A Faster Algorithm for the Single Source Shortest Path Problem in the Presence of Few Sources or Destinations of 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

1 Introduction

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 most general solution of the SSSP problem is given by the Bellman-Ford-Moore algorithm [Bel58,For56,Moo59], which is characterized by a $\mathcal{O}(mn)$-time complexity. However, several algorithms have been proposed over the years to handle efficiently restricted families of weighted directed graphs. Here we are interested on restrictions related to the presence of negative weight arcs.

The case of nonnegative weight functions is solved efficiently by the celebrated Dijkstra's algorithm [Dij59], which admits a $\mathcal{O}(m+n\log n)$-time implementation based on Fibonacci heaps [FT87].

An additional type of restriction, particularly relevant to the present paper, concerns the number of negative weight arcs contained in the graph. In [Yap83], C.K. Yap describes a hybrid algorithm for the shortest path problem between *two* nodes in a directed graph in the presence of few negative weight arcs, whose running time is $\mathcal{O}(h(m+n\log n+h^2))$, where h is the minimum between n and the number of the negative weight arcs contained in the graph.

In this paper we propose a new algorithm for the single source shortest path problem, which further exploits Yap's approach. If we denote by ℓ_σ and ℓ_τ the number of nodes which are, respectively, sources and destinations of negative weight arcs in a directed graph with n nodes and m arcs, then our algorithm has a $\mathcal{O}(\ell(m+n\log n+\ell^2))$-time complexity, where $\ell=\min\{\ell_\sigma,\ell_\tau\}$. It easily turns out that when $\ell=o(\sqrt[3]{mn})$, the algorithm we propose is asymptotically faster than the Bellman-Ford-Moore algorithm. Plainly, our algorithm is always at least as fast as Yap's one, and it turns to be faster when $\ell=o(h)$.

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 set $n=|V|$ and $m=|E|$. A *weight function*

ω on $G = (V, E)$ is any real function $\omega : E \to \mathbb{R}$. A *path* in $G = (V, E)$ is any finite sequence $(v_0, v_1, \ldots, v_k)$ of nodes such that (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 $0 < j < k-1$. 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).

Throughout the paper we will assume that graphs satisfy the relation $n = \mathcal{O}(m)$. This is certainly the case, for instance, when all nodes are reachable from the given source node, which can be assumed without any loss of generality.

Nearly all known SSSP algorithms can be divided into two classes: those which assume *real*-weighted graphs, where reals are manipulated only by *comparison* and *addition* operations, and those which assume *integer* weighted graphs, where integers can be manipulated by a given suite of RAM-type operations. In this note we are interested only into algorithms belonging to the first class.

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 $d : V \to \mathbb{R} \cup \{+\infty\}$, a *predecessor* function $\pi : V \to V \cup \{\text{NIL}\}$, and a *status* function

$$S : V \to \{\text{UNREACHED}, \text{LABELED}, \text{SCANNED}\}.$$

Initially, one sets $d(s) := 0$, $\pi(s) :=$ NIL, $S(s) :=$ LABELED, where s is the source node, and also sets $d(v) := +\infty$, $\pi(v) :=$ NIL, and $S(v) :=$ UNREACHED, for $v \in V \setminus \{s\}$. Subsequently, the shortest-path estimate d is updated only within the SCAN operation, called for a given node u, which performs assignments of the form $d(v) := d(u) + \omega(u, v)$ for each $(u, v) \in E$, provided that $d(v) > d(u) + \omega(u, v)$ holds, in which case one also sets $\pi(v) := u$ and $S(v) :=$ LABELED. It turns out that $d(v) \geq \delta(s, v)$ is maintained as an invariant, for $v \in V$. Additionally, when $d(v) = \delta(s, v)$, the predecessor function π can be used to reconstruct a shortest path from s to v backwards (cf. [CLRS01]).

2 A new algorithm in the presence of few sources or destinations of negative arcs

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, by generalizing Yap's approach.

We begin with some notations. As before, let $G = (V, E)$ be a directed graph with weight function $\omega : E \to \mathbb{R}$ and source $s \in V$, and 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. Let us also set $\ell_\sigma = |S_G^-|$, $\ell_\tau = |T_G^-|$, and $\ell = \min(\ell_\sigma, \ell_\tau)$. Next, let us define the *hinge set* H *of* (G, ω) by setting

$$H = \begin{cases} S_G^- & \text{if } \ell_\sigma \leq \ell_\tau \\ T_G^- & \text{otherwise}\,. \end{cases}$$

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

We are now ready to describe our algorithm.

Step 1. For each $s_i \in \overline{H}$, apply Dijkstra's algorithm 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 $\overline{H}$, i.e., $\ddot{G} = (\overline{H}, \overline{E})$, 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 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 follows

$$d(v) = \begin{cases} \ddot{d}(s_i) & \text{if } v = s_i, \text{ for some } i = 1, \ldots, \bar{\ell} \\ +\infty & \text{otherwise}\,, \end{cases}$$

call procedure $\textsc{Scan}(s_i)$ for each $s_i \in \overline{H}$.

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

Remark 1. Notice that the calls to Dijkstra's algorithm within the above steps are intended to its standard implementation, namely the one which guarantees correct results only when negative weight arcs are absent (see [CLRS01]). ■

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 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.

If $\ell = o(\sqrt[3]{mn})$ then our algorithm is asymptotically faster than the Bellman-Ford-Moore algorithm. This can be checked easily by observing that 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 $\ell(m + n \log n + \ell^2) = o(mn)$.

Next we compare our algorithm with Yap's one. Notice that $\ell = \mathcal{O}(h)$, where as before h denotes the minimum between n and the number of the negative weight arcs in the graph. Therefore we have $\ell(m + n \log n + \ell^2) = \mathcal{O}(h(m + n \log n + h^2))$, i.e. our algorithm is always asymptotically at least as good as Yap's one. In addition, if $\ell = o(h)$, then $\ell(m + n \log n + \ell^2) = o(h(m + n \log n + h^2))$, i.e. in this case our algorithm is asymptotically faster.

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.

[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.

[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.