Paths in Graphs

Denis TRYSTRAM

Lecture notes **Maths for Computer Science** – *MOSIG 1 – 2018*

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 their structures: **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 index** (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).

We investigate in this lecture some well-known path problems for the view point of both edges and vertices. There is a duality between vertices and edges in graphs.

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 transitive) 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 $\lfloor \frac{n}{2} \rfloor$, its chromatic index is equal to 2 for even values of n and 3 for odd n.
- K_n has a degree $\frac{n(n-1)}{2}$, n edges and a diameter equal to 1, its chromatic index is equal to n.
- Hypercube H_k has $n = 2^k$ vertices (their numbers double at each successive ranks from k to $k+1$). Its vertices have all 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 rather simple recurrence equation: 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 $N_0 = 0$. We obtain finally: $N_k = k \times 2^{k-1}$.

  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 $log_2(n)$.

1.3 Some preliminary results

We describe in this section some graph problems that will be useful for further proofs. We will use these results as a black box.

- **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 the degrees is even (more precisely, $\Sigma_{x \in V} \delta(x) = 2|E|$ since each edge counts exactly for two vertices – its extremities).

  Alternatively, this result can be proved by applying the Fubini's principle using the adjacency matrix. The two ways of counting the non-zero elements are by rows (giving the sum of degrees) and globally .

  Now, decompose this sum into even and odd degrees.

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

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

  Thus, the number of odd vertices is even.

- **Perfect matchings.** A *matching* is a set of edges that have no vertices in common.

  It is *perfect* if its vertices are all belonging to an edge (thus, the number of vertices is even and the cardinality of the matching is exactly half of this number).

  Proposition. The number of perfect matchings in a graph of order $n = 2k$ grows exponentially with k.

  Proof. by recurrence on k, let denote the number of perfect matchings by N_k.

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

  Thus, N_k is the product of the k first odd numbers.

  However, determining a perfect matching of minimal weight in a weighted graph can be obtained in polynomial time (using the Hungarian assignment algorithm).

- **Minimum Spanning Trees.** (MST)

  A spanning tree of a weighted graph G is a tree whose edges span all the vertices of G.

  Determining a minimal spanning tree can be done is linear time (in the number of edges).

  There are mainly two ways for constructing such a MST, each one emphasizes a different propriety of the MST, namely, avoid cycles and minimize the span. In both cases, the edges are sorted in increasing order of weights. More precisely, the first one constructs a subtree which partially spans the graph by adding at each step the minimum neighboring edge while the other add successively the edges of minimal weights that do not create a cycle.

- **Shortest path problems.**

  Determining the shortest (or longest) path can be done in polynomial time.

2 Path problems

The main interest of graphs is to determine *paths*, i.e. how vertices are connected to each other (determine if they are connected, find the shortest or the longest paths, count the number of edge disjoint paths, etc.).

There are two types of problems: those which concern the edges and those which concern the vertices. In this sense, these problems are *dual*. However, surprisingly, determining a *tour* (i.e. a cycle) that visits exactly once each edge is an easy problem while determining a tour that visits exactly once each vertex is hard.

3 Eulerian cycles

3.1 Existence

Let us first consider the problem of determining a cycle which visits all the edges of a given (undirected) graph $G = (V, E)$ 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. The solution of this problem is characterized by the following proposition.

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

We distinguish below two types of proofs based on the same claims, one is simply the existence while the other one provides a solution. Let us first establish three basic claims (elementary facts) that will be useful.

Claim 1. if all the degrees are even (and not null) 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 focus on the sufficient condition.

Proof 1 (existence).
By contradiction, let us assume that all the vertices of a connected graph 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, it remains some edges and from Claim 3 they 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 described by its successive vertices) and consider the sub-graph of G without the edges of Γ: $G' = (V - \Gamma, E')$. By induction hypothesis, there exists an eulerian cycle $\mathcal{C}_i$ in each connected component of G' (from Claim 3). The eulerian tour of G is obtained by the concatenation of pieces of Γ and the eulerian cycles in the successive $\mathcal{C}_i$.

3.2 Chinese postman problem

Let us now present the more general problem of determining a cycle that contains all the edges in any graph, in particular when there exist some odd vertices. From the previous section, we know that there is no eulerian cycle in this case and thus, any feasible 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 to his office. 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.

Fig. 1 presents an example of the chinese postman problem.

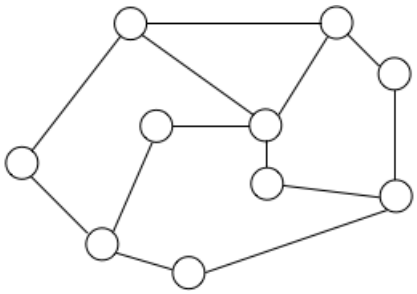

Figure 1: Example of an instance of the chinese postman with 10 cross vertices.

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 tour of section 2.1 that will help the postman to determine the optimal tour (of minimal length) using some simple mathematical properties.

First, we know that there is an even number of odd vertices. Considering the previous instance of the postman problem, there are 4 such vertices (represented in grey in Fig. 2).

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

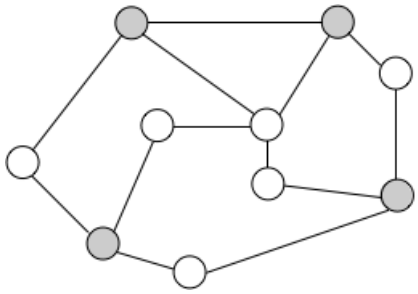

Figure 2: The 4 vertices with an odd degree in the previous instance.

vertices weighting the edges with the shortest paths (denoted by K_{odd}). As we mentioned in the preliminary properties, computing the shortest paths is a classical problem, which can be solved in polynomial time.

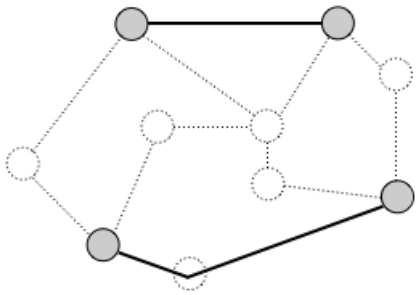

Figure 3: The minimum weight perfect matching labelled by the shortest distances between the vertices of V_{odd}.

Then, it is possible to do the correspondence between the optimal solution of the postman problem and a perfect matching of minimal weight in K_{odd} by duplicating the edges of the minimal perfect matching.

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

- Consider the complete graph with the odd vertices and compute its weight by the shortest paths. Compute a perfect matching of minimal weight between these vertices.
- Duplicate all the edges along the paths of this matching.
- Determine an eulerian tour in this new graph with even degrees.

The optimality of this algorithm comes from the fact that the duplicated edges are the minimum possible ones. Finally, all the vertices of the new

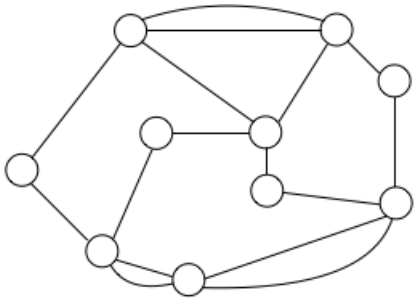

Figure 4: Final step: adding the edges of the minimum perfect matching.

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.

4 Hamiltonian tours

Let us now turn to the problem of cycles going through the vertices of a given graph instead of edges.

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

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

TSP is a classical problem in Operational Research. Let us consider a saleswoman who wants to organize the visit of her clients as best as possible. It consists in visiting them in various cities with her vehicle. Of course, she must go in every city and her objective is to minimize the total distance done in the tour. The only information she has is the list of the cities and a map with all inter-cities distances. We assume a *Euclidian distance* (for instance the weights correspond to number of kilometers between two cities). This property means that the straight line is always the minimum distance, or in other words that the distance between two vertices/cities is larger if the path is going through any other vertex.

Proposition. The solution of Christofides algorithm is not worse than $\frac{3}{2}$ time the optimal tour.

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, unless $\mathcal{P} = \mathcal{NP}$. Fig. 9 presents an example of euclidian TSP with $n = 7$ cities.

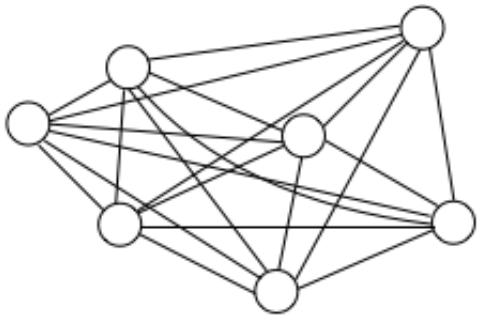

Figure 5: Example of an instance of TSP with 7 cities.

Let us construct a good solution (not *too far* from the optimal) in polynomial time. Let us denote by ω_G the weight of graph G (i.e. the sum of the weights on its edges). The Chritofides algorithm proceeds in three steps.

Step 1. Determine a minimal weight spanning tree T^*. As we recalled in the preliminaries, a minimal weight spanning tree can be determined in polynomial time.

ω_{T^*} is a lower bound of the value of the optimal tour ω_{H^*}. Indeed, H^* is a cycle, then, removing any edge in H^* leads to a chain, which is a particular spanning tree. As T^* is the minimal spanning tree, we have: $\omega_{T^*} \leq \omega_{H^*}$.

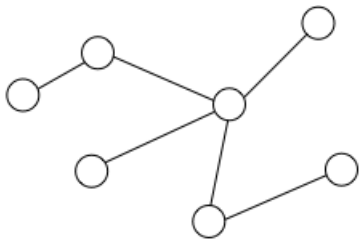

Figure 6: Construction of an optimal spanning Tree T^*.

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

We proved in the preliminary properties that the cardinality of V_{odd} is even.

Let us construct the perfect matching C^* of minimum weight between the vertices in V_{odd}. Fig. 7 shows all possible perfect matchings on the previous example, the optimal one (with minimal weight) is represented in bold.

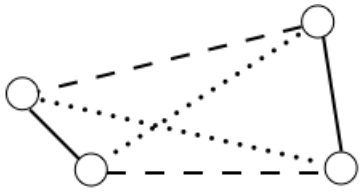

Figure 7: The three different perfect matchings on the set of four vertices of odd degree. The optimal one is represented in bold (the two others are in dashed and dot lines).

Fig. 8 illustrates the graph obtained by considering the edges of both T^* and C^*.

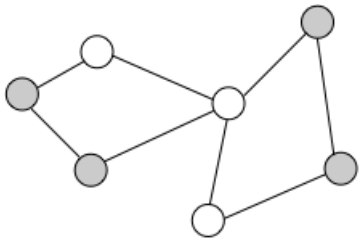

Figure 8: Adding the optimal perfect matching C^* to the minimal spanning tree T^*.

Let us now determine a lower bound of the optimal tour H^* (represented in Fig. 9).

$2\omega_{C^*}$ is a lower bound of the value of the optimal tour ($\omega_{C^*} \leq \frac{1}{2}\omega_{H^*}$). Indeed, consider first the perfect matching C^*. As its vertices belong to H^*, ω_{C*} is lower than the piece of hamiltonian tour contained between these vertices because of the euclidian property (see Fig. 10). Similarly for the *complementary* perfect matching C (Fig. 11). Thus, the weight of the cycle formed by the concatenation of both perfect matchings is lower than the hamiltonian tour $\omega_{C^* \bigcup C} \leq \omega_{H^*}$. Moreover, as C^* is the minimum perfect matching, we have $\omega_{C^*} \leq \omega_C$, this concludes the proof.

Step 3. All the vertices of $T^* \cup C^*$ have an even degree since we added an edge of C^* to every odd degree vertices of T^*. We are now going to transform this graph by replacing iteratively the high degree vertices by shortcuts, which decreases the degree until reaching 2.

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. There are $2k$ ways to remove 2 edges and replace them by the triangle edge. Some of them disconnect the graph and

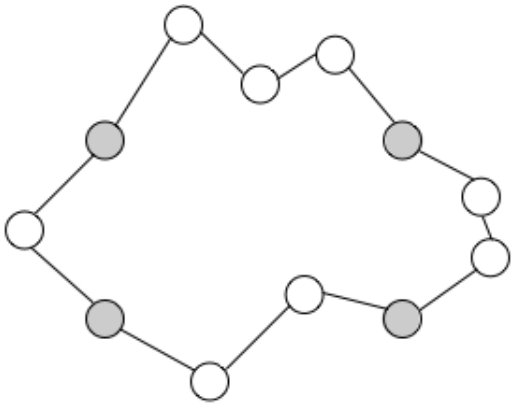

Figure 9: An optimal hamiltonian cycle H^*.

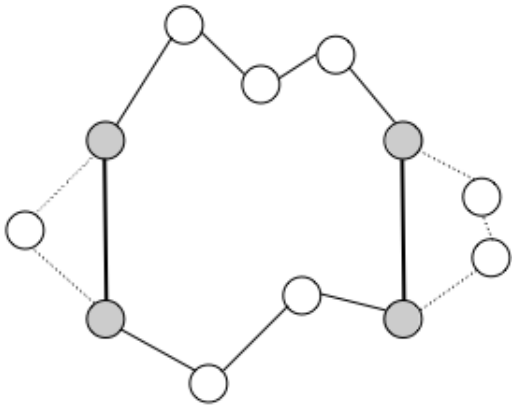

Figure 10: Perfect matching C^* between the vertices of odd degrees.

thus, must be avoided. Fig. 12 shows such a transformation on the previous example, Fig. 13 shows a valid transformation.

This process leads to a feasible tour. Such transformations do not increase the total weight.

Finally, as $\omega_{T^*} \leq \omega_{H^*}$ and $\omega_{C^*} \leq 1/2\omega_{H^*}$, we deduce that the value of such a tour is lower than $3/2\omega_{H^*}$.

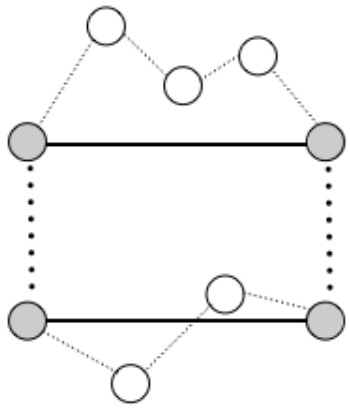

Figure 11: Cycle $C^* \bigcup C$ (in dashed and bold).

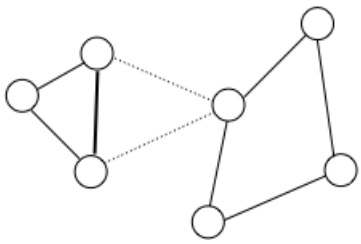

Figure 12: Reduction of the degree in $T^* \bigcup C^*$, disconnected solution.

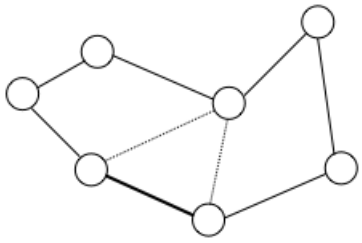

Figure 13: Reduction of the degree in $T^* \bigcup C^*$, connected solution.