Parma University, Math. Department

Algorithms and Data Structures APSP with negative weights

Marco Pellegrini

Spring 2002, Lecture 21

Summary:

- This lecture:
  Johnson's algorithm to compute APSP on a weighted directed graph with possibly negative weights.

Johnson's algorithm.

- Gets as input a directed graph G=(V,E), a function w:E →R. Produces a matrix d[1..n][1..n] where d[i,j] is the shortest path between node i and j, or returns a flag: "negative cycle" if it finds one.
- It is very fast on sparse graphs: we can make it run in time $O(|V||E| + |V|^2 \log |V|)$. which is better than Floyd-Marshall when the graph is not dense.
- It uses Dijkstra and Bellman-Ford as subroutines.
- It uses a "reweighting" technique.

main idea

- The main idea is to change the weight function from w:E → R to another weight function w':E → R.
- What we want is that:
  (1) A path p that is a shortest path between its end-points in G with w, is still a shortest path in G with w'.
  (2) For every edge e in E, $w'(e) \geq 0$.
- Suppose that we find such function w'. What we can do now is to run |V| times Dijkstra using each node in turn as source node to compute all the shortest paths (with weights w').
- So we get the right paths (but the wrong shortest distance) so we have to do a correction taking into account the difference w(e)-w'(e) on each edge.

Satisfying property (1).

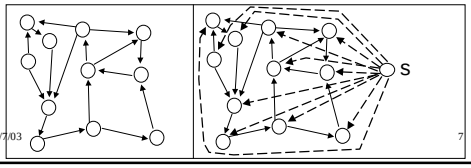
- Let h be any function h:V → R, that associate a node in V with a real number.
- Define w'(u,v) = w(u,v) +h(u) - h(v).
- We prove now that a shortest path p using w, is still a shortest path using w' and vice versa.
- Take any path $p=(v_0,\ldots,v_k)$, its w'-weight is:
  $w'(p) = \Sigma_{i=1..k}\, w'(v_{i-1},v_i)=$
  $= \Sigma_{i=1..k}\, [w(v_{i-1},v_i) + h(v_{i-1}) - h(v_i)] =$
  $= w(p) +h(v_0) - h(v_k)$.
- In the last passage we have cancelled lots of terms except the value of h for the first and the last node in the path.

Proof of property (1) cntd.

- Assume that a path p between i and j is shortest with w (having weight w(p), but we have a different path q shortest in w' (of weigh w'(q) < w'(p)).
- We derive a contradiction in this way:
  w(q) +h(i) - h(j) = w'(q) < w'(p) = w(p) +h(i)-h(j).
- Simplifying we get w(q) < w(p) which contradicts our choice of p a shortest path in w between i and j.
- Also, we show that cycles do not change weight by using w' instead of w. In particular no negative cycles can appear in G by switching from w to w'.
- w'(c) = w(c) + h(i) -h(j) = w(c), because in a cycle the first and last node are the same.

Satisfying Property (2)

- Now we look for a function h:V → R that makes w' positive on every edge.
- The idea is to use an SSSP algorithm to find an interesting function to associate to the nodes.
- For this idea to work we must make G reachable from a node without changing the picture.
- We add a node s to V: V'=V+{s}. We add an edge from s to any v in V. The weight of each new edge is 0.



Property (2) cntd.

- So by adding s to V, and (s,u) for every u in V to E we do not generate any new path between any two nodes i and j in G (because we have only outgoing edges form s). Also the new graph has negative cycles if and only if the old one had negative cycles.
- Now we run Bellman-Ford on the modified graph starting from s. If we find a negative cycle we stop. Otherwise at the end d[v] = D[s,v] and moreover from the fact that the last loop did not return false we know that d[v] < d[u] + w(u,v) for any edge (u,v).
- So define h=d.
- We get w'(u,v) = w(u,v) + d[u] - d[v] and we are set. We have found our re-weighting function w'.

Johnson's algorithm

We pass the weight functions as a parameter to subroutines.

```
Johnson(G)
    compute G' where V'=V+{s}, E'=E + {(s,u)|u in V}
    IF Bellman-Ford(G',s,w) = false
      THEN print "negative cycle"
      ELSE FOR each v in V DO
             { h[v] = D(s,v) %as computed by Bellman-Ford
           END FOR
           FOR each (u,v) in E DO
           { w'(u,v) = w(u,v) +h[u]-h[v]
           END FOR
           FOR each u in V DO
           { run Dijkstra(G,u,w')  %fill in matrix D'
           END FOR
           FOR each u and v in V DO
           { D[u,v] := D'[u,v] +h[v] - h[u]
           END FOR
           Return D
```


Observations

- Modifying the graph G takes time O(|V|).
- Bellman-Ford runs in time O(|V||E|).
- Computing h and w' takes time O(|V| + |E|)
- Implementing Dijkstra with Fibonacci heaps we take time O(|V|(|V| log |V| + |E|)) for all executions of Dijkstra.
- The last loop takes time $O(|V|^2)$.
- So in total Johnson's algorithm takes time $O(|V|^2 \log V + |E||V|)$.

Summary of algorithms

SSSP problem

	all positive weights	some negative weights
dense	Dijkstra + array	Bellman-Ford
sparse	Dijkstra + Fibonacci	Bellman-Ford

APSP problem

	all positive weights	some negative weights
dense	Floyd-Warshall	Floyd-Warshall
sparse	repeated Dijkstra + Fibonacci	Johnson

Homework 1

- CLR 26.3-3 If w(u,v)>0 for any edge (u,v) in G what is the relation between the weight function w and the weight function w' computed in Johnson's algorithm?

Homework 2

CLR 25.3. Arbitrage is the use of discrepancy between currencies to turn a profit. Example if 1 dollar buys 0.7 Pounds, 1 pound 9.5 francs and 1 franc 0.16 dollars. Starting with 1 dollar and changing dollar-pound-franc-dollar we get 1x0.7x9.5x0.16=1.064.

Suppose that you have currencies $c_1,c_2,..c_n$ and the exchange rates matrix R[i,j] where one unit of c_i buys R[i,j] units of c_j. Find efficiently whether there is a sequence of currencies so that:

$R[i_1,i_2]$ x $R[i_2,i_3]$ x… x $R[i_k,i_1] > 1$.

Conclusions

- We have seen an efficient algorithm by Johnson to find shortest paths in sparse directed graph with possibly negative weights.
- Now you have a whole range of techniques to handle shortest path problems.