Math 3322: Graph Theory[1] Mikhail Lavrov

Lecture 17: Eulerian tours and cycle decompositions

October 15, 2024 *Kennesaw State University*

1 Two problems

In this lecture and the next, we will consider two very similar-sounding problems.

1. An **Eulerian tour** in a graph G is a closed walk that uses every edge of G exactly once.
2. A **Hamiltonian cycle** in a graph G is a cycle[2] that visits every vertex of G exactly once before ending up back where it started.

We call a graph Eulerian if it has an Eulerian tour, and we call a graph Hamiltonian if it has a Hamiltonian cycle.

It will turn out that it's easy to determine if a graph is Eulerian. On the other hand, figuring out if a graph is Hamiltonian or not is hard. We will see a few situations in which we can answer that question, but a lot of the time, we will not know the answer except by brute force.

We will consider the Eulerian tour problem for multigraphs. I will put the word "multigraph" in statements of theorems to remind you, but you should remember that throughout this lecture, everything we call a graph could be a multigraph. Of course, everything we say will also be true for graphs: graphs are just multigraphs that happen not to have any loops or parallel edges.

See the practice problems for a generalization of the Eulerian tour problem to directed graphs: the idea is almost exactly the same, but a few details need to be checked.

2 Finding Eulerian tours

The following result has a very short proof:

Lemma 2.1. *If a multigraph G has an Eulerian tour, then every vertex in G has an even degree.*

Proof. Every time the Eulerian tour enters a vertex v, it must also leave, using two edges out of v. If, after doing this k times, all the edges out of v are used up, then v must have degree $2k$, which is even.

(We might also use a loop to leave v and immediately come back. This only uses one edge, not two—but it's an edge that contributes degree 2 to v, so the logic remains the same.) □

Another way to say this: the condition "every vertex in G has a even degree" is **necessary** for the graph to be Eulerian. No graph can be Eulerian without this condition. We would also like to

[1] This document comes from the Math 3322 course webpage: http://facultyweb.kennesaw.edu/mlavrov/courses/3322-fall-2024.php

[2] we could have said "closed walk" to match the previous problem, but if your closed walk does not revisit vertices, then it's a cycle.

know if this condition is **sufficient** for the graph to be Eulerian: is checking this condition enough to know that G has an Eulerian tour?

Thinking about necessary and sufficient conditions is the key to understanding the difference between the Eulerian tour problem and the Hamiltonian cycle problem.

- For the Eulerian tour problem, we will be able to state a condition which is both necessary and sufficient (and which can be checked without doing very much work).
- For the Hamiltonian cycle problem, we will find some necessary conditions and some that are sufficient. However, graph theory knows no simple necessary *and* sufficient condition.

Back in the world of Eulerian graphs, before we prove the big result we want, we will take a detour[3] to discuss cycle decompositions.

2.1 Cycle decompositions

A **decomposition** of a graph G is a partition of the edges of G: we split up the graph into many pieces that share no edges. We can think of the pieces as sets of edges (in which case their union as sets is $E(G)$) or as subgraphs (in which case their union as graphs is G).

In particular, a **cycle decomposition** of G is a set of cycles in G such that every edge of G is in exactly one cycle.

Lemma 2.2. *A multigraph G has a cycle decomposition if and only if every vertex of G has an even degree.*

Proof. The "only if" direction is due to the fact that every cycle contributes 2 to the degree of the vertices it passes through. (This remains true for the "unusual" cycles present in multigraphs: cycles with just 1 or 2 edges.) Therefore if G has a cycle decomposition, the degree of a vertex that lies on k of the cycles is $2k$, which is even.

To prove the "if" direction, we will use strong induction on the number of edges in G. As our base case: if G has 0 edges, then the empty set is a cycle decomposition: it includes every edge exactly once!

Suppose G has $m > 0$ edges. Pick any component of G that has an edge in it. That component has minimum degree 2 (because all the degrees in it are even and positive), so we know it contains a cycle C.

In $G - C$ (the graph we get by deleting the edges of C, leaving the vertices as they are), all the degrees are still even: the degrees of vertices of C went down by 2, and the rest are unchanged. Also, $G - C$ has fewer than m edges (since C contained at least one edge), so by induction, $G - C$ has a cycle decomposition.

Add C to that cycle decomposition, and we get a cycle decomposition of G; by induction, this exist for any number of vertices. □

[3] Heh-heh.

This proof also suggests an algorithm for finding a cycle decomposition. Simply find any cycle (which we can do by aimless walking around until we revisit a vertex), set it aside, and repeat with the remainder of the graph until we are out of edges.

2.2 Gluing cycles together

The cycle decomposition lemma lets us prove the big result we actually wanted:

Theorem 2.3. *A multigraph G is Eulerian if and only if it is connected (except possibly for isolated vertices) and every vertex of G has an even degree.*

Proof. We know that the even-degree condition is necessary by Lemma 2.1. We also know that an Eulerian graph cannot have two edges in different connected components: in that case, a tour would never be able to visit them both. So it remains to show that together, these two conditions are sufficient.

Assume that G has only one connected component that is not an isolated vertex, and every vertex of G has an even degree. The first thing we'll do is use Lemma 2.2 to find a cycle decomposition of G.

Now, we'll build up our Eulerian tour step by step. This is a proof by algorithm. Throughout this process, we will always have two things:

- a "partial tour" PT, which is a closed walk that walks each edge at most once, but might miss some edges.
- a set $\mathcal{U}$ of "unprocessed cycles": a cycle decomposition of just the edges that are **not** part of PT.

Initially, we'll take PT to be one of the cycles in our cycle decomposition, and $\mathcal{U}$ to be all the other cycles.

Then, one by one, we'll remove a cycle from $\mathcal{U}$, and splice it into PT. To make this possible, it's important to find a cycle in $\mathcal{U}$ that contains one of the vertices visited by PT. There must always be such a cycle: otherwise, we'd get two connected components in G! There would be no edge between two disjoint sets: the set of vertices visited by PT, and the set of vertices that are part of a cycle in $\mathcal{U}$. Both of these sets contain some edges. That would contradict our assumption about G.

So now suppose we've found such a cycle C. Say that PT is the closed walk $(w_0, w_1, \ldots, w_t, w_0)$, C is the closed walk $(v_0, v_1, \ldots, v_k, v_0)$, and these share vertex $w_i = v_j$. Then replace PT by the closed walk

$$(w_0, w_1, \ldots, w_{i-1}, w_i = v_j, v_{j+1}, \ldots, v_k, v_0, \ldots, v_{j-1}, v_j = w_i, w_{i+1}, \ldots, w_t, w_0).$$

Essentially, insert C into PT at a point where the shared vertex is visited.

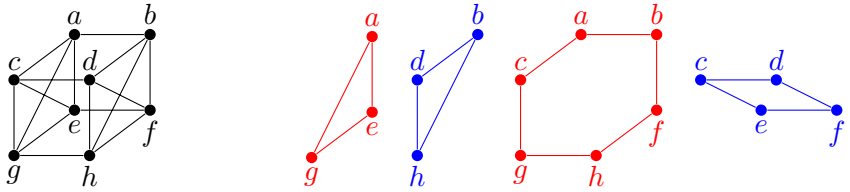
Note that after this process, PT walks all the edges which were previously traversed by C. Therefore after we do this (and remove C from $\mathcal{U}$), it's still true that $\mathcal{U}$ is a cycle decomposition of the edges not part of PT. Also, it's still true that PT uses every edge at most once: none of the new edges were previously used by PT.

Repeat this until $\mathcal{U}$ is empty; then PT is an Eulerian tour. □

The algorithm we get here is more or less the algorithm known as **Hierholzer's algorithm** for finding Eulerian tours.

2.3 An example

Let's use this method to find an Eulerian tour in the following modified version of a cube graph (on the left):



Our first step is to find a cycle decomposition. To do this, just take any cycle in the graph, remove it, and keep going with the rest. One possible decomposition is into the four cycles on the right.

(It is not the shortest one; there's actually a decomposition into two cycles of length 8. If we wanted to find the Eulerian tour as quickly as possible, the shorter decomposition would be better. But here, I want to demonstrate the method in action.)

Now let's splice them together into a tour.

1. Initially, our partial tour PT is just the first cycle: (a, g, e, a).
2. Next, we want to add a cycle that shares a vertex with PT. We can choose either the third or the fourth cycle (the second one doesn't work). Let's take the fourth cycle: (c, d, f, e, c).

   The only vertex this shares with PT is e. So we follow PT from a to e, then walk around the cycle starting at e and returning to e, then follow the remainder of PT. This gives us:

   $$(a, g, e, c, d, f, e, a).$$

3. At this point, PT has lots of vertices, so we can choose any cycle to add. Let's add the second cycle: (b, d, h, b). We can splice this into PT at vertex d, getting:

   $$(a, g, e, c, d, h, b, d, f, e, a).$$

4. Finally, we add the only remaining cycle: (a, b, f, h, g, c, a). Since PT and this cycle both start and end at a, combining them is particularly easy. Just follow one then the other, getting

   $$(a, b, f, h, g, c, a, g, e, c, d, h, b, d, f, e, a).$$

The result is an Eulerian tour; you can check this by tracing it out, and seeing that you get all the edges of the original graph G without doubling back over any edge.

3 Practice problems

1. Refresh your memory by going back to Lecture 7 and looking up the definition of Harary graphs. Then, use our algorithm to find an Eulerian tour of $H_{10,4}$.

   (There are other ways to find an Eulerian tour, including trial and error. But if you're here doing the completely optional practice problems, you might as well practice the algorithm. The "ingredient graphs" should help you find a cycle decomposition.)

2. Find an Eulerian tour of the 4-dimensional hypercube graph, Q_4.

3. In the complete graph K_7, every vertex has degree 6, so by Lemma 2.2, K_7 has a cycle decomposition.

   (a) Suppose we wanted to decompose K_7 into as few cycles as possible. How many cycles would we want to use, and of what lengths? Give an example of such a decomposition.

   (b) Suppose we wanted to decompose K_7 into as many cycles as possible. How many cycles would we want to use, and of what lengths? Give an example of such a decomposition.

4. Let G be a multigraph; suppose that G is connected, every vertex in G has an even degree, *and* G has an even number of edges. Prove that it's possible to color half of G's edges red and the other half blue in such a way that every vertex v is the endpoint of $\frac{1}{2}\deg(v)$ red edges and $\frac{1}{2}\deg(v)$ blue edges.

5. Without using any of our results on matchings, but using only the previous problem, prove that for all $k \geq 0$, every 2^k-regular bipartite graph (or multigraph) has a perfect matching.

6. **Generalizing to directed graphs.**

   When looking for Eulerian tours in directed graphs, almost nothing changes. There is still a condition on degrees that guarantees a cycle decomposition, and we can still glue cycles together to get closed walks.

   This exercise is mainly challenging because it asks you to use definitions you've only learned in the previous lecture. That makes it especially useful!

   (a) Prove that if G is a directed graph with an Eulerian tour, then G is strongly connected, and every vertex v satisfies $\deg^+(v) = \deg^-(v)$. (The indegree of every vertex is equal to the outdegree.)

   (b) Prove that if G is a directed graph with $\deg^+(v) = \deg^-(v)$ for all vertices v, then G contains a cycle. (This is the only ingredient of our proof today that's a bit different.)

   (c) Verify that all the other parts of our algorithm work with no modification.

7. **Generalizing to arbitrary walks.**

   Sometimes, rather than finding an Eulerian tour (a closed walk) we consider the problem of finding *any* walk that visits all the edges once. (This may be called an **Eulerian trail**.)

   (a) Suppose that a multigraph G has a $v - w$ walk (where $v \neq w$) that visits all the edges exactly once. What can you say about the degrees of vertices in G: which are even, and which are odd?

(b) We can prove a necessary and sufficient condition on Eulerian trails by *reducing* the problem to the problem we've already solved, rather tha prove anything from scratch. This saves effort.

Suppose that G is a connected multigraph satisfying the degree condition you found. We can modify G to create a new graph G' with two properties:

- G' is connected, and every vertex of G' has even degree.
- When we find an Eulerian tour of G', it will create an Eulerian trail in G.

Then our theorem about Eulerian tours does the rest of the work: it guarantees that G' does in fact have an Eulerian tour.

Explain how to find G', and why it has these properties.