \leq 264$, $78 \leq \lvert A\rvert \leq 489$ (Grötschel and Win 1988, 1992).	Rounding up fractional variables in LP relaxation (Grötschel and Win 1992).
Mixed **CPP**	NP-hard. Branch-and-cut. Grötschel and Win (1992) solve 9 instances out of 9 without branching: $52 \leq \lvert V\rvert \leq 172$, $37 \leq \lvert E\rvert \leq 154$, $31 \leq \lvert A\rvert \leq 116$. Nobert and Picard (1991) solve 313 instances out 414 without branching: $16 \leq \lvert V\rvert \leq 225$, $2 \leq \lvert A\rvert \leq 5{,}569$, $15 \leq \lvert E\rvert \leq 4{,}455$.	**MIXED1** and **MIXED2** applied in succession yields a worst-case ratio of 5/3 (Frederickson 1979). Christofides et al. (1984) propose an algorithm equivalent to **MIXED2**: solution values are on the average 3% above optimum.
Hierarchical **CPP**	NP-hard. Can be solved polynomially in $O(k\lvert V\rvert^5)$ time if each of the k subgraphs is connected and the order relation is complete (Dror, Stern and Trudeau 1987). DP-based algorithm for the undirected case, with strongly connected subgraphs and general order relation: $\lvert V\rvert = 16$, $\lvert A\rvert = 48$, $1 \leq k \leq 10$ (Gélinas 1992).	None available.
	Determining an Eulerian Cycle or Circuit	
Undirected case	Polynomial. End-pairing algorithm (Edmonds and Johnson 1973).	—
Directed case	Polynomial. Spanning arborescence algorithm (van Aardenne-Ehrenfest and de Bruijn 1951).	—

by means of a dynamic programming algorithm in which the states correspond to the subgraphs already traversed, and the last node of the current path. Various versions of the algorithm were tested on a problem with $|V| = 16$, $|A| = 48$, and $k = 1, 2, 3, 4, 6, 8$ and 10. In addition, a real-life problem associated with snow plowing operations in a Montreal district was solved to optimality. The dimensions of this problem were $|V| = 171$, $|A| = 544$, and $k = 4$.

7. CONCLUSION

The **CPP** is a basic graph theory problem that arises in a number of practical contexts. Table I summarizes the main computational results developed in this paper.

The pure undirected and directed versions of the problem are solvable in polynomial time and have attracted little attention lately. However, a number of graph covering problems by cycles or circuits are NP-hard and are still relatively unexplored. The mixed and windy **CPP** are also NP-hard. For these problems, branch-and-cut algorithms, such as those proposed by Nobert and Picard and by Grötschel and Win (1992), are promising optimizing approaches. Work on heuristics is scant and is badly needed. Finally, little is known on the hierarchical **CPP** despite the practical importance of this problem. Much research remains to be done on the development of fast, exact algorithms for the solvable cases, and on the design of heuristics.

ACKNOWLEDGMENT

This research was in part supported by the Canadian Natural Sciences and Engineering Research Council, under grants OGP0009160, OGP0038816, and OGP0039682. This support is gratefully acknowledged. Thanks are due to an anonymous referee for valuable comments.

REFERENCES

ALFA, A. S., AND D. Q. LIU. 1988. Postman Routing Problem in a Hierarchical Network. *Engin. Optim.* **14,** 127–138.

BARAHONA, F. 1990. On Some Applications of the Chinese Postman Problem. In *Algorithms and Combinatorics, Volume 9, Paths, Flows and VLSI-Layout*, B. Korte, L. Lovasz, H. J. Prömel, and A. Schriver (eds.). Springer-Verlag, Berlin.

BARAHONA, F., R. MAYNARD, R. RAMMAL AND J. P. UHRY. 1982. Morphology of Ground States of Two-Dimensional Frustration Model. *J. Phys. A (Math. and Gen.)* **15,** 673–699.

BELTRAMI, E. L., AND L. D. BODIN. 1974. Networks and Vehicle Routing for Municipal Waste Collection. *Networks* **4,** 65–94.

BENAVENT, E., V. CAMPOS, A. CORBERAN AND E. MOTA. 1983. Problemas de Rutas por Arcos. *Quaderns d'Estadística, Sistemas, Informàtica i Investigació Operativa (QÜESTIIÓ)* **7,** 479–490.

BODIN, L. D., AND S. J. KURSH. 1978. A Computer-Assisted System for the Routing and Scheduling of Street Sweepers. *Opns. Res.* **26,** 525–537.

BODIN, L. D., B. L. GOLDEN, A. A. ASSAD, AND M. O. BALL. 1983. Routing and Scheduling of Vehicles and Crews. The State of the Art. *Comput. & Opns. Res.* **10,** 63–211.

BRUCKER, P. 1981. The Chinese Postman Problem for Mixed Graphs. In *Graph Theoretic Concepts in Computer Science*, H. Noltemeier (ed.). Springer, Berlin, 354–366.

BUSACKER, R. G., AND T. L. SAATY. 1965. *Finite Graphs and Networks*. McGraw-Hill, New York.

CHRISTOFIDES, N. 1973. The Optimum Traversal of a Graph. *Omega* **1,** 719–732.

CHRISTOFIDES, N. 1975. *Graph Theory. An Algorithm Approach*. Academic Press, London.

CHRISTOFIDES, N., E. BENAVENT, V. CAMPOS, A. CORBERAN AND E. MOTA. 1984. An Optimal Method for the Mixed Postman Problem. In *System Modelling and Optimization*, Lecture Notes in Control and Information Sciences **59,** P. Thoft-Christensen (ed.). Springer, Berlin.

CROSS, H. 1936. Analysis of Flow in Networks of Conduits of Conductors. Bull. No. 286, University of Illinois Engineering Experimental Station, Urbana, Illinois.

DERIGS, U., AND A. METZ. 1991. Solving (Large Scale) Matching Problems. *Math. Prog.* **50,** 113–121.

DROR, M., H. STERN AND P. TRUDEAU. 1987. Postman Tour on a Graph With Precedence Relation on Arcs. *Networks* **17,** 283–294.

EDMONDS, J. 1965a. The Chinese Postman's Problem. *ORSA Bull.* **13,** 73.

EDMONDS, J. 1965b. Paths, Trees and Flowers. *Canadian J. Math.* **17,** 449–467.

EDMONDS, J., AND E. L. JOHNSON. 1973. Matching, Euler Tours and the Chinese Postman Problem. *Math. Prog.* **5,** 88–124.

EULER, L. 1736. Solutio Problematis ad Geometrian Situs Pertinentis. *Commentarii academiae scientarum Petropolitanae* **8,** 128–140.

EULER, L. (J. R. NEWMAN, ED.). 1953. Leonhard Euler and the Koenigsberg Bridges. *Sci. Am.* **189,** 66–70.

EVANS, J. R., AND E. MINIEKA. 1992. *Optimization Algorithms for Networks and Graphs*. Marcel Dekker, New York.

FLEISCHNER, H. 1990. *Eulerian Graphs and Related Topics* (Part 1, Volume 1), *Annals of Discrete Mathematics* **45.** North-Holland, Amsterdam.

FLEISCHNER, H. 1991. *Eulerian Graphs and Related Topics* (Part 1, Volume 2), *Annals of Discrete Mathematics* **50.** North-Holland, Amsterdam.

FORD, L. R., AND D. R. FULKERSON. 1962. *Flows in Networks*. Princeton University Press, Princeton, N. J.

FREDERICKSON, G. N. 1979. Approximation Algorithms for Some Postman Problems. *J. Assoc. Comp. Mach.* **26,** 538–554.

FREDERICKSON, G. N., M. S. HECHT AND C. E. KIM. 1978. Approximation Algorithms for Some Routing Problems. *SIAM J. Comp.* **7,** 178–193.

GALIL, Z., S. MICALI AND H. GABOW. 1986. An $O(EV \log V)$ Algorithm for Finding a Maximal Weighted Matching in General Graphs. *SIAM J. Comp.* **15,** 120–130.

GAREY, M. R., AND D. S. JOHNSON. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York.

Gélinas, É. 1992. Le problème du postier chinois avec contraintes générales de préséance. M.Sc.A. Dissertation, École Polytechnique de Montréal, Canada.

Gendreau, M., G. Laporte and Y. Zhao. 1990. The Windy Postman Problem on General Graphs. Publication #698, Centre de recherche sur les transports, Montreal, Canada.

Grötschel, M., and O. Holland. 1991. Solution of Large-Scale Symmetric Traveling Salesman Problems. *Math. Prog.* **51,** 141–202.

Grötschel, M., and Z. Win. 1988. On the Windy Postman Polyhedron. Report No 75, Schwerpunkt-program der Deutschen Forschungsgemeinschaft, Universität Augsburg, Germany.

Grötschel, M., and Z. Win. 1992. A Cutting Plane Algorithm for the Windy Postman Problem. *Math. Prog.* **55,** 339–358.

Guan, M. 1962. Graphic Programming Using Odd and Even Points. *Chinese Math.* **1,** 273–277.

Guan, M. 1984a. A Survey of the Chinese Postman Problem. *J. Math. Res. and Expos.* **4,** 113–119 (in Chinese).

Guan, M. 1984b. On the Windy Postman Problem. *Discr. Appl. Math.* **9,** 41–46.

Guan, M., and W. Pulleyblank. 1985. Eulerian Orientations and Circulations. *SIAM J. Alg. and Discr. Math.* **6,** 657–664.

Harary, F. 1969. *Graph Theory*. Addison-Wesley, Reading, Mass.

Hardgrave, W. W., and G. L. Nemhauser. 1962. On the Relation Between the Traveling Salesman Problem and the Longest Path Problem. *Opns. Res.* **10,** 647–657.

Haslam, E., and J. R. Wright. 1991. Application of Routing Technologies to Rural Snow and Ice Control. *Trans. Res. Rec.* No. 1304, 202–211.

Hierholzer, C. 1873. Uber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen* **VI,** 30–32.

Itai, A., and M. Rodeh. 1978. Covering a Graph by Circuits. *Lecture Notes in Computer Science* **62,** 289–299.

Itai, A., R. J. Lipton, C. H. Papadimitriou and M. Rodeh. 1981. Covering Graphs by Simple Circuits. *SIAM J. Comp.* **10,** 746–750.

Kappauf, C. H., and G. J. Koehler. 1979. The Mixed Postman Problem. *Discr. Appl. Math.* **1,** 89–103.

Kaufmann, A. 1967. *Graphs, Dynamic Programming and Finite Games*. Academic Press, New York.

Kesel'man, D. Y. 1987. Covering the Edges of a Graph by Circuits. *Kibernetica* **3,** 16–22 (translated from Russian).

Korte, B. 1989. Applications of Combinatorial Optimization. In *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe (eds.). Kluwer, Tokyo, 1–55.

Larson, R. C., and A. R. Odoni. 1981. *Urban Operations Research*. Prentice-Hall, Englewood Cliffs, N. J.

Lawler, E. L. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, New York.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. 1985. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, Chichester, U.K.

Lemieux, P. F., and L. Campagna. 1984. The Snow Ploughing Problem Solved by a Graph Theory Algorithm. *Civil Eng. Syst.* **1,** 337–341.

Lenstra, J. K., and A. H. G. Rinnooy Kan. 1976. On General Routing Problems. *Networks* **6,** 273–280.

Liebling, T. M. 1970. *Graphentheorie in Planungs—und Tourenproblemen*. Lecture Notes in Operations Research and Mathematical Systems **21,** Springer, Berlin.

Lin, Y., and Y. Zhao. 1988. A New Algorithm for the Directed Chinese Postman Problem. *Comput. & Opns. Res.* **15,** 577–584.

Male, J. W., J. C. Liebman and C. S. Orloff. 1977. An Improvement of Orloff's General Routing Problem. *Networks* **7,** 89–92.

Malek, M., A. Mourad and M. Pandya. 1989. Topological Testing. *Proceedings of the IEEE 1989 International Test Conference*, Paper 4.4, 103–110.

Manber, U., and S. Israni. 1984. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting. *J. Manuf. Syst.* **3,** 81–89.

Maurras, J. 1972. Optimization of Flow Through Networks With Gains. *Math. Prog.* **3,** 135–144.

Miliotis, P., G. Laporte and Y. Nobert. 1981. Computational Comparison of Two Methods for Finding the Shortest Complete Cycle or Circuit in a Graph. *RAIRO (Recherche Opérationnelle)* **15,** 233–239.

Minieka, E. 1979. The Chinese Postman Problem for Mixed Networks. *Mgmt. Sci.* **25,** 643–648.

Nobert, Y., and J.-C. Picard. 1991. An Optimal Algorithm for the Mixed Chinese Postman Problem. Publication #799, Centre de recherche sur les transports, Montreal, Canada.

Orloff, C. S. 1974. A Fundamental Problem in Vehicle Routing. *Networks* **4,** 35–64.

Padberg, M. W., and G. Rinaldi. 1991. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Rev.* **33,** 60–100.

Padberg, M. W., and M. R. Rao. 1982. Odd Minimum Cut-Sets and b-Matchings. *Math. Opns. Res.* **7,** 67–80.

Papadimitriou, C. H. 1976. On the Complexity of Edge Traversing. *J. ACM* **23,** 544–554.

Picard, J.-C., and H. D. Ratliff. 1975. Minimum Cuts and Related Problems. *Networks* **5,** 357–370.

Picard, J.-C., and M. Queyranne. 1980. On the Structure of all Minimum Cuts in a Network and Applications. *Math. Prog. Study* **13,** 8–16.

Richey, M. B., and R. G. Parker. 1991. A Cubic Algorithm for the Directed Eulerian Subgraph Problem. *Eur. J. Opnl. Res.* **50,** 345–352.

Sahni, S. K., and T. Gonzalez. 1976. P-Complete Approximations Problems. *J. ACM* **23,** 555–565.

Stricker, R. 1970. Public Sector Vehicle Routing: The Chinese Postman Problem. M.Sc. Dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge.

van Aardenne-Ehrenfest, T., and N. G. de Bruijn. 1951. Circuits and Trees in Oriented Linear Graphs. *Simon Stevin* **28,** 203–217.

Win, Z. 1987. Contributions to Routing Problems. Doctoral Dissertation, Universität Augsburg, Germany.

Win, Z. 1989. On the Windy Postman Problem on Eulerian Graphs. *Math. Prog.* **44,** 97–112.