Paths in Graphs

Denis TRYSTRAM

Lecture notes **Maths for Computer Science** – *MOSIG 1 – 2017*

1 Preliminaries

1.1 Motivation et definitions

Graphs are a very natural mathematical structure used in Computer Science (and many other fields). A graph is defined as a finite set of vertices, linked according to a given *relation*. Formally, we define graph G as the pair (V, E). V is the (finite) set of vertices. The set of edges E represents the relation between the pairs of vertices. Usually, a graph is represented by a matrix $A_{i,j} = 1$ if vertex i is linked with vertex j otherwise $A_{i,j} = 0$. This matrix is called the *adjacency matrix* since it reflects the adjacency relations between vertices. Graphs may be weighted, in this case we simply associate an integer to each edge (similarly, we may also consider weights on the vertices).

Graphs are characterized by several notions which tell us about its structure: **degree** of a vertex (number of adjacent vertices denoted by $\delta(x)$ for $x \in V$), **diameter** (maximum distance between any pairs of vertices denoted by D) and **chromatic number** (minimum number of colors for coloring the graph, where any adjacent vertices are associated with different colors). This last characteristic is hard to compute while both others are easy to obtain (in polynomial time).

There is a duality between vertices and edges in graphs. We will investigate in this lecture some well-known path problems for the view point of both edges and vertices. We refer naturally to the *hamming* distance (defined as the minimum number of links to cross for reaching a vertex from another one for non-weighted graphs) and to the natural distance obtained by the minimum sum of weights of the crossed edges over the paths between two vertices.

Definition.
We define the *connected relation* between any two vertices x and y as the existence of a path from x to y.
A connected graph is a graph whose vertices are all connected.

The connected relation is an *equivalence relation* (reflexive, symmetric and transitve) and the *quotient graph* is the graph composed of its connected components.

1.2 Study of particular graphs

Let us start by studying structural properties for some particular regular graphs (*i.e.* those whose degree is equal on each vertex).

Definitions.

- A cycle C_n of order n is a graph where each vertex i is linked with $i-1$ and $i+1$ modulo n.
- The complete graph K_n of order n is the graph where each vertex is linked with all the others.
- An hypercube of rank k is defined recursively as follows:

  H_0 is reduced to one vertex.

  H_k is obtained by linking every pair of relative vertices of two H_{k-1}.

  H_2 and H_3 are the two classical square and cube.

Let us compute for all these graphs their number of edges, degree and diameter.

Both first graphs are very simple. They have the same number of edges. The first graph (the cycle) has a constant degree and its diameter is linear in the order of the graph while the second has a linear degree and a constant diameter. The hypercube is an intermediate graph with logarithmic degree and diameter. More precisely:

- C_n has a degree 2, n edges and a diameter of $\lfloor \frac{n}{2} \rfloor$.
- K_n has a degree $\frac{n(n-1)}{2}$, n edges and a diameter equal to 1.
- Hypercube H_k has $n = 2^k$ vertices (they double at each successive ranks). Its vertices have a degree $k = log_2(n)$ since an edge is added to each vertex at each successive rank.

  The number of edges is obtained by writing a recurrence equation is rather simple: The graph at rank $k+1$ is obtained by two copies of H_k plus 2^k edges for linking each relative vertex, thus, $N_{k+1} = 2 \times N_k + 2^k$ with $N_1 = 0$. We obtain: $N_k = k \times 2^{k-1}$.

  The diameter can be computed easily with the natural encoding of the hypercube using Gray codes: each vertex is encoded in a binary notation using $O(log_2(n))$ bits and its adjacent vertices are obtained by complementing each bit one after the other. Thus, there is a path from any vertex in crossing at most $log_2(n)$ other vertices.

There exist other graphs with a constant degree and low diameters (like the *de Bruijn* graph whose diameter is in also $og_2(n)$.

2 Eulerian paths

2.1 Existence

Let us first consider the problem of finding paths (or cycles) which visits all the edges of a given graph G exactly once. This problem of determining if such a tour (called *eulerian cycle*) exists is one of the oldest problem in the field of Operational Research (it was introduced in 1736). It has been studied by the famous swiss mathematician Leonard Euler for the specific case of a tour going through all the bridges of the Koenigsberg town. The graph is called *eulerian* if it admits such an eulerian cycle. This problem can be characterized by the following proposition.

Proposition. A connected graph is eulerian *iff* all its vertices have an even degree.

Proof (existence). Let us first establish three basic claims that will be useful.

Claim 1. if all the degree are even (and not nul) then there exists a cycle.

Claim 2. A tour is an union of disjoint cycles.

Claim 3. if we remove a cycle in a tour then the degrees remain even.

The necessary condition of the proposition is straightforward. Let us prove the sufficient condition.

By contradiction, let us assume that all the vertices are even and there is no tour that contains all the edges. Let consider a tour with a maximum number of edges. If we remove its edges, from Claim 3 the remaining edges are even. Then, from Claim 1, there exists a cycle within these remaining edges (say Γ). The contradiction comes from Claim 2 since the union of the maximal tour plus the cycle Γ is another tour which contains more edges than the initial one.

This proof can be adapted in a constructive way and thus, leads to an algorithm. It is as follows:
Proof 2 (constructive). By induction on the number of edges.

- The basis case is simple to verify for $m = 2$ (where two vertices linked by two edges correspond to the cycle of minimal length).
- Let consider a connected graph with $m + 1$ edges where all its vertices have an even degree. Let assume that the property holds for connected

graphs of even vertices with k edges ($k \leq m$), which means there exist eulerian tours in these sub-graphs.

From Claim 1, there exists a cycle (let denote it by Γ) and consider the sub-graph of G without the edges of Γ: $G' = (V - \Gamma, E')$. By induction hypothesis, there exists an eulerian $\mathcal{C}_i$ in each connected component of G' (removing the edges of a cycle did not change the parity of the vertices). The eulerian tour of G is obtained by the concatenation of pieces of Γ and the successive $\mathcal{C}_i$.

2.2 Chinese postman problem

Let us now discuss the problem of determining a cycle that contains all the edges when there exist some odd vertices. From the previous section, we know that there is no eulerian cycle in this case and thus, any solution should duplicate some edges. The problem is to duplicate the minimum. This problem is known as the *chinese postman* and it is described below (in a french equivalent version).

A postman moved recently from Grenoble to a small village in the country side. He asked himself how to organize his daily tour by bike for distributing the letters in the shortest possible time. The director of the post office gives him the map and fortunately, the postman had some old souvenir of previous lectures in Graph Theory. The tour starts from the post office and of course, the postman has to go through every roads for distributing the letters before coming back. The underlying graph is $G = (V, E)$ where V is the (finite) set of cross points and E is the set of the links between the cross roads weighted by the distances.

This problem may be formulated mathematically in term of eulerian cycles. Intuitively, the basic idea is to duplicate some edges that are carefully chosen in order to use the previous construction of an eulerian tours that will help the postman to determine the optimal tour (of minimal length) using some simple mathematical properties.

Proposition. In any graph $G = (V, E)$ the number of vertices of odd degree is even.

Proof. First, let remark that the sum of all degrees is even (more precisely, $\Sigma_{x \in V} \delta(x) = 2|E|$ since each edge counts exactly for two vertices – its extremities).

$\Sigma_{x \in V} \delta(x) = \Sigma_{x \in V_{even}} \delta(x) + \Sigma_{x \in V_{odd}} \delta(x)$.

As $\Sigma_{x \in V_{even}} \delta(x)$ is obviously even, $\Sigma_{x \in V_{odd}} \delta(x)$ should also be even.

Thus, the number of odd vertices is even.

Alternatively, this result can be proved by applying the Fubini's principle

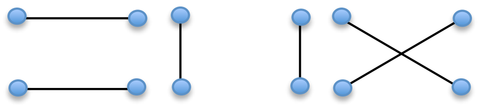

Figure 1: The 3 perfect matching ($k = 2$).

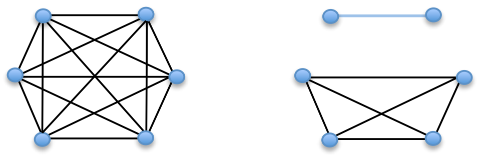

Figure 2: Perfect matching ($k = 3$).

using the adjacency matrix.

As there exists a path between any pair of vertices of odd degree in V_{odd}, we consider the complete graph whose vertices are the odd degree vertices weighting the edges with the shortest paths (denoted by K_{odd}). Computing the shortest paths is a classical problem, which can be solved in polynomial time (for instance by the well-known Dikjstra's algorithm).

Then, it is possible to make a correspondence between the optimal solution of the postman problem and a perfect matching of minimal weight in K_{odd}. Recall that a matching is a set of edges without common vertices. It is perfect if it has the maximum number of edges.

Proposition. The number of perfect matchings in a complete graph grows exponentially with n.

Proof. by recurrence on $n = 2k$, let denote the number of perfect matching by N_k.

- For $k = 1$, there is only one perfect matching $N_1 = 1$.
- For $k = 2$, there are 3 different perfect matchings $N_2 = 3$.
- For $k = 3$, there are 5 possibilities for a vertex to choose an edge $(2k - 1)$ which each leads to 3 perfect matchings $N_k = (2k - 1).N_{k-1}$.

The main steps of the algorithm for determining the optimal tour are the following:

- Build the complete graph with the odd vertices and compute its weight by the shortest paths.

- Compute the perfect matching of minimal weight.
- Duplicate all the edges along the paths of this matching.
- Determine an eulerian tour in this new graph.

The optimality of this algorithm comes from the fact that the duplicated edges are the minimum. Finally, all the vertices of the new graph are even since the degree of the odd vertices in G is augmented by 1 (extremities of the paths) and the other even vertices which are intermediate vertices of the paths remain even.

3 Hamiltonian paths

Let us now turn to the problem of paths going through the vertices of a given graph. There are two types of Hamiltonian cycles: for any graphs (structural) and the weighted variants for complete graphs.

Definition. An hamiltonian cycle is a tour that visits all vertices exactly once.

3.1 Tournaments

Suppose that n players compete in a round-robin tournament. Thus, for every pair of players-teams i and j, either i beats j or else j beats i. Interpreting the results of such a round-robin tournament can be problematic because there might be all kind of cycles where x beats y, y beats z and z beats x. Graph theory provides at least a partial solution to this problem. The results of a round-robin tournament can be represented with a tournament graph defined as follows.

Definition. A tournament is a directed graph in which the vertices represent players and the edges indicate the outcomes of the games. In particular, a (directed) edge from i to j indicates that player i defeated j.

In a round-robin tournament, every pair of players has a match to play. Thus, in a tournament graph there is either an edge from i to j or an edge from j to i for every pair of vertices i and j. A directed Hamiltonian path is a directed tour that visits every vertex exactly once.

We can show that in every round-robin tournament, there exists a ranking of the players such that each player loses to the player ranked one position higher. In other words:

Proposition. Every tournament graph contains a directed Hamiltonian path.

Proof.
The proof is by induction.

- **Basis.** A tournament with 2 vertices is obviously hamiltonian.
- Let $P(G, n)$ be the proposition that every tournament graph G with n vertices contains a directed Hamiltonian path. Let consider a graph of order $n + 1$, the idea is to isolate an arbitrary vertex (let denote it by x_0) and to construct a hamiltonian path using directed Hamiltonian paths in dimension n.

  Let consider a directed Hamiltonian paths in dimension n. Every vertex is linked with x_0 by in- or out- coming edges,. as the graph has

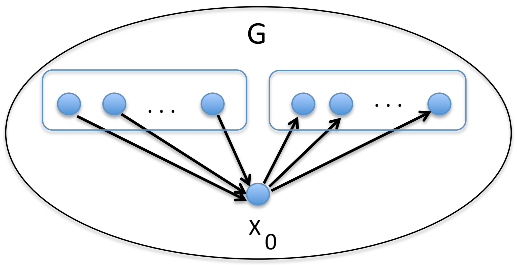

Figure 3: Principle of the construction of a directed Hamiltonian path in a tournament.

no circuits, these edges are ordered like in figure 3. The new hamiltonian path is obtained by ordering the vertices of the first (in-coming) vertices (which is a directed complete graph) by the hamiltonian path, then going through x_0 on the second group of out-coming edges, ordered similarly by the hamiltonian path.

3.2 Traveling salesman problem

This problem called TSP corresponds to determine a minimal hamiltonian cycle in a weighted complete graphs K_n. Obviously, K_n is hamiltonian (it exists $n!$ hamiltonian paths), thus, the question here is to determine the minimum one.

TSP is classical in Operational Research. Let us consider a salesman who wants to organize the visit of his clients as best as possible. It consists in visiting them in various cities with his vehicle. Of course, he must go in every cities and his objective is to minimize the total distance done with his vehicle. The only information he has is the list of the cities and a map with all inter-cities distances. We assume any Euclidian distance (for instance the weights correspond to number of kilometers between two cities). More formally, the input of the problem is a weighted matrix with an infinite weight on the diagonal.

Proposition. Christophides algorithm is a $\frac{3}{2}$-approximation.

Let us construct an efficient solution for this problem. It is well-known that TSP is a hard problem, that means we can not expect a polynomial time algorithm which solves exactly the problem. Let us construct a good solution (not too far from the optimal) in polynomial time proposed. It proceeds in three phases.

Phase 1. Determine a minimal weight spanning tree T^*. A spanning tree of G is a tree (connected graph with no cycle) with the same set of vertices as G. Let us denote by ω the weight of a graph G (sum of the weights on its edges). Determining a minimal weight spanning tree is easy in polynomial time.

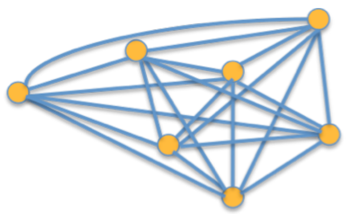

Figure 4: Example of an instance of TSP.

ω_{T^*} is a lower bound of the value of the optimal tour.

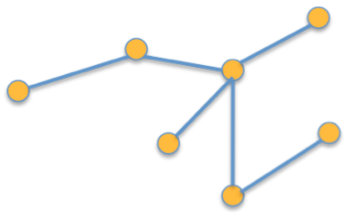

Figure 5: Construction of an optimal spanning Tree.

Phase 2. Consider now the set V_{odd} of the vertices of T^* whose degrees are odd.

We proved in the previous section that the cardinality of V_{odd} is even. Let us construct now the perfect matching C^* of minimum weight between these vertices in V_{odd}.

$2\omega_{C^*}$ is a lower bound of the value of the optimal tour.

Phase 3. Let us now consider the graph $G' = T^* \cup C^*$.

All the vertices of G' have an even degree.

We are now going to transform this graph in the following way: We replace iteratively some edges by shortcuts.

While it exists a vertex of degree greater than 4, we remove two of these consecutive edges and replace them by the opposite edge of this triangle without disconnecting the graph.

This process leads to a feasible tour.

Such transformations do not increase the total weight.

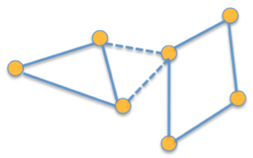

Figure 6: Final touch: replacing edges by short cuts (first possibility). This solution disconnects the graph.

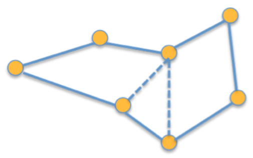

Figure 7: Final touch: replacing edges by short cuts (second possibility).

Finally we deduce that the value of such a tour is lower than 3/2 of the optimal tour.