Hamiltonian and Euler Paths

Eulerian Graphs

»The Königsberg Bridge Problem«
Definitions
Theorem of Euler
Applications

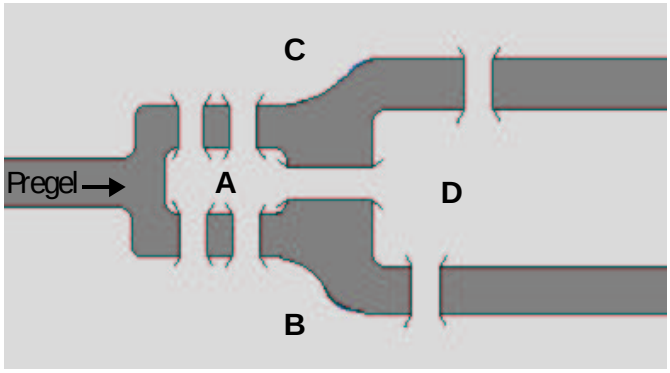
Hamiltonian Graphs

Definitions
Characteristics of Hamiltonian graphs
Introduction to the TSP
Approximation Algorithms
Applications

Proseminar Algorithms SS 01
Bianka Backin

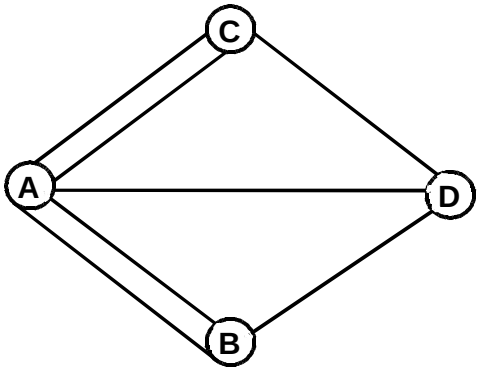
Eulerian Graphs

»The Königsberg Bridge Problem«



Given is the inner city of Königsberg with the river Pregel, which devides the city into four land regions A, B, C and D. In order to travel from one part of the city to another one there exist 7 bridges. Now there was the townsfolk suspicion that it is not possible to find a tour through Königsberg which crosses each bridge exactly once. In 1736 L. Euler solved the problem by using a graph. The vertices of the graph stand for the land regions and the edges for the bridges.

If there exists a tour, it can be described by a sequence of 8 letters, in which two adjacent letters represent an edge.



Now Euler said that each time an edge enters an vertex it also leaves it again, except the beginning and ending vertices. So we can say that A has to appear at least 3 times and B,C,D at least 2 times. But then we get a sequence longer than 8 and therefore the townsfolk suspicion was right.

The solution of Euler for »The Königsberg Bridge Problem« is said to be the origin of graph theory.

To continue we need some definitions.

The first definition is the one of Eulerian trails.

Definitions

An Eulerian trail is an open trail in a graph G that contains every edge of G exactly once.

If the Eulerian trail is a circuit we speak of an Eulerian circuit and the graph is then called an Eulerian graph.

With these definitions it is possible to understand the Theorem of Euler.

The Theorem of Euler (1736)

A connected multigraph G contains an Eulerian trail, if the number of vertices p with odd degree is either 0 or 2.

If p=0 there exists an Eulerian circuit.

Proof: "⇒" An Eulerian trail exists. Therefore all vertices except the beginning and ending vertices a and b must have even degrees. If a≠b, then a and b are the only vertices of odd degree. If a=b, there does not exist any vertex of odd degree and hence G is an Eulerian circuit.
"⇐" Induction on the number of vertices.
Induction beginning for $|V| \leq 2$. Distinction between p=0 and p=2.
Induction assumption: For $|V| \leq n$ the assertion is correct.
Induction step |V|=n+1: For p=2 and the vertices a,b of odd degree we find a longest path beginning at a and ending in b. If edges are left after this step the induction assumption can be used. For p=0 it works the same way, but the path begins and ends in a.

With this proof it is possible to create an algorithm which finds Euler paths in graphs. In the first step it is necessary to find the vertices of odd degree. Then a path can be build up. If there are unvisited edges left, find circuits which begin at points of the already found path and put them into the path.
The complexity of the algorithm is O(|V|+|E|).

As it is often very important to find paths that visit certain things exactly once, there exist many problems in real life which can be solved by Eulerian graphs.

Applications of Eulerian graphs

Eulerian graphs are necessary to solve problems in coding, telecommunications and the developement of parallel programming, but also to find results for less important problems as for example the Domino Problem.
Another very important field where Eulerian graphs are used in a quiet different way is the Chinese postman problem. The aim of this problem is to find the shortest path in a graph which can also have only one vertex of odd degree or more than two vertices of odd degree. Therefore it is necessary to visit edges more often than once and of course it’s then better to visit edges of small weights.
The algorithm of the Chinese postman problem makes use of the minimum and maximum weight matching which offers the "best" combination of edges to be visited more often than once.
An example where the algorithm of this problem is used is the garbage collection.

Hamiltonian Graphs

Definitions

In contrast to Eulerian graphs the aim is not to find a path in a graph that contains all the edges, but to find a tour that visits every vertex of a graph exactly once.
Hence there exist other definitions of trails and circuits.
A Hamiltonian trail is a path in a graph that passes every vertex exactly once. Whereas a Hamiltonian circuit is a circuit in a graph that contains every vertex. The graph is then called a Hamiltonian graph.
Although Eulerian and Hamiltonian graphs seem to be quite similar there are big differences.

Characteristics of Hamiltonian graphs

Intuitivly one can say that a graph is more likely to be Hamiltonian the more edges it has, because then there exist more possibilities to connect the different vertices with each other.
Dirac says in his theorem that a graph G of order p = 3 is Hamiltonian, if deg v = p/2 for every vertex of G.
With this theorem it's practicable to find out whether a graph is Hamiltonian or not, but there doesn't exist any efficient algorithm to determine the circuit. The reason for that is that the problem is NP−complete.
This means for example when one tries to find a circuit in a graph it is necessary to determine all permutations of vertices. If there exists a sequence $\{v_1,v_2\},\{v_2,v_3\}$, ... ,$\{v_{n-1},v_n\},\{v_n,v_1\}$ the graph is Hamiltonian. The complexity of this is $O((n/e)^n)$, at which $(n/e)^n = n!$ is the number of permutations of vertices.
As there does not exist any satisfactory solution for this problem it is helpful to look at similar problems.

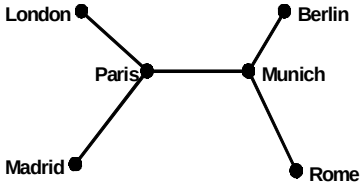
Introduction to the TSP

The traveling−salesman−problem (TSP) deals with the subject of determining a tour of lowest cost in a complete weighted graph by visiting each vertex exactly once.
As before with the Hamiltonian circuit the number of possible solutions is |V|! and therefore an exhaustive search is not efficient.
In contrast to the Hamiltonian problem there exist approximations for the TSP which make use of the triangle inequality.

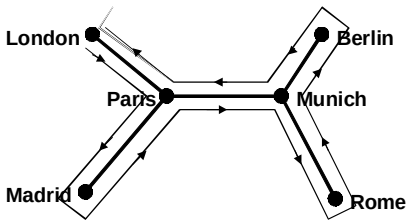
The triangle inequality means that, when there exists a set of three vertices A,B,C the cost of traveling directly from A to C is lower than traveling from A to B and then from B to C.
With this precondition it is possible to find effective approximation algorithms to solve the TSP.

Approximation Algorithms

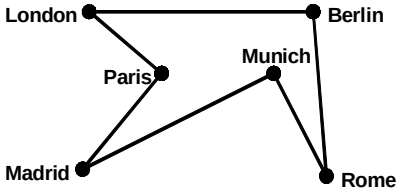
In the following lines two approximation algorithms are presented. The first one determines a tour which cost is less than two times the cost of the optimal tour. The second one after Christofides is even better.
For the Approx−TSP−Tour a complete weighted graph G is given.



minimum spanning treeT

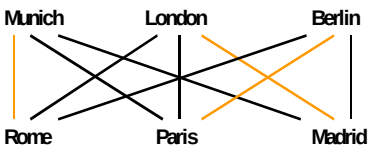


fullwalk of T

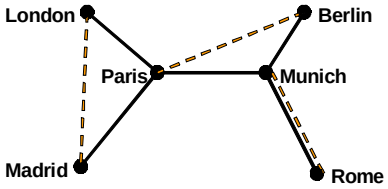


Approx−TSP−Tour

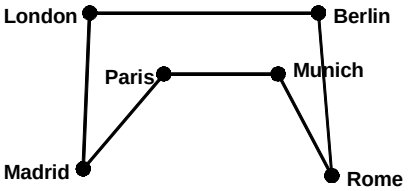
In the first step the algorithm finds the minimum spanning tree T of G e.g. after Kruskal. A full walk W of T visits many vertices of G more often than once as every edge appears twice. Because of the triangle inequality a preorder walk is possible and the result is an approximation tour H.
The complexity of this algorithm is O(|V|log|V|).
Why the cost for this tour is lower than two times the cost of the optimal tour H* is shown in the following proof.
Proof: The cost of the minimum spanning tree T is lower than the cost of the optimal tour H*, because when one deletes the edge of highest weight of H* the rest is either T or a tree of higher cost than T $\Rightarrow$ cost(T) $\leq$ cost(H*). The cost of the fullwalk W is 2*cost(T) as every edge is visited twice $\Rightarrow$ cost(W) $\leq$ 2*cost(H*). Because of the triangle inequality the cost of the preorder walk H is smaller than the cost of W $\Rightarrow$ cost(H) $\leq$ 2*cost(H*).
Although this is a quite satisfactory solution there exists a better one.The approximation algorithm of Christofides finds a tour which cost is maximal 3/2 higher than the cost of the optimal tour.



minimum weight matching M



combined graph M



Approx−TSP−Tour

This algorithm also starts with the minimum spanning tree T of the complete weighted graph G. But then there is made a minimum weight matching for all the vertices of T of odd degree. After that the minimum spanning tree is combined with the matching to a graph M. Now all the vertices have even degree and it is possible to find an Eulerian circuit E which contains all the edges and all the vertices. Because of the triangle inequality it is then practicable to delete the vertices which appear twice. The result is a tour H which cost is at the most 3/2 times bigger than the cost of the optimal tour.

The reason for that is that the cost(T) $\leq$ cost(H*) and the cost of the minimum weight matching is smaller than ½−times the cost of the optimal tour H* $\Rightarrow$ cost(H) $\leq$ 3/2*cost(H*).

Even though the algorithm of Christofides does not find the optimal tour it is at the moment the best way to solve the traveling salesman problem.

Finally some areas where the TSP and the Hamiltonian graphs are of interest.

Applications

The solution of the TSP is used for the design of computer systems where a given number of pins has to be connected in the shortest possible way.

In the Job−Sequencing−Problem the TSP is used in connection with a machine which needs a certain time to be prepared for doing the next job. Since time is money it is very important to find the least time consuming sequence of the jobs to be done at this machine.

Then the TSP is helpful for route collectors of coins e.g. from pay phones or for postmen emptying mail−boxes to find the shortest way to do their job.

Even for less serious problems as the knight's tour problem a solution can be found. The aim is to visit each field of a chess board exactly once by moving two fields in one direction and one field in the crossing direction (as the horse). The result is then a Hamiltonian circuit.

Bibliography

1. Brandstädt, Graphen und Algorithmen, Teubner, Stuttgart, 1994
   good first overview especially for Eulerian graphs
2. G. Chartand; O. R. Oellermann, Applied and algorithmic graph theory, McGraw−Hill, New York, 1993
   very informative, lots of details and applications
3. A. Aho; J. E. Hopcroft; J. D. Ullman, Data Structures and Algorithms, Addison−Wesley Publishing Company, Reading Massachusetts, 1983
   p.323f: introduction of the TSP (good and easy to understand)
   p.331ff: TSP with backtracking
   p.338ff: TSP with k−opting
   } too many details
4. R. Sedgewick, Algorithms, Addison−Wesley, 1983
   p.621f: introduction to the problem of exhaustive search
   p.630f: approximation algorithm for the TSP (too short)
   rest: not so interesting for a short overview
5. T. H. Cormen; C. E. Leiserson; R.L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts, 1990
   very good explanation of the Approx−TSP−Tour with triangle inequality
6. F. P. Preparata; M. I. Shamos, Computational Geometry, Springer, New York, 1988
   details of both approximation algorithms for the TSP
7. Duden „Informatik“, Dudenverlag, Mannheim a.o., 1993
   helpful as an overview and for examples