Fast Shortest Paths Algorithms in the Presence of Few Negative Arcs

Domenico Cantone and Simone Faro

Università di Catania, Dipartimento di Matematica e Informatica
Viale Andrea Doria 6, I-95125 Catania, Italy
{cantone | faro}@dmi.unict.it

The *shortest paths problem* on weighted directed graphs is one of the basic network optimization problems. Its importance is mainly due to its applications in various areas, such as communication and transportation. Given a source node s in a weighted directed graph G, with n nodes and m arcs, the *single-source shortest path problem* (SSSP, for short) from s is the problem of finding the minimum weight paths from s to all other nodes of G. The *all-pairs shortest paths problem* (APSP, for short) consists in finding the minimum weight paths for each pair of nodes in G. In this paper we present hybrid algorithms for the SSSP and the APSP problems which are asymptotically fast when run on graphs with *few* negative weight arcs.

We begin by reviewing the relevant notations and terminology. A *directed graph* is represented as a pair $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V$ is a set of arcs such that E does not contain any self-loop of the form (v, v). In this context, we usually put $n = |V|$ and $m = |E|$. A *weight function* ω on $G = (V, E)$ is any real function $\omega : E \to \mathbb{R}$. A *path* in $G = (V, E)$ from u to v is any finite sequence $(v_0, v_1, \ldots, v_k)$ of nodes such that $v_0 = u$, $v_k = v$, and (v_i, v_{i+1}) is an arc of G, for $i = 0, 1, \ldots, k-1$. An arc (v_j, v_{j+1}) in a path $(v_0, v_1, \ldots, v_k)$ is *internal* if $0 < j < k-1$. An unspecified path from u to v will be also denoted by $(u \leadsto v)$. The weight function can be naturally extended over paths by setting $\omega(v_0, v_1, \ldots, v_k) = \sum_{i=0}^{k-1} \omega(v_i, v_{i+1})$. A *minimum weight path* (or shortest path) from u to v is a path in $G = (V, E)$ whose weight is minimum among all paths from u to v.
Provided that v is reachable from u and that no path from u to v goes through a negative weight cycle, a minimum weight path from u to v always exists; in such a case we denote by $\delta(u, v)$ the weight of a minimum path from u to v. If v is not reachable from u, we set $\delta(u, v) = +\infty$. Finally, if there is a path from u to v through a negative weight cycle, we set $\delta(u, v) = -\infty$. The function $\delta : V \times V \to \mathbb{R} \cup \{+\infty, -\infty\}$ is called the *distance function* on (G, ω). Finally an arc of graph G is said to be *optimal* if it participates in some shortest path. We denote by $m^*(G)$ (or simply m^*) the number of optimal arcs in G. Observe that an arc (u, v) is optimal if and only if $\omega(u, v) = \delta(u, v)$.
Most of the algorithms working on the fundamental *comparison-addition model* are based on the general *labeling* method. Such method maintains a *shortest-path estimate* function $d : V \times V \to \mathbb{R} \cup \{+\infty\}$, which is initialized (in procedure Initialize) by setting $d(u, v) := \omega(u, v)$, if $(u, v) \in E$, otherwise $d(u, v) := +\infty$. Subsequently shortest-path estimate d is updated by assignments of the form $d(u, v) := d(u, z) + d(z, v)$, provided that $d(u, v) > d(u, z) + d(z, v)$ holds, within Update(u, z, v) operations. It turns out that $d(u, v) \geq \delta(u, v)$ is maintained as an invariant, for each ordered pair $u, v \in V$ (cf. [CLRS01]).

1 A New Algorithm for the SSSP problem

In this section we present a new algorithm for the single source shortest path problem in the presence of few sources or few destinations of negative arcs. It is a generalization of Yap's approach [Yap83].

We begin with some notations. As before, let $G = (V, E)$ be a directed graph with weight function $\omega : E \rightarrow \mathbb{R}$ and source $s \in V$.

The SSSP algorithms based on the general *labeling* method maintains the shortest-path estimate as a function $d : V \rightarrow \mathbb{R} \cup \{+\infty\}$, where $d(v) = d(s, v)$. Initially, one sets $d(s) := 0$ and $d(v) := +\infty$ for $v \in V \setminus \{s\}$. Subsequently, the shortest-path estimate d is updated only by assignments of the form $d(v) := d(u) + \omega(u, v)$, provided that $d(v) > d(u) + \omega(u, v)$ holds, for all $(u, v) \in E$, within procedure $\textsc{Scan}(u)$. It turns out that $d(v) \geq \delta(s, v)$ is maintained as an invariant, for $v \in V$.

Let $e_1 = (p_1, q_1)\,, \quad e_2 = (p_2, q_2)\,, \quad \ldots\,, \quad e_\eta = (p_\eta, q_\eta)$ be the negative weight arcs in G. Let $S_G^- = \{p_1, p_2, \ldots, p_\eta\}$ be the set of sources of the negative arcs in G and let $T_G^- = \{q_1, q_2, \ldots, q_\eta\}$ be the set of destinations of the negative arcs in G. Let us also set $\ell_\sigma = |S_G^-|$, $\ell_\tau = |T_G^-|$, and $\ell = \min(\ell_\sigma, \ell_\tau)$. Next, we define the *hinge set* H *of* (G, ω) by setting $H = S_G^-$ if $\ell_\sigma \leq \ell_\tau$, while $H = T_G^-$ otherwise.

Plainly, $|H| = \ell$. Additionally, we define the *extended hinge set* H *of* (G, w) by $H = H \cup \{s\}$. Let $s_1, s_2, \ldots, s_{\bar{\ell}}$ be the elements of H, where s_1 is the source s and $\bar{\ell} = |H|$. Notice that $\ell \leq \bar{\ell} \leq \ell + 1$.

We are now ready to describe our algorithm.

Step 1. For each $s_i \in H$, apply Dijkstra's algorithm [Dij59] to (G, ω) with source node s_i. Let $\dot{d}(s_i, v)$ be the distances computed by the call to Dijkstra's algorithm from source s_i, with $v \in V$. (Notice that in general $\dot{d}(s_i, v) \neq \delta(s_i, v)$.)

Step 2. Construct the weighted graph $(\ddot{G}, \ddot{\omega})$, where $\ddot{G}$ is the complete directed graph (with no self-loops) over H, i.e., $\ddot{G} = (H, \overline{E})$, with $\overline{E} = \{(s_i, s_j) \mid i, j = 1, \ldots, \bar{\ell}\,,\ i \neq j\}\,,$ and where $\ddot{\omega}(s_i, s_j) = \dot{d}(s_i, s_j)$, for all $i, j = 1, \ldots, \bar{\ell}$ such that $i \neq j$.

Step 3. Apply the Bellman-Ford-Moore algorithm [Bel58,For56,Moo59] to the weighted graph $(\ddot{G}, \ddot{\omega})$ from the source $s_1 = s$. If negative weight cycles reachable from s_1 in $(\ddot{G}, \ddot{\omega})$ are detected, notify the presence in (G, ω) of negative weight cycles reachable from the source node and exit. Otherwise go to the next step.

Step 4. After having initialized the shortest-path estimate $d(v)$, for each $v \in G$, as $d(v) = \ddot{d}(s_i)$ if $v = s_i$, for some $i = 1, \ldots, \bar{\ell}$, and $d(v) = +\infty$ otherwise, call procedure $\textsc{Scan}(s_i)$ for each $s_i \in H$.

Step 5. Apply Dijkstra's algorithm to the weighted graph (G, ω) with source s, where the shortest-path estimate $d(v)$ is initialized with the values computed in Step 4.

Since, as noted before, we have $\ell \leq \bar{\ell} \leq \ell + 1$, then the $\bar{\ell}$ applications of Dijkstra's algorithm in Step 1 take a total time complexity of $\mathcal{O}(\ell(m + n \log n))$, provided that we use Fibonacci heaps [FT87] to implement the service priority queue. In addition, Step 2 and Step 3 take $\mathcal{O}(\ell^2)$-time and $\mathcal{O}(\ell^3)$-time, respectively, whereas Step 4 takes $\mathcal{O}(\ell + m)$-time. Finally, the last application of Dijkstra's algorithm in Step 5 takes $\mathcal{O}(m + n \log n)$-time.

Summing up, it follows that our algorithm has an overall $\mathcal{O}(\ell(m + n\log n + \ell^2))$-time complexity.
It turns out that, if $\ell = o(\sqrt[3]{mn})$ then our algorithm is asymptotically faster than the Bellman-Ford-Moore algorithm. Indeed, if $\ell = o(\sqrt[3]{mn})$ then we have: $\ell^3 = o(mn)$; $\ell m = o(mn)$, since $m = \mathcal{O}(n^2)$ so that $\ell = o(\sqrt[3]{n^3}) = o(n)$; $\ell n \log n = o(mn)$; indeed, the assumption $n = \mathcal{O}(m)$ yields $n\sqrt[3]{mn}\log n = \mathcal{O}(n\sqrt[3]{m^2}\log m)$; in addition, as $\sqrt[3]{m^2}\log m = \mathcal{O}(m)$ we also have $n\sqrt[3]{m^2}\log m = \mathcal{O}(mn)$. Thus, since $\ell n \log n = o(n\sqrt[3]{mn}\log n)$, we have $n\log n = o(mn)$. The above considerations yield immediately that if $\ell = o(\sqrt[3]{mn})$, then $\ell(m + n\log n + \ell^2) = o(mn)$.

2 A New Algorithm for the APSP Problem

In this section we present a new algorithm for the all-pairs shortest paths problem in the presence of few negative weight arcs. Our algorithm generalizes the approach presented above and is a hybridization of Dijkstra's, Floyd's [Flo62], and the Hidden Paths [KKP93] algorithms.
As before, let $G = (V, E)$ be a directed graph, with $n = |V|$ and $m = |E|$, having a weight function $\omega : E \to \mathbb{R}$. Let $e_1 = (p_1, q_1)\,, \quad e_2 = (p_2, q_2)\,, \quad \ldots\,, \quad e_\eta = (p_\eta, q_\eta)$ be the negative weight arcs in G. Let $S_G^- = \{p_1, p_2, \ldots, p_\eta\}$ be the set of sources of the negative arcs in G and let $T_G^- = \{q_1, q_2, \ldots, q_\eta\}$ be the the set of destinations of the negative arcs in G.
In this case, the *hinge set* H *of* (G, ω) is defined by $H = S_G^- \cup T_G^-$. Let $k = |H|$. Plainly, $k \leq 2\eta$, where η is the number of negative arcs in G.
We are now ready to describe our algorithm.

Step 1. For each $s_i \in H$, apply Dijkstra's algorithm to (G, ω) from the source node s_i, and let $\dot{d}(s_i, v)$ be the distance function so computed. Notice in general we have $\dot{d}(s_i, v) \neq \delta(s_i, v)$, i.e., the distances computed are not necessarily correct.
Step 2. Construct the weighted graph $(\ddot{G}, \ddot{\omega})$, where $\ddot{G} = (H, \ddot{E})$ is the directed graph (with no self-loops) over H, with $\ddot{E} = \{(s_i, s_j) \mid i, j = 1, \ldots, k\,,\ i \neq j \text{ and } \dot{d}(s_i, s_j) \neq +\infty\}$, and where $\ddot{\omega}(s_i, s_j) = \dot{d}(s_i, s_j)$, for all $(s_i, s_j) \in \ddot{E}$.
Observe that if every node in H is reacheable from all nodes in H, then the graph $\ddot{G}$ is a complete directed graph, with no self-loops.
Step 3. Apply Floyd's algorithm to the weighted graph $(\ddot{G}, \ddot{\omega})$, and let $\ddot{d}(s_i, s_j)$ be the distances computed, for $i, j = 1, \ldots, k$ with $i \neq j$. If no negative weight cycle is present, then the distance function $\ddot{d}$ computed by the above call to the Floyd's algorithm is correct, in the sense that $\ddot{d}(s_i, s_j) = \delta(s_i, s_j)$, for all $i, j = 1, \ldots, k$ such that $i \neq j$.
Step 4. Construct the weighted graph $(\tilde{G}, \tilde{\omega})$, by superimposing $(\ddot{G}, \ddot{\omega})$ to (G, ω) in the following way. Let $\tilde{E} = E \cup \ddot{E}$. Then we put $\tilde{G} = (V, \tilde{E})$. Also, we put $\tilde{\omega}(u, v) = \ddot{d}(u, v)$ if $u, v \in H$, and $\tilde{\omega}(u, v) = \omega(u, v)$ otherwise. Notice that we plainly have $|E| \leq |\tilde{E}| < m + k^2$.
Step 5. For each $s_i \in H$, apply Dijkstra's algorithm to $(\tilde{G}, \tilde{\omega})$, from source node s_i, and let $\tilde{d}(s_i, v)$ be the distance function so computed, $v \in V$.
It turns out that the distance function $\tilde{d}$ computed by Step 5 is correct, in the sense that $\tilde{d}(s_i, v) = \delta(s_i, v)$, for all $s_i \in H$ and $v \in V$.

Step 6. Let $\hat{G} = (V, \hat{E})$, where $\hat{E} = E \cup \{(u, v) : u \in H \text{ and } v \in V\}$ and let the weight function $\hat{\omega}$ be defined as $\hat{\omega}(u, v) = \tilde{d}(u, v)$ if $u \in H$, and $\hat{\omega}(u, v) = \omega(u, v)$ otherwise. Apply the Hidden Paths algorithm to the weighted graph $(\hat{G}, \hat{\omega})$ and let $\hat{d}(u, v)$ be the resulting distance function.
In view of the observation at the end of the previous step, the weighted graph $(\hat{G}, \hat{\omega})$ can contain at most kn optimal arcs more than (G, ω).
At the end of Step 6 we have $\hat{d}(u, v) = \delta(u, v)$, for every $u, v \in V$, i.e. our algorithm is correct.

The k applications of Dijkstra's algorithm in Step 1 take a total time complexity of $\mathcal{O}(k(m+n \log n))$. Then, the two graph constructions in Steps 2 and 4 take respectively $\mathcal{O}(k^2)$ and $\mathcal{O}(n + m + k^2)$-time. The execution of Floyd's algorithm in Step 3, with an input graph with k nodes, takes $\mathcal{O}(k^3)$-time, whereas the k applications of Dijkstra's algorithm in Step 5 take a total time complexity of $\mathcal{O}(k(m + k^2 + n \log n))$, since $\tilde{G}$ contains $\mathcal{O}(m + k^2)$ arcs. Finally, Step 6 takes $\mathcal{O}(n^2)$-time for the initialization of the weight function $\hat{\omega}$, and $\mathcal{O}(n\hat{m}^* + n^2 \log n)$-time for the last application of the Hidden Paths algorithm on $\hat{G}$, where $\hat{m}^*$ is the number of arcs participating in shortest paths in $\hat{G}$, yielding a total time complexity for Step 6 of $\mathcal{O}(nm^* + kn^2 + n^2 \log n)$.
Summing up, our algorithm has an overall $\mathcal{O}(k^3+kn^2+nm^*+n^2 \log n)$-time complexity. It follows immediately that when $k = o(n)$ and $m^* = o(n^2)$ our algorithm is asymptotically faster than Floyd's algorithm.
In addition, if $k = o(n)$ and $k = \mathcal{O}(\frac{m^*}{n} + \log n)$, then our algorithm achieves the same time complexity $\mathcal{O}(nm^* + n^2 \log n)$ of the Hidden Paths algorithm, which however solves the APSP problem only for nonnegative weighted graphs. Indeed, if $k = o(n)$ and $k = \mathcal{O}(\frac{m^*}{n} + \log n)$, then we have $k^3 = o(kn^2)$ and $kn^2 = \mathcal{O}(nm^* + n^2 \log n)$, so that $\mathcal{O}(k^3 + kn^2 + nm^* + n^2 \log n) = \mathcal{O}(nm^* + n^2 \log n)$.
We also observe that if $k = \mathcal{O}(\frac{m^*}{n} + \log n)$ and $m^* = \mathcal{O}(n \log n)$, then our algorithm achieves a $\mathcal{O}(n^2 \log n)$-time complexity. Indeed, if $k = \mathcal{O}(\frac{m^*}{n} + \log n)$ and $m^* = \mathcal{O}(n \log n)$, then $k = \mathcal{O}(\log n)$, so that $k^3 = \mathcal{O}(\log^3 n)$, $kn^2 = \mathcal{O}(n^2 \log n)$, and $nm^* = \mathcal{O}(n^2 \log n)$, which all together yield $\mathcal{O}(k^3 + kn^2 + nm^* + n^2 \log n) = \mathcal{O}(n^2 \log n)$.

References

[Bel58] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[Dij59] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume 1, pages 269–271. Mathematisch Centrum, Amsterdam, The Netherlands, 1959.

[Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[For56] L.R. Ford. Network flow theory. Paper P-923, The RAND Corperation, Santa Monica, California, August 1956.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[KKP93] David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal on Computing*, 22:1199–1217, 1993.

[Moo59] Edward F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.

[Yap83] Chee-Keng Yap. A hybrid algorithm for the shortest path between two nodes in the presence of few negative arcs. *Inf. Process. Lett.*, 16(4):181–182, 1983.