CSL851: Algorithmic Graph Theory **Semester I 2013-2014**

Lecture 6: August 12

Lecturer: Naveen Garg *Scribes: Shubham Gupta*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

This lecture discusses some important applications of Min-cost matching, and introduces the concept of Ear decomposition of undirected graphs.

6.1 Applications of Min-cost Matching

6.1.1 T-joins

Definition 6.1 *Given a graph $G = (V, E)$, a cost function $C : E \to R$ and a set of vertices $T \subset V$ such that $|T|$ is even, a T-join is defined to be a set of edges $E' \subset E$ such that*

- $\forall v \in V \setminus T, deg_{E'}(v)$ *is even, and*
- $\forall v \in T, deg_{E'}(v)$ *is odd.*

Any such set of edges can be decomposed into a set of paths joining the vertices of T.

(a) Original graph (b) T-join of (a)

Figure 6.1: T-join

For instance, one can start with any vertex in T (which will have at least one incident edge in E', its degree being odd). If the adjacent vertex belongs to T, we have found a path between two of T's vertices, and we're done. If not, then this vertex will have at least one more edge incident to it (its degree being even), and we can continue this way until we encounter a vertex belonging to T.

After we find a path between a pair of T's vertices, we can remove them from T and the edges which form this path can be removed from the graph. Note that the parity of degree of the internal vetices of this path remains unchanged by this removal operation. This process can then be continued until T becomes empty.

Figure 6.1 illustrates this property of T-joins. The red vertices in are the vertices in T, and the red edges in (b) form a T-join for T.

6.1.1.1 Finding a min-cost T-join

The algorithm for finding the min-cost matching in a graph can be utilized in finding the min-cost T-join for the given graph and a set of vertices T.

Lemma 6.2 *Consider a graph $G = (V, E)$ and $T \subset V$. Let $H = (T, F)$ be a complete weighted graph on T, with a weight function $l : T \times T \to \mathbb{R}^+$ defined as l(u,v) = weight of the min-weight path between u and v in the original graph. Then a min-cost matching in H, when expanded in G, forms a min-cost T-join in G.*

Proof: It is obvious from the construction that the resulting set of edges obtained in G will connect all vertices of T in a pair-wise manner. For this set to be a valid T-join, the degrees of vertices in V must satisfy the parity requirement as specified in the definition of T-join. We're instead going to prove that all the paths which form the set are edge disjoint. It is easy to see that edge disjointedness is sufficient to prove the parity requirement (in our case, all the paths are between a distinct pair of vertices, every vertex of T occurs as a path's endpoint exactly once, and no other vertex in V is the endpoint in any of the paths. Also, the internal vertices of a path may belong to T or not. Thus, if the paths are edge-disjoint, vertices of T will have an odd degree, and those outside T will have an even degree).

Figure 6.2

To prove path-disjointedness, refer to Figure 6.2. Let us assume that the vertices marked 1,2,3 and 4 belong to T, and the path joining vertices 1 and 3 has an edge in common with the path joining 2 and 4. This implies that 1 was matched with 3 and 2 was matched with 4 in the min-cost matching of H.

Consider an alternative matching of these four vertices, as shown in Figure 6.2(b). It shows a path between vertices 1 and 2, and vertices 3 and 4. The weight of edges joining these pairs of vertices in H would be less than or equal to the weight of these paths, since H was formed by shortest paths beteen every pair of vertices. Thus we have an alternative matching of these four vertices which has a cost less than or equal to the min-cost matching of these vertices, which implies that the matching formed by joining the pairs (1,2)

and (3,4) is also a min-cost matching, paths of which don't have common edges, which is a contradiction of the fact that the matching in Figure 6.2(a) is a min-cost matching. ■

6.1.1.2 Applications of min-cost T-join

- **Chinese Postman Problem**

  The Chinese Postman problem involves finding a shortest closed path or circuit that visits every edge of a connected undirected graph. The tour might not be a simple cycle - an edge might have to be visited multiple times. Figure 6.3 illustrates this with an example. Figure 6.3 (b) shows the optimal path solving the Chinese Postman problem for the graph in Figure 6.3 (a) - by starting from the vertex numbered 1, and traversing the edges in the specified order. Note that the edge between vertices numbered 2 and 4 is traversed twice.

  Equivalently, given a non-Eulerian graph, it is concerned with finding the smallest number of edges which need to be doubled up so that the resulting multigraph is Eulerian. It is easy to see that no optimum solution would visit an edge more than 2 times, because then the 2 edges from the multigraph could be removed, and the degree of all the vertices would still be even.

  This problem can be solved by using the algorithm for finding the min-cost T-join. The set T

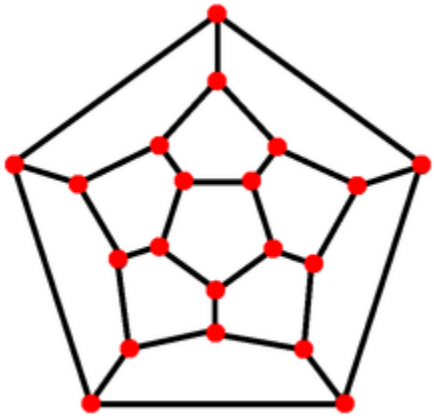
Figure 6.3

  can be defined to be consisting of all the odd degree vertices, and we can then find the min-cost T-join of the graph. The resulting subset of edges obtained is the minimal subset of edges which need to be doubled up so that the resulting multigraph is Eulerian. This approach works because it selects the minimum number of edges to be doubled up so that the parity of all the vertices in T, which are the odd-degree vertices of the original graph, changes, while that of all the other vertices remains the same, finally resulting in a multigraph in which degree of all the vertices is even.

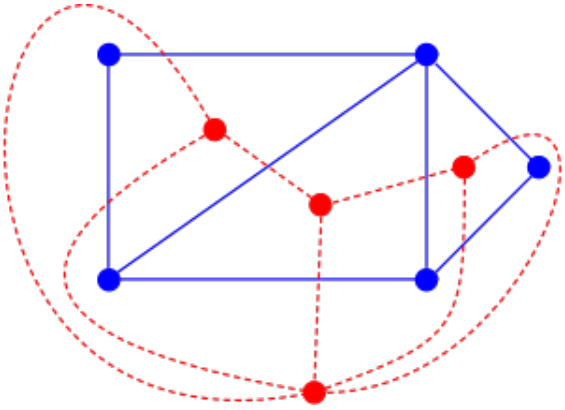
- **Max-cut in Planar Graphs**

  Finding the max-cut in a general graph is an NP-hard problem. For unweighted graphs, it is equivalent to finding a cut having maximum number of edges across it, i.e. to minimize the number of edges in the same partition, which is same as minimizing the number of edges which have to be removed so that the graph becomes bipartite.

  A *planar graph* is one which can be drawn so that none of the edges cross each other. A planar graph can be partitioned into a number of faces, which are regions bounded by edges and have no edges inside. Also, faces form a cycle basis, which means that any cycle in the graph can be expressed as the sum (XOR) of these faces.



(a) A planar graph, partitioned into faces



(b) A planar graph and its dual

Figure 6.4

From the discussion above, it is clear that finding the max-cut in planar graphs is equivalent to finding the minimum number of edges that need to be removed from the graph so that all the odd faces in the given planar graph become even.

This problem can be solved by considering the dual of the given planar graph.

Definition 6.3 *The dual of a planar graph G is a graph that has a vertex corresponding to each face of G, and an edge joining two neighbouring faces for each edge in G.*

For instance, in Figure 6.4 (b), the red gaph is a dual of the blue graph. Note that if the boundary separating two faces consists of multiple edges, then the dual will have multiple edges between the vertices corresponding to those two faces, the number of edges being same in both the cases. Hence for every edge in G, there is a corresponding edge in the dual. For weighted graphs, weights of corresponding edges should be kept equal.

It is clear that odd faces in the planar graph correspond to vertices with odd degree in the dual graph. Consider a set T consisting of all the odd degree vertices in the dual graph. Find the min-cost T-join with respect to T in the dual graph. Similar to the solution of the Chinese Postman problem, this set give us the minimum 'cost' of removing edges so that all the vertices of the resulting graph have an even degree. Removing the corresponding edges in the original graph would give us a planar graph in which all faces are even, which is what we required.

6.2 Ear Decomposition

Definition 6.4 *An ear decomposition of a graph $G = (V, E)$ is a collection of paths $P_0, P_1, P_2, \ldots, P_n$, where P_0 is a simple cycle in G, and $P_i (i \geq 1)$ is a path in G both of whose end points are in $P_0 \cup P_1 \cup \ldots \cup P_{i-1}$.*

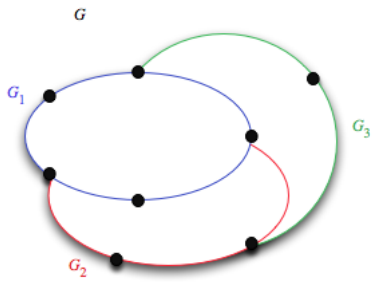


Figure 6.5: An ear decomposition of a graph consisting of 3 ears

Lemma 6.5 *A graph $G = (V, E)$ has an ear decomposition if and only if it is bridgeless.*

Proof: A graph being bridgeless implies that it doesn't disconnect by removal of any of its edges. It is equivalent to saying that every edge of the graph is part of a cycle. If a graph is ear-decompressible, it is clear by the definition of ear decomposition that removal of any edge won't disconnect the graph, since every edge is part of a cycle. This implies that the graph is bridgeless. Conversely, let the graph be bridgeless. To start with, choose any edge of the graph, and complete the cycle it is a part of. This cycle forms our P_0. For the ith step, consider any edge at least one of whose endpoints is in $P_0 \cup P_1 \cup \ldots \cup P_n$. This edge is part of some cycle too, which will intersect the previously formed ears at some point. Complete this cycle until this point and proceed likewise until the entire graph is covered. Hence every bridgeless graph will have an ear decomposition. ■