CSE 347 Analysis of Algorithms February 18, 2019

Lecture 10 Summary

Instructor: Brendan Juba

1. Maximum Bipartite Matching: Given a bipartite graph, find a matching of maximum size.
   (a) We saw an algorithm that runs in time $O(|V||E|)$ that creates a graph with a flow of value at least the maximum matching size.
   (b) Since at most one unit of flow enters/leaves each vertex on the unit capacity edges we create, at most one edge (with flow 1) has each vertex as an endpoint: therefore, the set of edges with flow 1 is a matching.
   (c) Since the value of the flow is, in particular, equal to the amount of flow crossing from the left vertices to the right vertices, and the flow is integer valued, the number of edges in this matching equals the value of the flow.
   (d) Therefore, the matching we return is a maximum matching.
2. Image Segmentation: we wish to separate an image into a "foreground" set of pixels and a "background" set of pixels. We want to maximize the total likelihood of the foreground and background assignments (p_i and $1 - p_i$ respectively for each pixel i), minus the (given) cost of separating a neighboring pair of pixels c_{ij} for neighbors i and j.
   (a) Equivalently, we are choosing a partition (A, B) to minimize $\sum_{i \in A}(1 - p_i) + \sum_{i \in B} p_i + \sum_{\text{neighbors } i \in A, j \in B} c_{ij}$.
   (b) We can reduce this problem to minimum cut: we create a vertex for each pixel, and add vertices s and t. We connect s to each pixel i with capacity p_i and connect i to t with capacity $(1 - p_i)$. We then connect each neighboring i to j with capacity c_{ij}.
   (c) s-t cuts in this graph (A, B) correspond to segmentations $(A \setminus \{s\}, B \setminus \{t\})$, where the capacity of the cut is identical to the equivalent minimization objective. Therefore a minimum cut (removing s and t) is an optimal segmentation.
   (d) The size of the graph is proportional to the size of the image. If we have rational costs, we can either clear the denominators and use Ford-Fulkerson, or we can use the more advanced scaling algorithm (see book) to solve this min-cut problem in polynomial time.
3. Reductions allow us to relate the solvability of computational problems, regardless of whether or not we actually have algorithms for solving either problem.
   (a) In general, we can say that "A reduces to B" if there is an algorithm that solves A in polynomial time, given the ability to solve instances of B in a single step (where we still have to pay for writing the input and reading from the output of B).
   (b) What we actually showed: Maximum Bipartite Matching reduces to Maximum Integer Flow and Image Segmentation reduces to Minimum Cut (under this formal definition).
   (c) If A reduces to B and B has a polynomial time algorithm, then A also has a polynomial time algorithm (by plugging in the algorithm for B in place of the magic instruction). In this sense, A can only be "easier" than B to solve.