20

The Chinese Postman Problem

Author: John G. Michaels, Department of Mathematics, State University of New York, College at Brockport.

Prerequisites: The prerequisites for this chapter are Euler circuits in graphs and shortest path algorithms. See, for example, Sections 7.5 and 7.6 of *Discrete Mathematics and Its Applications*, Second Edition, by Kenneth H. Rosen.

Introduction

The solution of the problem of the seven bridges of Königsberg in 1736 by Leonhard Euler is regarded as the beginning of graph theory. In the city of Königsberg there was a park through which a river ran. In this park seven bridges crossed the river. The problem at the time was to plan a walk that crossed each of the bridges exactly once, starting and ending at the same point. Euler set up the problem in terms of a graph, shown in Figure 1, where the vertices represented the four pieces of land in the park and the edges represented the seven bridges. Euler proved that such a walk (called an **Euler circuit**, i.e., a circuit that traverses each edge exactly once) was impossible because of the existence of vertices of odd degree in the graph. In fact, what is known as Euler's Theorem gives the precise condition under which an Euler circuit exists in a connected graph: the graph has an Euler circuit if and only if the degree of every vertex is even.

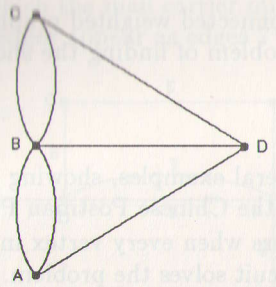


Figure 1. Königsberg bridge graph.

We now pose a related question about Figure 1. Starting and ending at point A, what is the minimum number of bridges that must be crossed in order to cross every bridge *at least* once? A problem of this type is referred to as a **Chinese Postman Problem**, named after the Chinese mathematician Mei-Ko Kwan, who posed such a problem in 1962 (see [3]). We will develop a method for solving problems of this type in this chapter.

Solution to the Problem

Before examining methods for solving Chinese Postman Problems, we first look at several applications of this type of problem. Suppose we have a road network which must be traversed by:

— a mail carrier delivering mail to buildings along the streets,
— a snowplow which must clear snow from each lane of the streets,
— a highway department crew which must paint a line down the center of each street,
— a police patrol car which makes its rounds through all streets several times a day.

In each case, the person (or vehicle) must traverse each street *at least once*. In the best situation, where every vertex (i.e., road intersection) has even degree, no retracing is needed. (Such retracing of edges is referred to as "deadheading".) In such a case, any Euler circuit solves the problem.

However, it is rarely the case that every vertex in a road network is even. (Examining a road map of any town or section of a city will confirm this.) According to Euler's Theorem, when there are odd vertices, it is impossible to plan a circuit that traces every edge exactly once. Since every road needs to be traced, some roads must be retraced. We pose a new problem — plan a route so that the total amount of retracing is as small as possible. More precisely, we have the following definition of the Chinese Postman Problem.

Definition 1 Given a connected weighted graph or digraph G, the *Chinese Postman Problem* is the problem of finding the shortest circuit that uses each edge in G at least once. □

We will now study several examples, showing how to solve problems that can be phrased in terms of the Chinese Postman Problem.

The simplest case occurs when every vertex in the graph has even degree, for in this case an Euler circuit solves the problem. Any Euler circuit will have the minimum total weight since no edges are retraced in an Euler circuit.

However, the following example illustrates the case where there are vertices of odd degree. This example will provide us with the first step toward a solution to the more general type of Chinese Postman Problem.

Example 1 A mail carrier delivers mail along each street in the weighted graph of Figure 2, starting and ending at point A. The first number on an edge gives the length of time (in minutes) needed to deliver mail along that block; the second number gives the time for traveling along the block without delivering mail (i.e., the deadheading time). Assuming that mail is delivered to houses on both sides of the street by traveling once down that street, find the shortest route and minimum time required for the mail carrier to deliver the mail.

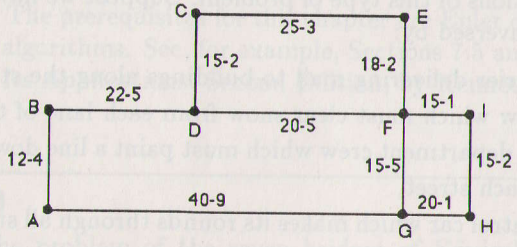


Figure 2. Mail carrier's weighted graph.

Solution: The total time spent on the route is the sum of the mail delivery time plus any deadheading time. The mail delivery time is simply the sum of the mail delivery weights on the edges, which is 217 minutes. The problem is now one of determining the minimum deadheading time.

Observe that the graph has no Euler circuit since vertices D and G have odd degree. Therefore the mail carrier must retrace at least one street in order to cover the entire route. For example, the mail carrier might retrace edges $\{D, F\}$ and $\{F, G\}$. If we insert these two retraced edges in the graph of Figure 2, we obtain a multigraph where every vertex is even. The new multigraph therefore has an Euler circuit. This graph is drawn in Figure 3, where the edges are

numbered in the order in which the mail carrier might follow them in an Euler circuit. (The deadheading edges appear as edges 7 and 12 in this multigraph.)

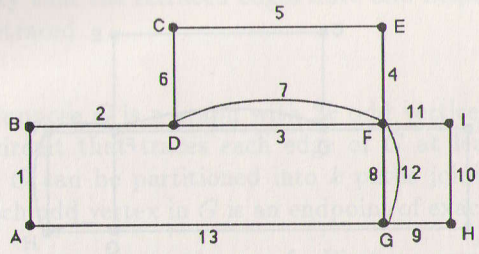


Figure 3. Mail carrier's route, including deadheading.

But is there a faster route? To answer this question, we examine the possible deadheading edges that could be added.

We claim that, no matter what route the mail carrier uses to deliver the mail, the edges used for deadheading will form a path joining D and G. To see this, consider the graph of Figure 3. Because D was odd in the original graph, a deadheading edge must be incident with D. (In our example, this is edge 7.) When this deadheading edge is added to the original graph, this causes its other endpoint, F, to become an odd vertex (its degree changed from 4 to 5). Since F is then odd, there must be another deadheading edge incident with F. (In our example, this is edge 12.) This will continue until a vertex that was originally odd is reached. (In our example, this stopped when G was reached.) Thus the deadheading edges will always form a path joining D and G.

Since the deadheading edges form a path joining D and G, to find the minimum weight of deadheading edges we need to find a path of minimum weight joining D and G in Figure 2.

The graph of Figure 2 is small enough that there are relatively few paths between D and G that we need to examine. (Note that we do not need to consider any path joining D and G that passes through a vertex more than once. Any such path could be replaced by a path of smaller weight that does not pass through any vertex more than once.) We list these paths along with their deadheading times:

Path	Time (minutes)
D, B, A, G	18
D, C, E, F, G	12
D, C, E, F, I, H, G	11
D, F, G	10
D, F, I, H, G	9

Therefore, the minimum deadheading time is nine minutes. This can be achieved by planning a route that uses the four edges $\{D, F\}$, $\{F, I\}$, $\{I, H\}$,

and $\{H, G\}$. The multigraph with these deadheading edges added is shown in Figure 4.

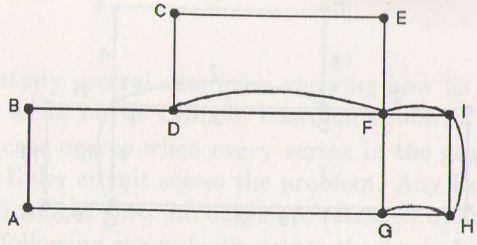


Figure 4. Mail carrier's weighted graph, with deadheading edges for minimum route added.

Any Euler circuit in the graph of Figure 4 achieves the minimum time. Thus,

$$A, B, D, C, E, F, I, H, G, F, D, F, I, H, G, A$$

and

$$A, B, D, C, E, F, D, F, I, H, G, F, I, H, G, A$$

are both examples of fastest routes.

The minimum time is

$$\text{delivery time} + \text{deadheading time} = 217 + 9 = 226 \text{ minutes.}$$

(Note that the four deadheading edges do not have to be used in succession. Also note that the *number* of streets retraced does not matter. The route in Figure 3 only retraced two blocks, but was not as fast as either of the two routes in the graph of Figure 4.) □

In this example we needed to traverse each edge at least once, forming a circuit that started and ended at A. No matter how we construct a circuit starting at A, we are forced to retrace edges. These retraced edges form a path between the two odd vertices. Therefore, to minimize the deadheading time, we need to find a path of minimum weight joining the two odd vertices.

In Example 1 there were sufficiently few paths joining the odd vertices that we could list them all and select one of minimum length. However, if there are many possibilities to consider, an algorithm for finding a shortest path should be used. (See the "Shortest Path Problems" chapter of this book or Dijkstra's shortest path algorithm in Section 7.6 of *Discrete Mathematics and Its Applications*, Second Edition, by Rosen.)

But now suppose that we are working with a weighted graph with *more than two* odd vertices. As in Example 1, we know that edges will need to

be retraced and we need to minimize the total weight of the retraced edges. The following theorem, generalizing the idea developed in Example 1, shows an important property that the retraced edges have and helps us efficiently select the edges to be retraced.

Theorem 1 Suppose G is a graph with $2k$ odd vertices, where $k \geq 1$, and suppose C is a circuit that traces each edge of G at least once. Then the retraced edges in C can be partitioned into k paths joining pairs of the odd vertices, where each odd vertex in G is an endpoint of exactly one of the paths.

Proof: There are two parts to the proof. First we prove that the retraced edges of C can be partitioned into k paths joining pairs of odd vertices, and then we prove that each odd vertex is an endpoint of exactly one of these paths.

The proof of the first part follows the argument of Example 1. We choose an odd vertex v_1. Since v_1 has odd degree, the vertex must have at least one retraced edge incident with it. We begin a path from v_1 along this edge, and continue adding retraced edges to extend the path, never using a retraced edge more than once. Continue extending this path, removing each edge used in the path, until it is impossible to go farther. The argument of Example 1 shows that this path will only terminate at a second odd vertex v_2. The vertex v_2 must be distinct from v_1. (To see this, note that v_1 has odd degree in G. When the retraced edges are added to G, v_1 has even degree. Therefore there must be an odd number of retraced edges incident with v_1. One of these is used to begin the path from v_1; when it is removed, v_1 has an even number of retraced edges remaining. Therefore, if the path from v_1 ever reenters v_1, there will be a retraced edge on which it will leave v_1. Therefore, the path will not terminate at v_1.) We then begin a second path from another odd vertex v_3, which will end at another odd vertex v_4. (The parenthetical remarks earlier in this paragraph showing that v_2 must be distinct from v_1 can also be applied here to show that v_4 must be different from v_1, v_2, and v_3.) Proceeding in this fashion, every retraced edge will be part of one of these paths.

The second part of the theorem follows immediately, since each odd vertex appeared as an endpoint of some path, and once this happened, that vertex could never be an endpoint of a second path. ■

The following example illustrates the use of Theorem 1 in solving a problem of the Chinese Postman type.

Example 2 A truck is to paint the center strip on each street in the weighted graph of Figure 5. Each edge is labeled with the time (in minutes) for painting the line on that street. Find the minimum time to complete this job and a route that accomplishes this goal. Assume that the truck begins and ends at

a highway department garage at A and that the time required for the truck to follow a street is the same regardless of whether or not the truck is painting or deadheading.

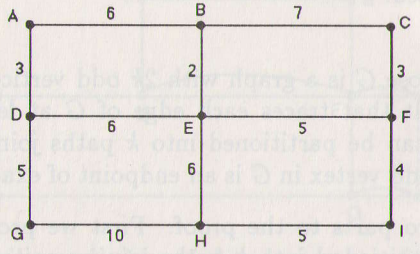


Figure 5. Weighted graph for the line-painting truck.

Solution: Observe that there are four odd vertices — B, D, F, H. Therefore, traversing each edge on this graph at least once will require deadheading at each of these vertices. To minimize the total time for the job, we need to minimize the deadheading time. That is, we need to minimize the sum of the weights of all duplicated edges. Using Theorem 1 (with $k = 2$), we know that the deadheading edges form two paths joining pairs of the four odd vertices.

Therefore, to solve the problem we need to find the two paths with total weight as small as possible. We first need to list all ways to put the four odd vertices in two pairs. Then, for each set of two pairs we find a shortest path joining the two vertices in each of the two pairs, and finally we choose the set of pairs that has the smallest total weight of its two paths.

In this example, the pairings and shortest paths (with their weights) are

Pairing	Path	Weight	Path	Weight
$\{B,D\}, \{F,H\}$:	B,E,D	8	F,I,H	9
$\{B,F\}, \{D,H\}$:	B,E,F	7	D,E,H	12
$\{B,H\}, \{D,F\}$:	B,E,H	8	D,E,F	11

Of these three possible pairings, the pair $\{B,D\}$,$\{F,H\}$ has smallest total weight, $8 + 9 = 17$ minutes. Therefore, the truck will spend 17 minutes deadheading, and the total time on the job spent by the truck will be

$$\text{painting time} + \text{deadheading time} = 62 + 17 = 79 \text{ minutes.}$$

To find a specific route that achieves this time, take the given graph and add the retraced streets as multiple edges, as in Figure 6. This new graph has an Euler circuit, such as

$$A, B, C, F, E, B, E, D, E, H, I, F, I, H, G, D, A. \qquad \square$$

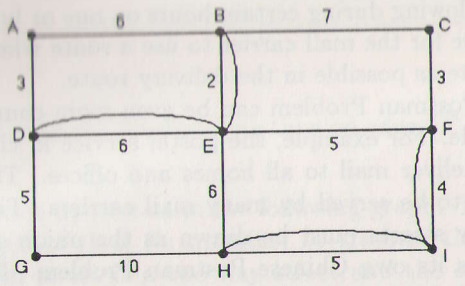


Figure 6. Weighted graph for a snow plow, with deadheading edges.

The key idea in the solution of the problem in Example 2 is finding a **perfect matching of minimum weight**, that is, a pairing of vertices such that the sum of the weights on the paths joining the vertices in each pair is as small as possible. To obtain such a matching, we examined all possible ways of matching the odd vertices in pairs, and then chose a minimum (i.e., shortest) matching. If there are n odd vertices, the number of perfect matchings to examine is $O(n^{n/2})$. (See the Chapter "Applications of Subgraph Enumeration" in this book for a method of enumerating these matchings.) Therefore, if n is large, we would need the assistance of a computer to accomplish the task. An algorithm of order $O(n^3)$ for finding a matching of minimum weight can be found in reference [4], for example.

Extensions

In each problem discussed in the last section, some simplifications were implicitly made. For example, in the mail carrier's problem of Example 1, we assumed that the person delivering the mail was restricted to traveling along only the streets in the given graph. Thus, when we minimized the deadheading time, we looked for the shortest path between D and G, *using only the given edges.* However, it is possible that there are additional paths or alleys (along which no mail is to be delivered) that will yield a shorter path between D and G. In this case we would need to expand the given graph to include all possible edges that could be used as parts of paths for deadheading, and then find a perfect matching of minimum weight in this new graph.

There are several other considerations that may need to be considered in a specific problem. For example, it may be difficult for a street sweeper to make a left turn at intersections. A snowplow may need to clear roads along a street network where some left turns are prohibited, some of the streets are one-way (and therefore may need to be followed twice in the same direction), or parked

cars may prevent plowing during certain hours on one or both sides of a street. Also, it may be wise for the mail carrier to use a route where the deadheading periods occur as late as possible in the delivery route.

The Chinese Postman Problem can be even more complicated when handled on a large scale. For example, the postal service in city does not depend on one person to deliver mail to all homes and offices. The city must be divided into sections to be served by many mail carriers. That is, the graph or digraph for the city streets must be drawn as the union of many subgraphs, each of which poses its own Chinese Postman Problem. But how should the large graph be divided into subgraphs so that various factors (such as time and money) are minimized?

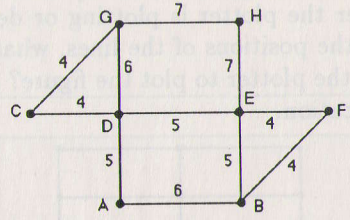
A model for solving problems such as this is explained in [7], where a model for street sweeping in New York City is discussed in detail. Two approaches to the problem are dealt with: one method that finds a single route for the entire graph and then divides this route into smaller pieces (the "route first–cluster second" approach), and a second method that first divides the graph into pieces and then works with each piece separately (the "cluster first–route second" approach). Methods such as these have been implemented and have resulted in considerable cost savings to the municipalities involved.

Suggested Readings

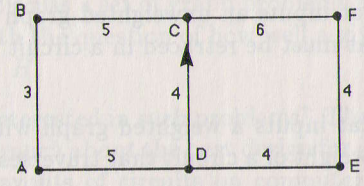
1. J. Edmonds, "The Chinese Postman Problem", *Operations Research*, Vol. 13, Supplement 1, 1965, B73.
2. S, Goodman and S. Hedetniemi, "Eulerian Walks in Graphs", *SIAM Journal of Computing*, Vol. 2, 1973, pp.16–27.
3. M. Kwan, "Graphics Programming Using Odd and Even Points", *Chinese Math.*, Vol. 1, 1962, pp. 237–77.
4. E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.
5. E. Reingold and R. Tarjan, "On a Greedy Heuristic for Complete Matching", *SIAM Journal of Computing*, Vol. 10, 1981, pp. 676–81.
6. F. Roberts, *Applied Combinatorics*, Prentice-Hall, Englewood Cliffs, N.J., 1984.
7. A. Tucker and L. Bodin, "A Model for Municipal Street Sweeping Operations", Chapter 6 in *Discrete and System Models*, ed. W. Lucas, F. Roberts, and R. Thrall (Volume 3 of *Models in Applied Mathematics*), Springer-Verlag, New York, 1983, pp. 76–111.

Exercises

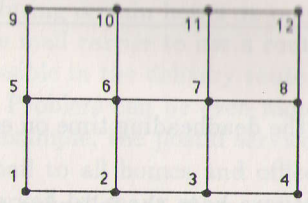
1. Solve Example 1 with the deadheading time on edge $\{F, I\}$ changed from 1 to 4.
2. A police car patrols the streets in the following graph. The weight on each edge is the length of time (in minutes) to traverse that street in either direction. If each street is a two-way street and the car starts and ends at A, what is the minimum time needed to patrol each street at least once?



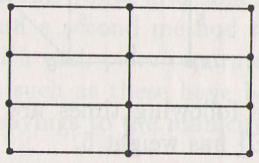
3. Solve Example 2 if the following times are changed: edge $\{G, H\}$ has weight 3 and edge $\{B, E\}$ has weight 5.
4. If each bridge in the Königsberg bridge graph (Figure 1) must be crossed at least once, what is the minimum number of bridges that must be crossed more than once, if the circuit begins and ends at A?
5. Find the minimum number of edges that must be retraced when drawing each graph as a circuit.
   a) K_6 b) K_n c) $K_{2,5}$ d) $K_{m,n}$.
6. A street washing truck needs to wash the streets of the following map. The labels on the edges give the time (in minutes) to wash that street. All streets are two-way, except that the street between D and C is one-way northbound. If the truck starts and ends its route at A, find a route that cleans all streets in the smallest amount of time. Assume that one pass down the center of a street washes the street and that the truck must observe the one-way restriction on the street joining D and C.



7. Find the minimum number of edges that must be retraced in the graph of the following figure, where each edge is traced at least once, starting and ending at vertex 1.



8. A computer plotter plots the following figure. It takes the plotter 0.05 seconds to plot each horizontal edge and 0.02 seconds to plot each vertical edge. Assuming that the time required to trace a line remains the same regardless of whether the plotter is plotting or deadheading and that the plotter must follow the positions of the lines, what is the minimum length of time required for the plotter to plot the figure? *Note*: See [5] for further details on this application.



9. Suppose a mail carrier delivers mail to both sides of each street in the graph of Figure 2, and does this by delivering mail on only one side of the street at a time. The mail delivery time for each side of a street is half the weight on the edge. (For example, it takes 6 minutes to deliver mail on each side of street $\{A, B\}$). Find the minimum time needed to complete the route.

10. An 8×8 checkerboard is to be traced as a circuit. What is the smallest number of edges that must be retraced in order to trace the board?

Computer Projects

1. Write a program that inputs an unweighted graph and finds the minimum number of edges that must be retraced in a circuit that traces each edge at least once.

2. Write a program that inputs a weighted graph with two odd vertices and finds the minimum weight of a circuit that traverses each edge at least once.

3. Write a program that inputs a weighted graph with $2k$ odd vertices and finds the minimum weight of a circuit that traverses each edge at least once.

21

Graph Layouts

Author: Zevi Miller, Department of Mathematics and Statistics, Miami University.

Prerequisites: The prerequisites for this chapter are big-O notation and basic concepts of graphs and trees. See, for example, Section 1.8 and Chapters 7 and 8 of *Discrete Mathematics and Its Applications*, Second Edition, by Kenneth H. Rosen.

Introduction

In this chapter we will discuss a set of graph problems having a common theme. These problems deal with the question of how well a given graph G "fits" inside some other given graph H.

Why might we be interested in such problems? The answer is that solutions to such problems tell us much about the best design for interconnection networks of computers and for layouts of circuits on computer chips. The solution to a special case of one of these problems also sheds light on how to represent matrices in a compact way that makes it convenient to manipulate them when executing standard matrix operations.