Lecture notes for "Analysis of Algorithms": **Computing shortest paths and detecting negative cycles**

Lecturer: *Uri Zwick* *

May 5, 2009

Abstract

We consider the problem of computing shortest paths in a weighted directed graph whose edges have positive or negative *lengths* associated with them. Some shortest paths are not well defined if the graph contains *negative* cycles. We shortly review the classical Bellman-Ford algorithm for finding shortest paths and/or negative cycles, which runs in $O(mn)$ time. We next describe a *scaling* algorithm of Goldberg that runs in $O(m\sqrt{n}\log N)$ when the length $\ell(e)$ of each edge e is an integer satisfying $\ell(e) \geq -N$.

1 Preliminaries

Let $G = (V, E)$ be a directed graph, and let $\ell : E \to \mathbb{R}$ be a *length* function defined on its edges. If $P = \langle u_0, u_1, \ldots, u_k \rangle$ is a path in G, we define its length $\ell(P)$ to be $\sum_{i=1}^{k} \ell(u_{i-1}, u_i)$. The length a cycle C in G is defined similarly. If $u, v \in V$, we define the *distance* $\delta(u, v)$ from u to v in the graph to be the smallest length of a path from u to v in the graph. There are two cases in which the distance $\delta(u, v)$ may not be well defined: (i) There is no directed path from u to v in the graph, in this case we define $\delta(u, v) = +\infty$. (ii) The graph G contains a *negative cycle*, i.e., a cycle C for which $\ell(C) < 0$, such that there is a directed path in G from u to a vertex w on C, and a directed path from w to v. It is clear that in this case there are paths from u to v with arbitrarily small length. In such a case we define $\delta(u, v) = -\infty$.

If $\ell : E \to \mathbb{R}^+$, i.e., $\ell(e) \geq 0$, for every $e \in E$, and $s \in V$ is vertex, then the distances $\delta(s, v)$ from s to all $v \in V$, and a *tree of shortest paths* from s to all the other vertices in the graph can be computed in $O(m + n \log n)$ time using the classical algorithm of Dijkstra [Dij59], implemented using Fibonacci heaps [FT87] or a similar data structure. For a detailed description of Dijkstra's

*School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E–mail: zwick@cs.tau.ac.il

algorithm and Fibonacci heaps, see [CLRS01]. Dijkstra's algorithm fails, however, in the presence of negative edges.

2 The Bellman-Ford algorithm

Distances and shortest paths in a graph with positive and negative edge weights, or negative cycles can be found, at a much higher cost, using the classical Bellman-Ford algorithm [Bel58, FF62].

The algorithm maintains an array $d[\cdot]$ of the best upper bounds currently available on the distances from s to all the other vertices of the graph. Initially $d[s] = 0$ and $d[v] = +\infty$, for every $v \in V - \{s\}$. The algorithm then performs n passes. In each pass it considers all the edges of the graph. For each edge $(u, v) \in E$, if $d[u] + \ell(u, v) < d[v]$, then a new upper bound on $\delta(s, v)$ can be obtained. The algorithm then sets $d[v] \leftarrow d[u] + \ell(u, v)$ and $pred[v] \leftarrow u$. The algorithm is given in Figure 1.

```
for every v ∈ V
    d[v] ← +∞
    pred[v] ← null
d[s] ← 0
for i ← 1 to n
    for every (u, v) ∈ E
        if d[u] + ℓ(u, v) < d[v] then
            d[v] ← d[u] + ℓ(u, v)
            pred[v] ← u
```

Figure 1: The classical Bellman-Ford algorithm.

It is easy to show that if there are no negative cycles in the graph, then at the end of the algorithm $d[v] = \delta(s, v)$, for every $v \in V$. (More specifically, we argue that after the k-th iteration, $d[v] \leq \delta_k(s, v)$, where $\delta_k(s, v)$ is the length of the shortest path from s to v that uses at most k edges.) Furthermore, the pointers $pred[v]$ define in this case a tree for shortest paths from s. Also, the graph contains a negative cycle accessible from s if and only if at least one of the values $d[v]$, for some $v \in V$, changes in the last iteration. The complexity of the algorithm is clearly $O(mn)$. Almost 50 years after it was first formulated, this is still the best strongly polynomial bound for the problem!

Several modifications can be made to improve the performance of the algorithm in practice. For example, the edges emanating from a vertex u should be examined in a given iteration only if $d[u]$ changed since the last time these edges were examined. Also, there are ways of aborting the algorithm as soon a negative cycle is found. For these and other issues, consult [CGR96].

3 Vertex potentials and reduced length

Let $p : V \to \mathbb{R}$ be an arbitrary function that assigns *potentials* to the vertices of $G = (V, E)$. The *reduced edge lengths* $\ell_p(u, v)$ with respect to p are defined as follows:

$$\ell_p(u, v) \;=\; \ell(u, v) + p(u) - p(v) \quad , \quad (u, v) \in E \; .$$

The usefulness of this concept comes from the following simple lemma:

Lemma 3.1 *(i)If $P = \langle u_0, u_1, \ldots, u_k \rangle$ is a path in G, then $\ell_p(P) = \ell(P) + p(u_0) - p(u_k)$.*
(ii) If $C = \langle u_0, u_1, \ldots, u_k, u_0 \rangle$ is a cycle in G, then $\ell_p(C) = \ell(C)$.

Proof: To prove *(i)* note that

$$\begin{aligned} \ell_p(P) &= \sum_{i=1}^{k} \ell_p(u_{i-1}, u_i) = \sum_{i=1}^{k} \Big(\ell(u_{i-1}, u_i) + p(u_{i-1}) - p(u_i) \Big) \\ &= \Big(\sum_{i=1}^{k} \ell(u_{i-1}, u_i) \Big) + p(u_0) - p(u_k) = \ell(P) + p(u_0) - p(u_k) \; . \end{aligned}$$

Using *(i)* we now get

$$\ell_p(C) \;=\; \ell(C) + p(u_0) - p(u_0) \;=\; \ell(C) \; .$$

□

It follows that shortest paths with respect to the reduced length function ℓ_p are also shortest paths with respect to the original length function ℓ, and vice versa. Also, a graph has a negative cycle with respect to ℓ if and only if it has one with respect to ℓ_p.

Theorem 3.2 *A graph $G = (V, E)$ with a length function $\ell : V \to \mathbb{R}$ defined on its edges contains no negative cycles if and only if there is a potential function $p : V \to \mathbb{R}$ for which $\ell_p(u, v) = \ell(u, v) + p(u) - p(v) \geq 0$, for every $(u, v) \in E$.*

Proof: If there is a potential function p for which $\ell_p(u, v) = \ell(u, v) + p(u) - p(v) \geq 0$, for every $(u, v) \in E$, then there are clearly no negative cycles in G with respect to ℓ_p, and hence, by Lemma 3.1, there are also no negative cycles with respect to ℓ.

Suppose now that there are no negative cycles in G with respect to ℓ. Add a new vertex s to G and connect it with zero length edges to all the original vertices of G. As every vertex in now reachable from s, and as there are no negative cycles in the graph, the distances $\delta(s, v)$, for every $v \in V$, are well defined. We claim that $p(v) = \delta(s, v)$ is a potential function for which $\ell_p(u, v) = \ell(u, v) + p(u) - p(v) \geq 0$, for every $(u, v) \in E$. Indeed, if $(u, v) \in E$, then by the triangle inequality we get that $p(v) = \delta(s, v) \leq \delta(s, u) + \ell(u, v) = p(u) + \ell(u, v)$, as required. □

As an immediate corollary, we get an $O(mn+n^2 \log n)$ algorithm for solving the *All-Pairs Shortest-Paths* problem in graphs with positive and negative edge lengths but with no negative cycles. We add a new source vertex s, as in the proof of Theorem 3.2. We use the Bellman-Ford algorithm to compute $p(v) = \delta(s, v)$, for every $v \in V$. We then compute the reduced lengths $\ell_p(u, v) = \ell(u, v) + p(u) - p(v) \geq 0$, for every $(u, v) \in E$. As the reduced lengths are non-negative, we can use n invocations of Dijkstra's algorithm to compute all the distances in the graph with respect to ℓ_p. Finally, if $\delta_p(u, v)$ is the computed distance from u to v with respect to ℓ_p, we know that $\delta(u, v) = \delta_p(u, v) - p(u) + p(v)$, for every $u, v \in V$.

4 Scaling algorithms

Suppose now that $\ell : E \to \mathbb{Z}$ and $\ell(e) \geq -N$, for every $e \in E$. In other words, the edge lengths are now *integers* not smaller than $-N$. In this and the next section we describe an algorithm of Goldberg [Gol95] that finds either a negative cycle in G, or a potential function $p : V \to \mathbb{Z}$ for which $\ell_p(u, v) = \ell(u, v) + p(u) - p(v) \geq 0$, for every $(u, v) \in E$.

The algorithm uses a *scaling* approach. It defines a coarser version $\ell'(e) = \lceil \frac{\ell(e)}{2} \rceil$, for every $e \in E$, of the length function, recursively solves the problem for this length function ℓ', and then tries to adjust the solution obtained to get a valid solution for the original weight function ℓ.

If the algorithm finds a negative cycle with respect to ℓ', then this cycle is also a negative cycle with respect to ℓ and we are done.

If there are no negative cycles with respect to ℓ', the recursive call returns a potential function $p' : V \to \mathbb{Z}$ for which $\ell'(u, v) + p'(u) - p'(v) \geq 0$. As $\ell(u, v) \geq 2\lceil \frac{\ell(u,v)}{2} \rceil - 1 = 2\ell'(u, v) - 1$, we get that the reduced lengths with respect to the potential function $p(v) = 2p'(v)$, for every $v \in V$, satisfy

$$\begin{aligned} \ell_p(u, v) &= \ell(u, v) + p(u) - p(v) \\ &\geq (2\ell'(u, v) - 1) + 2p'(u) - 2p'(v) \ \geq \ -1 \ , \end{aligned}$$

for every $(u, v) \in E$. We are now faced with the problem of deciding whether G has a negative cycle with respect to ℓ_p or not. Note, however, that the problem is now 'simpler', as $\ell_p(u, v) \geq -1$, for every $(u, v) \in E$. In the next section we show that we can solve this problem in $O(m\sqrt{n})$ time. If the algorithm finds a negative cycle with respect to ℓ_p, then this cycle is also a negative cycle with respect to ℓ. If it finds a potential function $q : V \to \mathbb{Z}$ for which $(\ell_p)_q(u, v) = \ell_p(u, v) + q(u) - q(v) \geq 0$, for every $(u, v) \in E$, then it is easy to check that $\pi(u) = p(u) + q(u)$, for every $u \in V$, is a potential function for which $\ell_\pi(u, v) \geq 0$, for every $(u, v) \in E$.

The algorithm performs at most $\lceil \log N \rceil$ recursive calls before $\ell'(e) \geq -1$, for every $e \in E$, or a negative cycle is found, and its total running time is therefore $O(m\sqrt{n} \log N)$. (We assume that $N \geq 2$.)

5 The algorithm of Goldberg

In this section we describe an $O(m\sqrt{n})$-time algorithm for the case $\ell(e) \geq -1$, for every $e \in E$.

5.1 The graph G^-

The algorithm begins by constructing a subgraph $G^- = (V, E^-)$ of G, where

$$E^- = \{e \in E \mid \ell(e) \leq 0\} .$$

It then finds the strongly connected components of G^-. This can be easily done in $O(m+n)$ time. If a negative edge e, i.e., an edge $e \in E$ for which $\ell(e) < 0$, connects two vertices that belong to the same strongly connected components of G^-, a negative cycle is found. We can assume, therefore, that all the edges connecting vertices in the same strongly connected component of G^- are zero edges. We now *contract* each strongly connected component of G^-. To avoid cumbersome notation, we use G^- to refer to this contracted graph. Note that G^- is an *acyclic* graph.

Similarly, we use G to refer to the contracted version of the original graph. Note that G is *not* necessarily acyclic. It is easy to check that the contracted version of G contains a negative cycle if and only if the original version does. Similarly, a potential function for which the reduced lengths of all the edges in the contracted graph are non-negative can be easily extended to such a potential function for the original graph.

5.2 Mending negative vertices

We say that vertex $v \in V$ is *negative*, if there is a negative edge $(u, v) \in E$ entering it. We can mend a negative vertex v by lowering its potential by 1. In other words, if we define $p(v) = -1$ and $p(u) = 0$, for every $u \neq v$, then v is no longer negative with respect to ℓ_p. Unfortunately, new vertices may now become negative with respect to ℓ_p. Indeed, if $(v, w) \in E$ and $\ell(v, w) = 0$, then $\ell_p(v, w) = -1$, and w is now negative. To avoid the creation of new negative vertices, we lower the potential of all the vertices reachable from v in G^-. More precisely, let $R(v) = \{w \in V \mid v \rightsquigarrow w\}$, where we use $v \rightsquigarrow w$ to denote the fact that there is a directed path from v to w in G^-. (Note that $v \rightsquigarrow v$.) It is easy to check that after lowering the potential of all the vertices of $R(v)$ by 1, the vertex v is no longer negative, and no new vertices become negative. (We later prove a more general version of this claim.)

Computing the set $R(v)$ and lowering the potential of all the vertices in it may require $\Omega(m)$ time. As there may be $\Omega(n)$ negative vertices to mend, we have no improvement yet over the classic $O(mn)$-time algorithm. We thus aim at mending as many negative vertices at once. To that end, we show that if there are currently k negative vertices in the graph, then we always find $\sqrt{k}$ vertices that we can mend simultaneously in $O(m)$ time. We shall see below that this leads to the promised $O(m\sqrt{n})$-time algorithm.

5.3 Partitioning the graph into layers

Add a new source vertex s to G^- and connect it with zero length edges to all the vertices of G^-. Find the distances $\delta^-(s,v)$ from s to all the other vertices of G^-. This can be easily done in $O(m+n)$ time, in spite of the fact that G^- contains negative edges, as G^- is acyclic. Let

$$L_i \;=\; \{v \in V \mid \delta^-(s,v) = -i\}\ .$$

We refer to L_i as the i-th layer. It is easy to see that if $(u,v) \in E$, where $u \in L_i$, $v \in L_j$, and $\ell(u,v) = 0$, then $j \geq i$. Similarly, if $\ell(u,v) = -1$, the $j > i$.

Let k be the number of negative vertices currently in the graph. Let r be the largest integer for which $L_r \neq \phi$. If $r \leq \sqrt{k}$, then there is a layer L_i, with $i > 0$, that contains at least $\sqrt{k}$ negative vertices. (Note that L_0 does not contain negative vertices.) If $r > \sqrt{k}$, we get a directed path in G^- with at least $\sqrt{k}$ negative edges, and hence at least $\sqrt{k}$ negative vertices on it. (Take a shortest path from s to a vertex of L_r.)

(Remark: The same reasoning can be used to prove Dilworth's theorem: Any *partial order* $(R, \preceq)$ on k elements contains either a *chain* or an *anti-chain* of at least $\sqrt{k}$ elements. A chain is a totally ordered subset of R. An anti-chain is a subset of R in which every two elements are incomparable.)

5.4 Mending the negative vertices in a layer

We can mend all the negative vertices in L_i by lowering the potential of all the vertices in $L_{\geq i} = \cup_{j \geq i} L_j$ by 1. Indeed, let $p(u) = 0$, if $u \notin L_{\geq i}$, and $p(v) = -1$, if $v \in L_{\geq i}$. We claim that this potential function mendes all the negative vertices in L_i, does not create any new negative vertices and edges, and does not create any edge with reduced cost below -1. This follows from the following three simple observations:

1. If $\ell(u,v) > 0$, then as $p(u) - p(v) \geq -1$, we get that $\ell_p(u,v) \geq 0$.
2. If $\ell(u,v) \leq 0$, we have $p(u) - p(v) \geq 0$ and thus $\ell_p(u,v) \geq \ell(u,v)$. Indeed, if $u \in L_{\geq i}$ and $\ell(u,v) \leq 0$, we must also have $v \in L_{\geq i}$.
3. Finally, if $\ell(u,v) = -1$ and $v \in L_i$, then $u \notin L_{\geq i}$, and thus $\ell_p(u,v) = 0$.

Thus, all the negative vertices in a layer L_i can be easily mended in $O(m)$ time.

5.5 Mending negative vertices on a path

Let P be a directed path in G^- and let $w_1, w_2, \ldots, w_r$ be the vertices of the path with negative incoming path edges, in the order in which they appear on the path.

Define a new length function $\ell^+(e) = \max\{\ell(e), 0\}$. Add a new source s to G and add edges from s to all other vertices of G. Define the length of these edges as follows: $\ell^+(s, w_i) = r - i$, for $1 \leq i \leq r$, and $\ell^+(s, v) = r$, for all other vertices.

Compute the distances $\delta^+(v) = \delta^+(s, v)$ from s to all vertices in G with respect to the length function ℓ^+. As all edge lengths are non-negative integers and as all distances are at most $r \leq n$, this can be easily done in $O(m)$ time. Now use the distances computed as potentials, i.e., $p(u) = \delta^+(u)$, for every $u \in V$. We claim that if the graph contains no negative cycles, then the vertices $w_1, w_2, \ldots, w_r$ are mended, no new negative vertices are introduced, and no edge length drops below -1. This is again verified by considering three cases:

1. If $\ell(u, v) \geq 0$, then $\ell^+(u, v) = \ell(u, v)$. By the triangle inequality, $\delta^+(v) \leq \delta^+(u) + \ell(u, v)$, and thus $\ell_p(u, v) \geq 0$.

2. If $\ell(u, v) = -1$, then $\ell^+(u, v) = 0$. By the triangle inequality, $\delta^+(v) \leq \delta^+(u)$, and thus $\ell_p(u, v) \geq -1$.

3. Finally, suppose that $\ell(u, w_i) = -1$. If $\delta^+(u) > \delta^+(w_i)$, then $\ell_p(u, w_i) \geq 0$, and we are done. We next show that if $\delta^+(u) \leq \delta^+(w_i)$, then a negative cycle in the graph can be easily identified. As $\delta^+(w_i) \leq r - i$, the first edge on the shortest path from s to u must be to a vertex w_j, where $j \geq i$. Thus,

$$\delta^+(u) \;=\; (r - j) + \delta^+(w_j, u) \;\leq\; \delta^+(w_i) \;\leq\; r - i \;.$$

   It follows that $\delta^+(w_j, u) \leq j - i$. With respect to the original length function ℓ we thus have $\delta(w_i, w_j) \leq i - j$, $\delta(w_j, u) \leq j - i$ and $\delta(u, w_i) < 0$. Thus the cycle formed by the portion of P from w_i to w_j, the shortest path (with respect to ℓ^+) from w_j to u, and then the edge (u, w_i) is a negative cycle!

Thus we can mend all the negative vertices on a given path in G^- in $O(m)$ time.

5.6 The time complexity of the algorithm

Initially there are at most n negative vertices in the graph. Each iteration removes at least $\sqrt{k}$ negative vertices, where k is the current number of negative vertices. The question, therefore, is how many iterations of the function $k \to k - \lceil\sqrt{k}\rceil$ are needed before the number of negative vertices drops to 0. It is easy to check that if $k \leq \frac{1}{4}n^2$, then $k - \lceil\sqrt{k}\rceil \leq \frac{1}{4}(n-1)^2$. Thus, the total number of iterations required is $O(\sqrt{n})$.

As each iteration requires only $O(m)$ time, the total running time of the algorithm, for the case $\ell(e) \geq -1$, is $O(m\sqrt{n})$, as required.

References

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[CGR96] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: theory and experimental evaluation. *Mathematical Programming (Series A)*, 73(2):129–174, 1996.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, 2nd edition, 2001.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[FT87] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[Gol95] A.V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995.