Eulerian and Hamiltonian Cycles

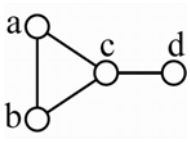
Complement to Chapter 6, "The Case of the Runaway Mouse"

Let's begin by recalling a few definitions we saw in the chapter about line graphs.

Definition

A cycle that travels exactly once over each edge in a graph is called "Eulerian."
A cycle that travels exactly once over each vertex in a graph is called "Hamiltonian."

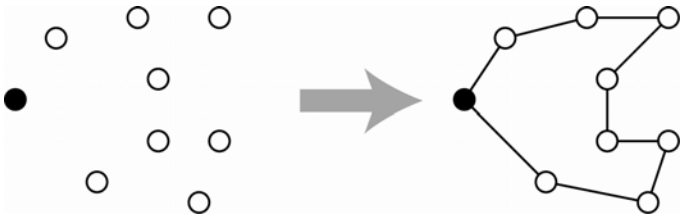
Some graphs possess neither a Hamiltonian nor a Eulerian cycle, such as the one below.



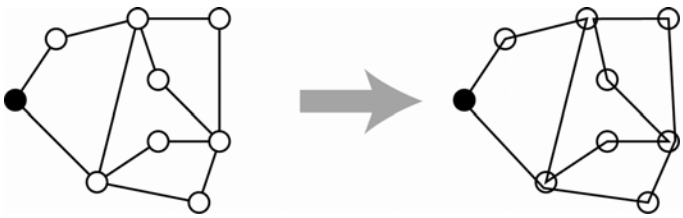
Let's note that we define Hamiltonian and Eulerian *chains* the same way, by replacing *cycle* with *chain*. The graph above has a Hamiltonian chain, for example: d-c-a-b. It also has a Eulerian chain, for example: d-c-a-b-c.

Applications

A business traveller leaves every morning from his home, which is represented in black below, and needs to visit a number of customers, represented in white, and then go back home. How should he go about minimizing the total distance he travels? (We are supposing that the distances between each pair of customers, as well as between the customers and the traveller's home, are known.) Here we are looking for a Hamiltonian cycle of the minimum possibly length.



A garbage collection truck leaves from the depot, represented in black below, and needs to travel along each street in the network below to carry out its garbage collection. How should it do that? Here, we are looking for a Eulerian cycle.



The cycles (not necessarily Hamiltonian or Eulerian) share a key property that Manori points out to Courtel when he's searching for the mouse. This property, set out below, can easily be extended to chains.

Property

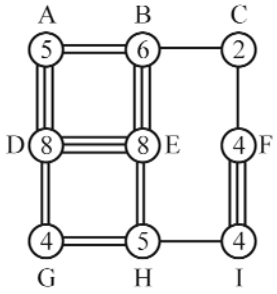
When a cycle is drawn in a graph, all the vertices are necessarily of an even degree because every time we enter a vertex, we leave it as well.

When a chain is drawn in a graph, all the intermediate vertices (meaning those that are not at the ends of the chain) are of an even degree.

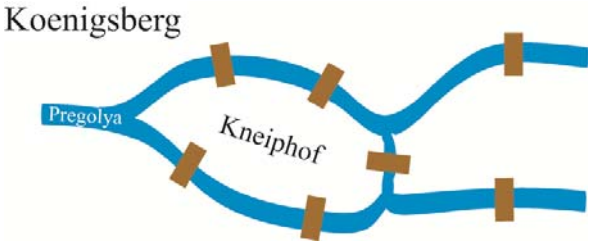
This is the property that Manori uses to find the mouse. The mouse's trajectory from its cage to its hiding place is a cycle (if it's hiding in the laboratory where it disappeared) or a chain. Manori draws the graph containing all the edges the mouse moved along. He can do this thanks to the information provided by the detectors. His reasoning goes like this:

- If all the vertices in the graph are of an even degree, the mouse came back to its departure point, and will be very difficult to find;
- If the graph only has two vertices of an odd degree, one of them is the location of its escape, and the other is its hiding place;
- If the graph includes more than two vertices of an odd degree, the information gathered is incoherent.

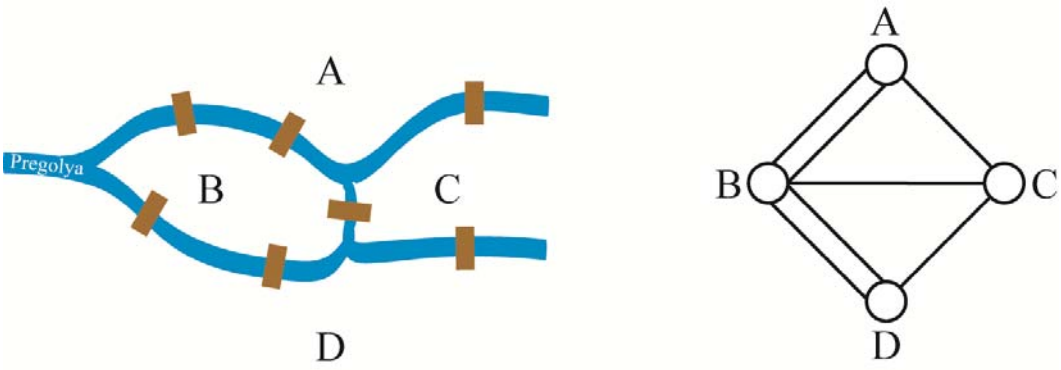
By chance, the second situation is applicable, and one of the odd-degree vertices (vertex A) is a laboratory while the other (vertex H) is a zone between laboratories with a single air duct. The mouse must necessarily be in there.



In the 18th century, the great mathematician Euler solved the following problem. The city of Koenigsberg (later called Kaliningrad) is crossed by the Pregolya, which runs from the island of Kneiphof and has seven bridges, as shown in the figure below. Can a pedestrian, when taking a walk, cross each bridge once and only once?



To solve this problem, Euler built a graph G whose vertices are the different adjacent regions, if there were no bridge, and where each edge represents a link between two regions via a bridge. In order for a pedestrian to cross each bridge exactly once, G must contain a Eulerian cycle, which is not the case because G contains four vertices of an odd degree.

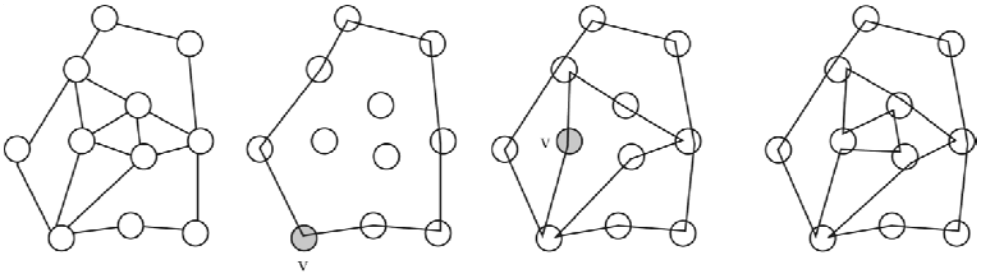


When all the vertices in a graph are of an even degree it is easy enough to draw a Eulerian cycle. We can do this as follows, for instance.

Construction of a Eulerian cycle in a graph where all the vertices are of an even degree

1. Determine any cycle C.
2. If C covers all the graph's edges, then stop. If not, continue to 3.
3. Choose a vertex v in C which is the end of an edge outside of C. Build a second cycle, C', containing v such that C and C' have no edges in common.
4. Combine C and C' to form a cycle C''. This fusion is done starting with vertex v, running through the entire C cycle, coming back to vertex v, and visiting the entire C' cycle to finally end on v.
5. Set C as equal to C'' and return to 2.

Illustration



Definition

Let's suppose that a distance is associated with each edge. The problem of determining a Hamiltonian cycle of a minimal total distance is called the "travelling salesman problem."

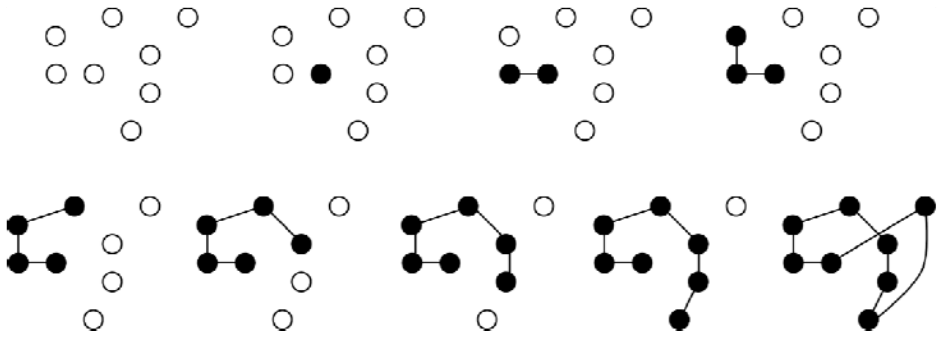
The travelling salesman problem is very difficult to solve. There are, however, software programs that can determine optimal solutions as long as the number of vertices does not go beyond a few hundred (such as Concorde for instance, which you can download from the website http://www.tsp.gatech.edu/concorde.html). Achieving an optimal solution can take hours, even days.

When we are dealing with large graphs or we want to obtain solutions very quickly, we use what we call *heuristics*, which provide solutions of reasonable quality in a reasonable time.

Nearest neighbour heuristic

1) Choose a starting vertex x.
2) As long as all the vertices have not yet been visited, move to the closest vertex not yet visited

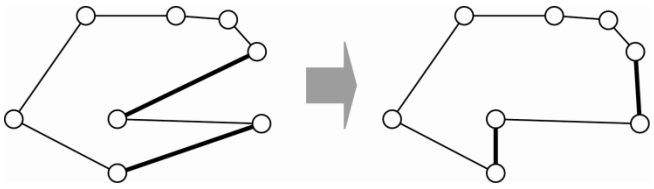
Example



The solutions obtained with this heuristic were compared to those produced by the Concorde program. The error is at an average of about 15%.

All heuristics, including that of the nearest neighbour, can easily be improved by adding a *post-optimization* procedure at the end. This consists in verifying whether there exists a pair of edges on the cycle that could be replaced by another pair of edges (there is only one replacement possible per pair of edges) while reducing the total distance travelled. As long as such pairs exist, the changes are made.

Illustration



This post-optimization procedure makes it possible to considerably reduce the difference between the solution and the optimum. For example, by comparing the solutions produced by the Concorde program with the combination of the nearest neighbour heuristic followed by post-optimization, we can observe that the error is in the range of 2 to 3%.

There are methods to reduce this gap to a few tenths of a percent, but these methods are much more complex to put into place.

Let's come back to Eulerian cycles for a moment. As already mentioned, if all the vertices in a graph G are of an even degree then it is easy to determine a Eulerian cycle, whereas when there is at least one vertex of an odd degree, the graph does not contain any Eulerian cycles. In a Eulerian cycle, we must travel *exactly* one time over each edge in the graph. Here we are looking at a problem that consists in determining a cycle that travels *at least once* over each edge in the graph.

Definition

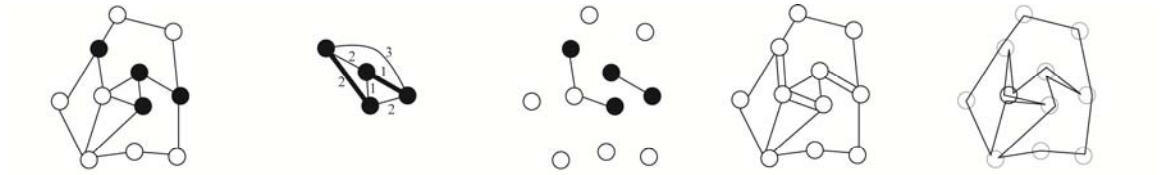
Let's suppose that a distance is associated with each edge of a connected graph. The "Chinese postman problem" consists in determining the shortest possible cycle that travels at least once over each edge in the graph.

Unlike the travelling salesman problem, the Chinese postman problem is "easy" to solve. If there are no vertices of an odd degree, we already know how to determine a Eulerian cycle, and this cycle is necessarily the shortest one because it travels exactly once over each edge. Otherwise, we can proceed as follows.

Algorithm for determining an optimal solution to the Chinese postman problem in a G graph containing vertices of an odd degree

1. Create a complete H graph in which the vertices are the odd-degree vertices from G and link every pair of vertices in H by an edge. We associate a cost with each edge in H; a cost is defined as the length of the shortest chain in G linking the vertices in question.
2. Determine a minimum-cost matching in H.
3. For each edge in the optimal matching determined in 2, add the shortest corresponding chain in G.
4. The resulting graph's vertices are all of an even degree. We can thus easily build a Eulerian cycle.

Illustration



Graph G with four vertices of an odd degree (in black)

H graph supposing that every distance between two vertices in G is of one unit; the minimum cost matching is in bold lines

Edges to add in G

Graph G with added edges; all vertices are of an even degree

Optimal cycle for the Chinese postman problem

In the example above, we suppose that every distance between two vertices in G is of one unit. Note that for point 2 of the algorithm to make sense, and so that we can transform all the odd-degree vertices in G into even-degree vertices, there needs to be an even number of odd-degree vertices in G. This is always the case, as demonstrated below.

Property

All graphs contain an even number of odd-degree vertices.

Proof

We can define a partition $V=E \cup O$ for the vertex set V in a graph by defining E as the set of even-degree vertices and O as the set of odd-degree vertices. So we get:

$$\sum_{v \in V} \text{degree}(v) = \sum_{v \in E} \text{degree}(v) + \sum_{v \in O} \text{degree}(v)$$

We have already shown that this total is even, because it is equal to twice the number of edges in the graph. Since the first term to the right of the equal sign in the equation above is even (it is a sum of even numbers), the second term must also be even. For a total of odd numbers to be even, we need to add together an even number of numbers, and so we deduce that the O set contains an even number of vertices. □

A problem similar to the Chinese postman problem consists in determining a minimum-cost trip that travels at least once over a fixed subset of edges. In practice, the road network of a city is made up of many edges, and garbage must only be collected on some edges of a graph. The truck tasked with garbage collection can still take other edges in the graph to travel from one point to another.

Definition

Let's suppose that a distance is associated with each edge of a connected graph G, and that D is a subset of edges to travel along in G. The "rural postman problem" consists in determining the shortest possible cycle in G that travels at least once over each edge in D.

The rural postman problem is a very difficult one to solve, like the travelling salesman problem. In fact, the real-life problems are much more complex than the ones mentioned above, because we need to take numerous constraints into account, such as vehicle capacity, driver breaks and so forth.

For example, when the total quantity to deliver or collect from clients exceeds a vehicle's capacity, several vehicles need to be used. The problem becomes more complicated then because in addition to determining the route for each vehicle, we also need to divide the customers among the vehicles.