Lowest-Cost (Not So) Simple Path

November 11, 2016

Imagine a weighted, directed graph G where edge weights may be positive, negative, or zero. We will consider the problem of finding the lowest-cost simple path between a source node s and terminal node t in such a graph. We'll call this problem GENSHORT for "general shortest path". (Recall that a **simple** path is a path with no vertex repeated, i.e., with no cycles.)

(Recall that the Bellman-Ford Algorithm—as presented in our text—finds the shortest path from any start vertex in the graph to a single terminal vertex t. It proceeds using dynamic programming using a table parameterized by which node is being considered as s and the maximum number of edges in the path from s to t. The first column (where the maximum number of edges is 0) has ∞ for all nodes except t itself and 0 for t. On each iteration, it updates each row s in the next column based on the lowest-cost path of all those that go from s to some node u (in one edge) and then from u to t using the already-computed value in the previous column.)

1. **Very briefly** explain why the Bellman-Ford algorithm cannot in general be used to solve GENSHORT.

2. Give a small instance of GENSHORT on which the Bellman-Ford algorithm **will** find the lowest-cost simple path from s to t. Be sure to indicate what that lowest-cost simple path is.

3. Here is a proposed reduction from GENSHORT to the problem of finding the lowest-cost simple path between a source node s and terminal node t in a weighted, directed graph with **only non-negative edge weights**:

   Reduction: Given the graph G that may contain negative edge weights, find the edge with minimum weight w_{min} (by scanning through all edges) and subtract w_{min} from the weight of every edge to create graph G'. In G' the minimum weight edge has weight 0, and no edge has negative weight. Find the lowest-cost simple path between s and t in G' (i.e., call on the solution to the underlying problem), and then return this list of vertices as the lowest-cost simple path in the original graph. (Of course, the edges connecting the vertices have different weights in G, but it's still the same path.)

   Give a small instance of GENSHORT on which this reduction does **not** produce the optimal solution. Indicate the solution produced by the reduction and the optimal solution.

1 NP-Completeness

In this part, we will consider a decision-variant of GENSHORT. In this variant, we add a number k to the format of an instance. The solution to the instance is YES if a simple path from s to t exists with cost less than or equal to k; otherwise, the solution is NO.

1. Prove—by reducing from the HAMPATH problem to GENSHORT—that GENSHORT is NP-hard. (Note: HAMPATH is NP-complete.) *Hint:* it may help to add a couple of nodes to be s and t. When thinking about edges to and from those nodes, consider that you can have zero-weight edges.

2. Prove that the decision version of GENSHORT is in NP by showing it is "efficiently certifiable". First, select a certificate. (Think of how you would describe the solution to the **original** version of GENSHORT.) Then, show how to prove in time polynomial in the size of the decision-variant GENSHORT instance that the answer to the decision problem is YES given such a certificate. (A decision-variant GENSHORT instance is a graph plus one extra number; think of its size as $O(n+m)$ as usual for graphs.)

(This isn't required, but you might want to work through how you could solve the original variant of GENSHORT using a polynomial number of calls to the decision-variant.)