On the All-Pairs Shortest-Path Algorithm of Moffat and Takaoka*

Kurt Mehlhorn, Volker Priebe
Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany

Received 20 December 1995; revised 25 August 1996

ABSTRACT: We review how to solve the all-pairs shortest-path problem in a nonnegatively weighted digraph with n vertices in expected time $O(n^2 \log n)$. This bound is shown to hold with high probability for a wide class of probability distributions on nonnegatively weighted digraphs. We also prove that, for a large class of probability distributions, $\Omega(n \log n)$ time is necessary with high probability to compute shortest-path distances with respect to a single source. © 1997 John Wiley & Sons, Inc. *Random Struct. Alg.*, **10**, 205–220 (1997)

Key Words: shortest-path problems; probabilistic analysis; tail estimates; lower bound

1. INTRODUCTION

Given a complete digraph in which all the edges have nonnegative length, we want to compute the shortest-path distance between each pair of vertices. This is one of the most basic questions in graph algorithms, since a variety of combinatorial-optimization problems can be expressed in these terms. As far as worst-case complexity is concerned, we can solve an n-vertex problem in time $O(n^3)$ either by Floyd's algorithm [4] or by n calls of Dijkstra's algorithm [2]. Fredman's algorithm [5] uses

*A preliminary version of this paper was presented at the Third Annual European Symposium on Algorithms, Corfu, Greece, 1995 [12].
Correspondence to: V. Priebe

efficient distance-matrix multiplication techniques and results in a running time of $O(n^3((\log\log n)/\log n)^{1/3})$ (slightly improved to $O(n^3((\log\log n)/\log n)^{1/2})$ by Takaoka [17]). Recently, Karger, Koller, and Phillips [10] presented an algorithm that runs in time $O(nm^* + n^2 \log n)$, where m^* denotes the number of edges that are a shortest path from their source to their target.

However, worst-case analysis sometimes fails to bring out the advantages of algorithms that perform well in practice; average-case analysis has turned out to be more appropriate for these purposes. We are not only interested in algorithms with good expected running time but in algorithms that finish their computations within a certain time bound *with high probability* (and might therefore be called reliable).

Two kinds of probability distributions on nonnegatively weighted complete digraphs have been considered in the literature. In the so-called *uniform model*, the edge weights are independent, identically distributed random variables. In the so-called *endpoint-independent model*, an arbitrary (multi-)set c_{vj}, $1 \le j \le n$, of n nonnegative weights (possibly ∞) is fixed for each vertex v. These weights are assigned randomly to the n edges with source v, i.e., a random bijective mapping $\pi^{(v)}$ from $\{1,\ldots,n\}$ to V is chosen and c_{vj} is made the weight of the edge $(v, \pi^{(v)}(j))$ for all j, $1 \le j \le n$.

Frieze and Grimmett [7] gave an algorithm with $O(n^2 \log n)$ expected running time in the uniform model when the common distribution function F of the edge weights satisfies $F(0) = 0$, $F'(0)$ exists, and $F'(0) > 0$. Under these assumptions, $m^* = O(n \log n)$ with high probability, and so the algorithm of Karger et al. [10] also achieves running time $O(n^2 \log n)$ with high probability.

The endpoint-independent model is more general and harder to analyze. Spira [16] proved an expected time bound of $O(n^2(\log n)^2)$, which was later improved by Bloniarz [1] to $O(n^2 \log n \log^* n)$. (We use log to denote logarithms to base 2 and ln to denote natural logarithms; $\log^* x := 1$ for $x \le 2$ and $\log^* x := 1 + \log^* \log x$ for $x > 2$.) In [13] and [14], Moffat and Takaoka describe two algorithms with an expected time bound of $O(n^2 \log n)$. The algorithm in [14] is a simplified version of that of [13]. In Section 3 of this paper, we review the algorithm in [14] and the analysis of its expected running time. We point out some easy-to-make mistakes in the analysis and show how to avoid them. Moreover, we prove that the running time of the algorithm is $O(n^2 \log n)$ with high probability. In Section 4, we show that under modest assumptions $\Omega(n \log n)$ edges need to be inspected to compute the shortest-path distances with respect to a single source.

2. PRELIMINARIES

We will refer to the following probabilistic experiments. Let an urn contain n balls that are either red or blue; let r be the number of red balls. The balls are repeatedly drawn from the urn (without replacement) uniformly at random. Let W be the number of drawings until the first red ball occurs; then

$$E[W] = \sum_{0 \le k \le n-r} \Pr(W > k) = \sum_{0 \le k \le n-r} \binom{n-r}{k} \Big/ \binom{n}{k} = \sum_{0 \le k \le n-r} \binom{n-k}{r} \Big/ \binom{n}{r},$$

and from $\binom{n-k}{r} = \binom{n+1-k}{r+1} - \binom{n-k}{r+1}$, we conclude that

$$E[W] = \binom{n+1}{r+1} \Big/ \binom{n}{r} = \frac{n+1}{r+1}.$$

We are also interested in sampling with replacement. Suppose that in a sequence of independent trials, the probability of success is p for each of the trials. Let Z be the number of trials until the first successful one; then

$$E[Z] = \sum_{k \geq 0} \Pr(Z > k) = \sum_{k \geq 0} (1-p)^k = 1/p.$$

In the so-called *coupon-collector problem*, we are given a set of n distinct coupons and we are trying to complete a collection of all coupons. In each trial, a coupon is drawn (with replacement) uniformly and independently at random. We call a trial a success if it results in adding a new coupon to our collection. Let Z^* denote the completion time, i.e., the number of trials required to have seen at least one copy of each coupon. $Z^* = 1 + Z_1 + \cdots + Z_{n-1}$, where for $1 \leq i < n$, Z_i is the number of trials (with probability of success $\frac{n-i}{n}$ each) between the ith and $(i+1)$th success (excluding the former, including the latter). By the argument above, $E[Z_i] = \frac{n}{n-i}$, and hence, $E[Z^*] = \sum_{0 \leq i < n} \frac{n}{n-i} \leq n(\ln n + 1) = O(n \ln n)$.

For a problem of size n, we will say that an event occurs *with high probability* if it occurs with probability $\geq 1 - O(n^{-C})$ for an arbitrary but fixed constant C. For example, in the coupon-collector problem (with n coupons), the probability that a particular coupon has not been collected after t trials equals $(1 - \frac{1}{n})^t$. Hence, for any $\beta > 1$,

$$\Pr(Z^* > \beta n \ln n) \leq n\left(1 - \frac{1}{n}\right)^{\beta n \ln n} \leq n e^{-\beta \ln n} = n^{-(\beta-1)}; \tag{1}$$

that is, the completion time in the coupon-collector problem (with n coupons) is $O(n \ln n)$ with high probability. (1) establishes what is called a *large-deviation estimate* for Z^*, i.e., a bound on $\Pr(Z^* > z)$, where z is of about the order of $E[Z^*]$.

We are actually interested in a large-deviation estimate for a random variable $X^* := 1 + X_1 + \cdots + X_{n-1}$, where each X_i is stochastically dominated by the random variable Z_i from the coupon-collector problem. For two random variables X and Y taking values in the positive integers, X is said to be *stochastically dominated* by Y, written $X \leq_{st} Y$, if $\Pr(X > k) \leq \Pr(Y > k)$ for all $k \geq 0$. Note that $X \leq_{st} Y$ implies $E[X] \leq E[Y]$. Stochastic dominance is preserved under taking sums of independent random variables, i.e., if $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ are independent and $X_i \leq_{st} Y_i$ for $1 \leq i \leq n$, then $X_1 + \cdots + X_n \leq_{st} Y_1 + \cdots + Y_n$; see, for example, [18]. We will need the following, slightly more general result, which is a generalization of Lemma 7 of [15].

Lemma 1. *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ be random variables that take values in the positive integers. Suppose that each $X_i, 1 \leq i \leq n$, conditioned on any possible tuple of values for $X_1, \ldots, X_{i-1}$, is stochastically dominated by Y_i, and that $Y_1, \ldots, Y_n$ are independent. Then $X^{(n)} := \sum_{1 \leq i \leq n} X_i$ is stochastically dominated by $Y^{(n)} := \sum_{1 \leq i \leq n} Y_i$.*

Proof. The proof is by induction on n. For the base case ($n = 1$), we have $X^{(1)} = X_1$, $Y^{(1)} = Y_1$, and $\Pr(X_1 > k) \le \Pr(Y_1 > k)$ for any $k \ge 0$ by assumption. For the induction step ($n - 1 \to n$), note that, on the one hand, for any $k \ge n$,

$$\Pr(Y^{(n)} > k) = \Pr(Y_n > k - n + 1) + \sum_{j=1}^{k-n+1} \Pr(Y_n = j) \cdot \Pr(Y^{(n-1)} > k - j), \quad (2)$$

since $\Pr(Y^{(n-1)} \ge n - 1) = 1$. On the other hand, for any $k \ge n$,

$$\begin{aligned}
&\Pr(X^{(n)} > k) \\
&\quad = \Pr(X^{(n-1)} > k - 1) + \sum_{j=1}^{k-n+1} \Pr(X_n > j \mid X^{(n-1)} = k - j) \cdot \Pr(X^{(n-1)} = k - j) \\
&\quad \le \Pr(X^{(n-1)} > k - 1) + \sum_{j=1}^{k-n+1} \Pr(Y_n > j) \cdot \Pr(X^{(n-1)} = k - j),
\end{aligned}$$

since X_n is stochastically dominated by Y_n, regardless of the value of $X^{(n-1)}$. Using $\Pr(X^{(n-1)} = k - j) = \Pr(X^{(n-1)} \ge k - j) - \Pr(X^{(n-1)} > k - j)$ and rearranging the sum, we get

$$\Pr(X^{(n)} > k) \le \Pr(Y_n > k - n + 1) + \sum_{j=1}^{k-n+1} \Pr(Y_n = j) \cdot \Pr(X^{(n-1)} > k - j). \quad (3)$$

By the induction hypothesis and (2), we deduce from (3) that

$$\begin{aligned}
\Pr(X^{(n)} > k) &\le \Pr(Y_n > k - n + 1) + \sum_{j=1}^{k-n+1} \Pr(Y_n = j) \cdot \Pr(Y^{(n-1)} > k - j) \\
&= \Pr(Y^{(n)} > k).
\end{aligned}$$

Since $\Pr(X^{(n)} > k) = 1 = \Pr(Y^{(n)} > k)$ for $0 \le k < n$, we have thus proved that $X^{(n)} \le_{st} Y^{(n)}$. ∎

We will also use the following two results on large-deviation estimates. The first lemma is usually referred to as the *Chernoff–Hoeffding bound* on the tail of the distribution of a sum of independent random variables; see [8, 9] for a proof.

Lemma 2. *Let X be the sum of independent random variables $X_1, \ldots, X_n$ with values in $[0, 1]$. (The X_i's need not be identically distributed.) Then, for any $\varepsilon > 0$,*

$$\Pr(X > (1 + \varepsilon) E[X]) \le \left(\frac{e^{\varepsilon}}{(1 + \varepsilon)^{(1+\varepsilon)}} \right)^{E[X]}. \quad (4)$$

In particular, (4) implies that

$$\Pr(X > a) \le 2^{-a} \qquad \text{for } a \ge 6E[X]. \quad (5)$$

The following formulation of *Azuma's inequality* appears in [11].

Lemma 3. *Let $X_1, \ldots, X_n$ be independent random variables, with X_k taking values in a set A_k for each k. Suppose that the function $f : \prod A_k \to \mathbb{R}$ satisfies $|f(x) - f(y)| \le c$ whenever the vectors x and y differ only in a single coordinate. Let Y be the random variable $f(X_1, \ldots, X_n)$. Then, for any $t > 0$,*

$$\Pr(|Y - E[Y]| \ge t) \le 2e^{-2t^2/(nc^2)}.$$

3. THE ALGORITHM OF MOFFAT AND TAKAOKA

We use the following terminology for weighted digraphs. For an edge $e = (v, w)$, we call v the *source* and w the *target* or *endpoint* of e. The weight of an edge e is denoted by $c(e)$. We will interpret an entry in the adjacency list of a vertex v either as the endpoint w of an edge with source v or as the edge (v, w) itself, as is convenient.

For the sake of future reference, we briefly review the algorithm of Moffat and Takaoka in [14]. We are given a complete digraph (with loops) on n vertices with nonnegative edge weights. The algorithm first sorts all adjacency lists in order of increasing weight (with ties resolved randomly, total time $O(n^2 \log n)$) and then solves n single-source shortest-path problems, one for each vertex. A single-source shortest-path problem with source $s \in V$ is solved by *labeling* the vertices in order of increasing distance from the source. If v is a labeled vertex, then its exact distance $d(v)$ from the source is known. We use S to denote the set of labeled vertices and $U = V - S$ to denote the set of unlabeled vertices. Initially, only the source vertex is labeled, i.e., $S = \{s\}$ with $d(s) = 0$. For each labeled vertex v, one of its outgoing edges is called its *current edge* and is denoted $ce(v)$; we maintain the invariant that all edges preceding the current edge $ce(v)$ in v's adjacency list have their endpoints already labeled. We say that the edges preceding $ce(v)$ (as well as their targets) have been *scanned* by the algorithm. The *potential* of v's current edge is defined as $d(v) + c(ce(v))$. The algorithm proceeds in *iterations*. In each iteration, the algorithm selects the current edge of minimum potential; suppose that $ce(v)$ is selected and that w is its target. If w is not yet labeled, then the algorithm labels w (i.e., adds w to S) and sets $d(w)$ to $d(v) + c(ce(v))$. (It follows by a standard argument as for Dijkstra's algorithm that this indeed sets $d(w)$ to the distance of w from s.) Moreover, some current-edge pointers are updated. The precise nature of these updates depends on the size of U.

As long as $|U| > n/\log n$, the algorithm is said to be in *Phase I*, and the additional invariant is maintained that the targets of all current edges are unlabeled. Whenever the algorithm selects a current edge $ce(v)$ of minimum potential, the target of $ce(v)$ will therefore be a vertex u in U. The algorithm labels u and sets $d(u)$ to $d(v) + c(ce(v))$. In order to maintain the invariant of Phase I, the algorithm advances the current-edge pointer of u and the current-edge pointers of all the vertices whose current edges end in u; the pointers are advanced to the next edge with target in $V - S$ in the respective adjacency lists. Phase I ends when $|U|$ becomes $n/\log n$; let U_0 be the set of unlabeled vertices at the end of Phase I.

If $|U| \le n/\log n$, the algorithm is said to be in *Phase II*, and the weaker additional invariant is maintained that the endpoint of every current edge belongs to U_0. Suppose that the current edge $ce(v) = (v, w)$ is selected in an iteration. The vertex $w \in U_0$ is not necessarily unlabeled. If w is unlabeled, it will be labeled, $d(w)$ is set to $d(v) + c(ce(v))$, and $ce(v)$ and $ce(w)$ are advanced to the next edge whose endpoint is in U_0. If w is already labeled, only $ce(v)$ is advanced.

Lemma 4. *The algorithm spends time $O((n + \xi)\log n + \mu)$ on solving a single-source shortest-path problem, where ξ is the number of iterations in Phase II and μ is the total number of edges scanned in the two phases.*

Proof. Since the algorithm does exactly $n - n/\log n$ iterations in Phase I, it performs a total number of $O(n + \xi)$ iterations. In each iteration, we select a current edge of minimum potential, and we have to update the current-edge pointers as well as the information on their potentials. The cost of updating the current-edge pointers in an iteration is proportional to the increase $\Delta\mu$ in the number of scanned edges. The lemma follows if we prove that, in each iteration, selecting the current edge of minimum potential and updating the information on the potentials can be done in $O(\log n + \Delta\mu)$ time.

Both phases use a *priority queue* for maintaining information on edge potentials. A priority queue stores a set of pairs (x, k), where k, the *key* of the pair, is a real number. For ease of presentation, we assume that priority queues are implemented as Fibonacci heaps [6]. Fibonacci heaps support the insertion of a new pair in constant time and the deletion of a pair with minimum key (a *delete min* operation) in amortized time $O(\log p)$, where p is the number of pairs in the priority queue. They also support an operation *decrease key* in constant amortized time. A *decrease key* operation takes a pointer to a pair (x, k) in a Fibonacci heap and allows the replacement of k by a smaller key k'.

We propose the following implementation of Phase I. We batch the current edges with respect to their endpoints, i.e., the priority queue contains all unlabeled vertices. For each vertex $u \in U$, we maintain a list $L(u)$ of all vertices $v \in S$ whose current edge ends in u; the key of a vertex $u \in U$ is $d_u := \min_{v \in L(u)} d(v) + c(v, u)$ (understood to be $+\infty$ if $L(u) = \emptyset$). An iteration of the algorithm corresponds to selecting the vertex $u \in U$ with minimal key value d_u and deleting u from the priority queue with a *delete min* operation. The current-edge pointer must be advanced for each vertex $v \in \{u\} \cup L(u)$. Let $ce(v) = (v, w)$ be the new current edge of v and denote w's current key by d_w. We add v to $L(w)$, and if $d(v) + c(v, w) < d_w$, we decrease d_w appropriately. This is realized through a *decrease key* operation on the priority queue. By our assumption on the implementation of the priority queue, all of this takes time $O(\log n + \Delta\mu)$.

In Phase II, we represent current edges by their sources. We keep the vertices $v \in S$ in the priority queue with key $d(v) + c(ce(v))$. An iteration of Phase II requires a *delete min* operation and the insertion of at most two new pairs in the queue. This takes time $O(\log n)$. ∎

Remark. Moffat and Takaoka use a binary heap instead of a Fibonacci heap to realize the priority queue; Fibonacci heaps had not been invented at that time. In the implementation of Phase I, they keep the vertices in S in the priority queue; the key of a vertex $v \in S$ is $d(v) + c(ce(v))$. Advancing the current-edge pointers will then require *increasing* the keys of certain labeled vertices, since the weight of the new current edge is greater than the weight of the old current edge. An *increase key* operation in general takes logarithmic time in a binary heap. However, Moffat and Takaoka show how to modify the implementation so that the *expected* cost of all the *increase key* operations in an iteration is $O(|S|/(n - |S|) + \log|S|)$, which is $O(\log n)$ during Phase I. The probabilistic model underlying their analysis is described in the next subsection.

3.1. The Probabilistic Analysis (and Its Pitfalls)

The algorithm is analyzed under the assumption that edge weights are distributed according to the endpoint-independent model. This means that with arbitrarily fixed nonnegative edge weights, the adjacency lists (sorted in order of increasing weight, ties resolved randomly) are independent random permutations of V. The algorithm of Moffat and Takaoka solves the all-pairs shortest-path problem in this model in expected time $O(n^2 \log n)$; more precisely, it solves each single-source shortest-path problem in expected time $O(n \log n)$. (Theorem 3 in Section 4 shows that the running time for solving the single-source shortest-path problem is actually optimal for a large class of related probability distributions.) As indicated in Lemma 4, the quantities of interest in the analysis are the number ξ of iterations in Phase II and the total number μ of scanned edges. We will argue in Theorem 1 that the expected values of ξ and μ are $O(n)$ and $O(n \log n)$, respectively.

The analysis of μ turns out to be intricate. We want to mention two possible pitfalls. What is the total number of edges scanned in Phase I? In [14], Moffat and Takaoka argue as follows. The cardinality of U_0, the set of unlabeled vertices at the end of Phase I, is $n/\log n$ by design, and at the end of Phase I, current-edge pointers will have been advanced to the first vertex in U_0 in each adjacency list. Since, for every vertex v, the endpoints of the edges out of v form a random permutation of V, the vertices in U_0 are randomly scattered through each adjacency list. We should therefore expect to scan about $\log n$ edges in each adjacency list during Phase I and hence about $n \log n$ edges altogether. This argument is incorrect as U_0 is determined by the orderings of the adjacency lists and cannot be fixed independently. The following example makes this clear. Assume that all edges out of the source have weight 1 and all other edges have weight 2. Then Phase I scans $n - n/\log n$ edges out of the source and U_0 is determined by the last $n/\log n$ edges in the adjacency list of the source.

In Phase II, the number of iterations is a random variable with expected value $O(n)$. Moreover, whenever the current edge of a vertex is advanced in Phase II, it is advanced to the next edge having its endpoint in U_0, and this requires scanning $O(\log n)$ edges on average. It is tempting to state that the expected number of edges scanned in Phase II is therefore $O(n \log n)$. The claimed result would follow if the expected number of scanned edges, given that the algorithm finishes Phase II in κ iterations, were $O(\kappa \log n)$, and in fact, in a preliminary version of this paper, [12], we analyzed Phase II along these lines. We now feel that a more careful

argumentation is needed. It is not clear whether the number of iterations and the distance between two consecutive edges with endpoints in U_0 are independent or, for example, positively correlated random variables. We are grateful to an anonymous referee for drawing our attention to this oversight.

It is for these reasons that we give a new proof of the following theorem. Our proof evolved from suggestions by Alistair Moffat (personal communication) and by two anonymous referees and replaces a considerably more involved argument in drafts of this paper.

Theorem 1. *For endpoint-independent distributions, the algorithm of Moffat and Takaoka runs in expected time* $O(n^2 \log n)$.

Proof. For the purpose of the analysis, we also consider Spira's algorithm [16]. This algorithm is similar to the algorithm by Moffat and Takaoka, the only difference being that Spira's algorithm does not impose any condition on the targets of current edges but always advances the current-edge pointer only to the next edge in the adjacency list. The algorithm does not distinguish between phases. It stops when all vertices have been labeled. Given an ordering of the adjacency lists, the algorithms by Moffat and Takaoka and by Spira show basically the same behavior. All edges that the algorithm by Moffat and Takaoka selects as edges of minimum potential are also selected by Spira's algorithm. However, upon termination, the current-edge pointers in the algorithm of Moffat and Takaoka may have been advanced beyond those in Spira's algorithm, since the invariants of the algorithm by Moffat and Takaoka enforce that the target of every current-edge pointer is a vertex in U_0. (Nevertheless, the algorithm by Moffat and Takaoka is more efficient, since the scanning strategies of Phase I and II tend to reduce the number of priority-queue operations.)

The following observations are crucial for the analysis of the algorithms. Suppose that we stop Spira's algorithm after its first κ iterations, where κ is an arbitrary but fixed number. The behavior of the algorithm in these iterations is completely determined by the edges scanned so far. For a set A of edges, denote by $\mathscr{A}_\kappa = A$ the event that Spira's algorithm scans exactly the edges in A in the first κ iterations. We consider an arbitrary but fixed set A with $\Pr(\mathscr{A}_\kappa = A) > 0$; assume that, for $v \in V$, A contains exactly n_v edges with source v and W_v is the set of their targets. The event $\mathscr{A}_\kappa = A$ does not yield any information about the remaining parts of the adjacency lists; for each $v \in V$, the remaining part of v's adjacency list is therefore a random permutation of $V - W_v$. (We may interpret this as fixing the permutations on-line, the so-called principle of deferred decisions.)

From this we conclude that the total number of edges scanned by Spira's algorithm is stochastically dominated by the completion time of the coupon-collector problem with n coupons. Namely, assume that $\mathscr{A}_\kappa = A$ implies that exactly i vertices have been labeled in the first κ iterations. If the next edge scanned has source v, then the target of the edge is already labeled with probability $(i - n_v)/(n - n_v) \le i/n$, since every vertex in $V - W_v$ is equally likely to occur as the target of v's current edge. More generally, the probability that the algorithm will select edges with labeled targets in the next k iterations is bounded from above by $(i/n)^k = (1 - (n-i)/n)^k$ for every $k \ge 0$. It follows that M_i, the number of edges scanned by Spira's algorithm between the labelings of the ith and the $(i+1)$th

vertex, is stochastically dominated by the random variable Z_i, introduced in the analysis of the coupon-collector problem in Section 2. By Lemma 1 in Section 2 we conclude that $M := 1 + M_1 + \cdots + M_{n-1}$, the total number of edges scanned by Spira's algorithm, is indeed stochastically dominated by the completion time of the coupon-collector problem with n coupons. In particular,

$$E[M] \le n(\ln n + 1), \tag{6}$$

i.e., Spira's algorithm scans an expected number of $O(n \log n)$ edges.

An argument of the same kind as in the previous paragraph applies to the targets of current edges in Phase II of the algorithm by Moffat and Takaoka. If X denotes the number of iterations in Phase II, then X is stochastically dominated by the completion time of a coupon-collector problem with $r := |U_0| = n/\log n$ coupons; in particular,

$$E[X] \le r(\ln r + 1) = O(n).$$

The expected value of ξ in Lemma 4 is therefore $O(n)$.

It remains to analyze the number of extra edges scanned by the algorithm of Moffat and Takaoka. For a set of edges A, denote by $\mathscr{A}^* = A$ the event that Spira's algorithm scans exactly these edges before it stops. We consider an arbitrary but fixed set A with $\Pr(\mathscr{A}^* = A) > 0$; assume that, for $v \in V$, A contains exactly n_v edges with source v and r_v of these edges have targets in U_0. Given that $\mathscr{A}^* = A$ occurs, for each $v \in V$, the current-edge pointer in the algorithm by Moffat and Takaoka will finally have been advanced to the $(n_v + Y_v)$th position in v's adjacency list, where $Y_v = 0$ if $r_v = r = |U_0|$ and, otherwise, $n_v + Y_v$ is the position of the next vertex in U_0 in v's adjacency list. Again, by the arguments above, the remaining part of v's adjacency list can be interpreted as a random permutation of $n - n_v$ elements, containing $r - r_v$ elements from U_0. This implies that Y_v is distributed as in the urn experiment in Section 2 with

$$E[Y_v \mid \mathscr{A}^* = A] = \frac{n - n_v + 1}{r - r_v + 1} \qquad \text{if} \quad r_v < r. \tag{7}$$

Note that $E[Y_v \mid \mathscr{A}^* = A] \le 2n/r$ as long as $r_v \le r/2$.

The expected amount of extra work is $E[\sum_v Y_v]$. Conditioning on $\mathscr{A}^* = A$, we get, using (7) (or the trivial bound $Y_v \le n$ if $r_v > r/2$),

$$\begin{aligned} E\left[\sum_v Y_v \,\middle|\, \mathscr{A}^* = A\right] &\le \frac{2n}{r} \cdot \left|\left\{v \,;\, r_v \le \frac{r}{2}\right\}\right| + n \cdot \left|\left\{v \,;\, r_v > \frac{r}{2}\right\}\right| \\ &\le \frac{2n^2}{r} + n \cdot \frac{2}{r} \cdot \sum_v r_v. \end{aligned} \tag{8}$$

Under the condition $\mathscr{A}^* = A$, the number X of iterations in Phase II equals $\sum_v r_v$. Hence, by summing over all sets A with $\Pr(\mathscr{A}^* = A) > 0$, (8) implies

$$E\left[\sum_v Y_v\right] = \sum_A E\left[\sum_v Y_v \,\middle|\, \mathscr{A}^* = A\right] \cdot \Pr(\mathscr{A}^* = A) \le \frac{2n}{r} \cdot (n + E[X]),$$

and since $r = n/\log n$ and $E[X] = O(n)$, we get $E[\sum_v Y_v] = O(n \log n)$. Combining this with (6), we conclude that the expected value of μ in Lemma 4 is $O(n \log n)$.

We deduce from Lemma 4 that the algorithm of Moffat and Takaoka solves a single-source shortest-path problem in expected time $O(n \log n)$ and has therefore an expected running time of $O(n^2 \log n)$. ■

We will next prove that the algorithm of Moffat and Takaoka is reliable, i.e., that, with high probability, its running time does not exceed its expectation by more than a constant multiplicative factor.

Theorem 2. *The running time of the all-pairs shortest-path algorithm of Moffat and Takaoka is $O(n^2 \log n)$ with high probability.*

Proof. It suffices to prove that the algorithm solves a single-source shortest-path problem in time $O(n \log n)$ with high probability. This follows from Lemma 4 if the total number of iterations and the total number of scanned edges can be proved to be, with high probability, $O(n)$ and $O(n \log n)$, respectively. We use the notation introduced for the proof of Theorem 1.

As already observed in the proof of Theorem 1, the total number X of iterations in Phase II is stochastically dominated by the completion time of a coupon-collector problem with $r = n/\log n$ coupons. Using the tail estimate (1) in Section 2, we deduce that the number of iterations in Phase II is $O(r \log r) = O(n)$ with high probability; for any arbitrary (but fixed) $C > 0$,

$$\Pr(X > Kn) \le \tfrac{n}{\log n}\left(1 - \tfrac{1}{n/\log n}\right)^{Kn} \le \tfrac{n}{\log n} \cdot e^{-K \log n} = O(n^{-C}) \tag{9}$$

for some constant K. Hence, the total number of iterations is $O(n)$ with high probability.

Again, by the tail estimate (1) for the coupon-collector problem, Spira's algorithm scans $O(n \log n)$ edges with high probability. Therefore, we only have to prove that the extra scanning $\sum_v Y_v$ of the algorithm by Moffat and Takaoka is $O(n \log n)$ with high probability. As in the proof of Theorem 1, we condition on $\mathscr{A}^* = A$, i.e., on the event that Spira's algorithm scans exactly the edges in A before it stops. For $v \in V$, let A contain exactly n_v edges with source v and let r_v of these edges have targets in U_0. Because of (9), we may assume that $X = \sum_v r_v \le Kn$. Given $\mathscr{A}^* = A$, we have, with $r = |U_0| = n/\log n$,

$$\begin{aligned}
\sum_v Y_v &\le n \cdot |\{v;\, r_v > r/2\}| + \sum_{v,\, r_v \le r/2} Y_v \\
&\le \frac{2n}{r} \cdot \left(\sum_v r_v\right) + \sum_{v,\, r_v \le r/2} Y_v \\
&\le \frac{2n}{r} \cdot Kn + \sum_{v,\, r_v \le r/2} Y_v = 2Kn \log n + \sum_{v,\, r_v \le r/2} Y_v,
\end{aligned}$$

which implies that

$$\Pr\left(\sum_v Y_v > 3Kn \log n \,\middle|\, \mathscr{A}^* = A\right) \le \Pr\left(\sum_{v,\, r_v \le r/2} Y_v/n > K \log n \,\middle|\, \mathscr{A}^* = A\right). \tag{10}$$

Conditionally on $\mathscr{A}^* = A$, $\sum_{v, r_v \le r/2} Y_v/n$ is a sum of independent (not necessarily identically distributed) random variables with values in $[0,1]$, and we can therefore apply the Chernoff–Hoeffding bound from Lemma 2. By (7),

$$E\left[\sum_{v, r_v \le r/2} Y_v/n \,\middle|\, \mathscr{A}^* = A\right] \le \sum_{v, r_v \le r/2} \frac{1}{n} \cdot \frac{n - n_v + 1}{r - r_v + 1} \le \frac{2n}{r} = 2 \log n,$$

independent of A. Hence, for K chosen sufficiently large, we deduce from (5) in Section 2 that

$$\Pr\left(\sum_{v, r_v \le r/2} Y_v/n > K \log n \,\middle|\, \mathscr{A}^* = A\right) \le 2^{-K \log n},$$

and by (10), this implies that $\sum_v Y_v = O(n \log n)$ with high probability. ■

4. A LOWER BOUND FOR THE SINGLE-SOURCE PROBLEM

Our underlying graph is $\tilde{K}_n = (V, E)$, the complete digraph on n vertices with loops. We restrict ourselves to the case of *simple* weight functions on the edges, i.e., for every vertex v and each integer k, $1 \le k \le n$, there is exactly one edge with weight k and source v. A single-source shortest-path algorithm gets as its input the problem size n, a source vertex s, and a simple weight function c. We assume that c is provided in the form of an oracle that answers questions of the following kind:

(1) What is the weight $c(e)$ of a given edge e?
(2) Given a vertex $v \in V$ and an integer $k \in \{1, \ldots, n\}$, what is the target of the edge with source v and weight k?

The algorithm is supposed to compute the function d of shortest distances from s. It is allowed to ask the oracle questions of type (1) and (2), thereby gaining partial information about c. The complexity of the algorithm on a fixed simple weight function c is defined to be the number of questions asked by the algorithm in order to compute the distance function d with respect to c.

For simple weight functions, the distance function d maps the set of vertices into $\mathbb{N}_0$. Define $D := \max\{d(v) \,;\, v \in V\}$, and, for all i, $0 \le i \le D$, let $V_i := \{v \,;\, d(v) = i\}$. We call V_i the ith layer with respect to d. For all i, $0 \le i \le D$, let $\ell(i) := |\{j \,;\, j > i \text{ and } V_j \ne \emptyset\}|$ be the number of nonempty layers above layer i. Clearly, D, the sets V_i, and the function ℓ depend on c; for ease of notation, we do not make this dependence visible in the notation.

We first provide a lower bound on the complexity of a single-source shortest-path algorithm in terms of ℓ.

Lemma 5. *Let c be a simple weight function and let d be the distance function with respect to c. Then any shortest-path algorithm has complexity at least*

$$\sum_{u \in V} (\ell(d(u)) - 1).$$

Proof. Let E' be the set of edges queried by the algorithm by a question of either type (1) or type (2). For an arbitrary but fixed vertex $u \in V$, let $E(u)$ be the set of edges with source u and let $E'(u) := E' \cap E(u)$. We prove that $|E'(u)| \geq \ell(d(u)) - 1$. This is clear if E' contains an edge $e \in E(u)$ of weight $c(e) = j$ for all j, $1 \leq j < \ell(d(u))$. If there is an edge $e_i \in E(u) - E'$ with weight $c(e_i) = i < \ell(d(u))$, then every nonempty layer V_j above layer $d(u) + i$ must contain the target of an edge in $E'(u)$. Assume otherwise; then there is an edge $e_j = (u, v) \notin E'$ with $v \in V_j$ for a $j > d(u) + i$. Define the simple weight function c' by

$$c'(e) := \begin{cases} c(e), & \text{if } e \notin \{e_i, e_j\}, \\ c(e_j), & \text{if } e = e_i, \\ c(e_i), & \text{if } e = e_j. \end{cases}$$

Then $c'(e) = c(e)$ for all $e \in E'$, and therefore the algorithm will output d, the distance function with respect to c, on input c' as well. However, $d(v) = j > d(u) + i = d(u) + c'(e_j)$ for $e_j = (u, v)$, which shows that d is incorrect with respect to c'.

We choose $i = \min\{c(e)\ ;\ e \in E(u) - E'\}$. Note that all edges in $E(u)$ with targets in layer V_j, $j > d(u) + i$, must have weight at least $j - d(u) > i$ by the correctness of the algorithm. Hence, $|E'(u)| \geq i - 1 + \ell(d(u) + i) \geq \ell(d(u)) - 1$. ∎

Table I shows the distribution of vertices over distances for a (typical) simple weight function on a graph of $n = 10{,}000$ vertices. Most vertices have distance about 14 ($\approx \log n$) from the source, but there are vertices that have distance as much as 24 ($\approx 2 \log n$). By the argument of Lemma 5, we can guess that any (correct) algorithm must inquire about $\Omega(n \log n)$ edges.

In the remainder of this section, we make this argument more precise. We derive a lower bound of $\Omega(n \log n)$ on the expected value of $\sum_{u \in V} \ell(d(u))$ for random simple weight functions c. More generally, we show that any algorithm has to ask $\Omega(n \log n)$ questions with high probability.

Our proof strategy is as follows. The lower bound given by Lemma 5 depends only on the distance function d. For random simple weight functions, we reinterpret the calculation of d and the construction of the layers V_i as the outcome of a

TABLE I A typical distribution of vertices over distances for $n = 10{,}000$

Distance d	0	1	2	3	4	5	6	7	8	9	10	11	12
# Vertices	1	1	2	4	8	16	32	64	120	237	449	796	1306
Distance d	13	14	15	16	17	18	19	20	21	22	23	24	
# Vertices	1845	1952	1562	910	415	181	58	20	16	2	1	2	

random labeling process. Note that a random simple weight function is given by n independent permutations of V, one for each vertex. The ith vertex of the permutation for vertex v is the target of the edge with weight i and source v. The labeling process proceeds in *stages*. In the zeroth stage, V_0 is set to $\{s\}$ and $d(s)$ is set to 0. In the ith stage, $i \geq 1$, each vertex $v \in S^{(i)} = \bigcup_{0 \leq j < i} V_j$ picks the $(i - d(v))$th vertex in its adjacency list. Note that each vertex that v has not yet seen is equally likely to occur. The newly reached vertices are put into V_i and their d-values are set to i. Instead of fixing the n permutations before-hand, we may also view them as being fixed on-line (principle of deferred decisions). This leads to the following re-interpretation of the random labeling process: In the ith stage, each vertex in $S^{(i)} = \bigcup_{0 \leq j < i} V_j$ chooses a vertex uniformly and independently at random from the set of vertices it has not yet seen. The labeling process stops when $S^{(k)} = V$ for some k.

A related process was considered by Frieze and Grimmett in [7]. They assumed that each vertex in $S^{(i)}$ chooses a vertex uniformly and independently at random from the set of *all* vertices. If D_A denotes the number of stages taken by their version of the process, then it is clear that D_A stochastically dominates D, i.e., for all k, $\Pr(D > k) \leq \Pr(D_A > k)$. Frieze and Grimmett prove in [7] that D_A (and hence D) is $O(\log n)$ with high probability. However, we need a lower bound on D and therefore their result is of no use to us. (Nevertheless, our proof strategy was inspired by theirs.)

The random labeling process is said to be in *state* j if $|S^{(i)}| = j$. We call stage i of the labeling process *central* if $n/e \leq |S^{(i)}| \leq n - \sqrt{n}$. Layers constructed in central stages are called central.

Our proof will proceed in two steps. First, we show in Lemma 6 that there are $\Omega(\log n)$ central stages with high probability. Second, we prove in Lemma 7 that each central stage gives rise to a nonempty layer with high probability.

Lemma 6. *With high probability, the labeling process has* $\Omega(\log n)$ *central stages.*

Proof. For a random simple weight function c, let i_0 be the first central stage with respect to c. Then $n/e \leq |S^{(i_0)}| \leq 2n/e$, since $|S^{(i+1)}| \leq 2|S^{(i)}|$ for any $i \geq 0$. We will show that $|S^{(i_0+k)}| \leq n - \sqrt{n}$ with high probability for $k = (\ln n)/17$. Let $U = V - S^{(i_0)}$ be the set of vertices that are still unlabeled after stage i_0. Note that $|U| \geq (e-2)n/e \geq n/4$.

Let us condition on $m = |U|$. Construct an $n \times m$ matrix A with 0-1 entries as follows. The rows correspond to the vertices in V and the columns correspond to the vertices in U; entry a_{vu} is 1 if and only if the edge (v, u) is among the k shortest edges in v's adjacency list whose head is an element of U. Let $f(A)$ be the number of all-zero columns in A. Then $|S^{(i_0+k)}| \leq n - f(A)$ because no vertex in U corresponding to an all-zero column will be labeled in the k stages following stage i_0. Since A models a process in which all vertices (and not only those that are currently labeled) are allowed to label new vertices, and in which each vertex is prevented from choosing vertices that have been labeled by other vertices before stage i_0, $f(A)$ may seem to be a rather crude lower bound on $|V - S^{(i_0+k)}|$. However, we will now prove that even $f(A) \geq \sqrt{n}$ with high probability.

A row of A is a random 0-1 vector of length m with exactly k ones. Moreover, the row entries $A_i.$, $1 \le i \le n$, are independent random variables, and if A, A' differ only in a single row, then $|f(A) - f(A')| \le k$. Hence, by Azuma's inequality (Lemma 3), we get the following tail estimate for $f(A) = f(A_1., \ldots, A_n.)$,

$$\Pr\big(f(A) \le E[f(A)]/2\big) \le 2\exp\big(-E[f(A)]^2/(2nk^2)\big). \tag{11}$$

The probability that a fixed column is all-zero is $(1 - k/m)^n$; therefore,

$$E[f(A)] = m\left(1 - \frac{k}{m}\right)^n. \tag{12}$$

Remember that $m = |U| \ge n/4$ and $k = (\ln n)/17$; since $(1 - 1/x)^x \ge e^{-2}$ for large enough x, we get from (12) that

$$E[f(A)] \ge me^{-2kn/m} \ge \tfrac{1}{4}n^{1-8/17} > 2\sqrt{n} \tag{13}$$

for large enough n, where $E[f(A)]$ is conditioned on $|U| = m$. However, the lower bound in (13) is independent of m. Hence,

$$\Pr\big(f(A) < \sqrt{n}\,\big) \le 2\exp\big(-\Theta(n^{1/17})/(\ln n)^2\big) = O(n^{-C})$$

for any fixed $C > 0$ and large enough n. Since $|S^{(i_0+k)}|$ is increasing in k, we have thus proved that, with high probability, it will take $\Omega(\ln n) = \Omega(\log n)$ central stages to label all but $\sqrt{n}$ vertices. ∎

Remark. $f(A)$ can be expressed as the sum of 0-1 indicator variables C_j, $1 \le j \le m$, where $C_j = 1$ if and only if column j of A is all-zero. The C_j's are *not* independent; for example, $\sum_j C_j \le m - k$. However, they can be shown to be *negatively associated*, i.e., negatively dependent in a strong sense; see [3] for a proof. This property of the C_j's suffices to prove analogues of the Chernoff–Hoeffding bound from Lemma 2 for the left tail of the distribution of $f(A)$, and (11) could be replaced by the sharper $\Pr(f(A) \le E[f(A)]/2) \le e^{-E[f(A)]/8}$.

Lemma 7. *With high probability, each central layer contains at least one vertex.*

Proof. Suppose the process is in state j at the beginning of stage i. For any vertex in $S^{(i)}$, the probability of selecting a vertex in $S^{(i)}$ during this stage is $\le j/n$. Therefore, the next layer will remain empty with probability $\le (j/n)^j$. Note that $x \mapsto (x/n)^x$ is an increasing function for $x > n/e$.

Let B denote the event that at least one central layer remains empty. By the estimates provided in the preceding paragraph,

$$\Pr(B) \le \sum_{j=n/e}^{n-\sqrt{n}} \left(\frac{j}{n}\right)^j \le n\left(\frac{n-\sqrt{n}}{n}\right)^{n-\sqrt{n}} \le ne^{-\sqrt{n}+1} = O(n^{-C})$$

for sufficiently large n. ∎

Theorem 3. *Any algorithm for the single-source shortest-path problem has complexity* $\Omega(n \log n)$ *with high probability on random simple weight functions.*

Proof. Suppose that i is the first central stage of the labeling process; as before, let $S^{(i)}$ denote the set of vertices that have already been labeled up to this stage. By Lemma 6, with high probability, the process has $\Omega(\log n)$ central layers. Lemma 7 tells us that all these layers will be nonempty with high probability. With the notation introduced in the discussion of the labeling process, this reads

$$\sum_{u \in S^{(i)}} (\ell(d(u)) - 1) = \Omega(n \log n) \qquad \text{with high probability.}$$

By Lemma 5, the left-hand side term is a lower bound on the complexity of any shortest-path algorithm. ■

ACKNOWLEDGMENTS

We learned from discussions with Paul Spirakis that analyzing the algorithm by Moffat and Takaoka [14] is not as easy as it might appear at first glance. The remarks of Alistair Moffat and of an anonymous referee for ICALP'94 allowed considerable simplification of our proofs of Theorems 1 and 2. We are grateful to the anonymous referees of this journal for their comments, especially for urging us to give a concise and sound presentation of both theorems. Rudolf Fleischer suggested the use of Fibonacci heaps in the implementation of the algorithm. Finally, numerous discussions with Hannah Bast and Torben Hagerup were particularly insightful, and conversations with Devdatt Dubhashi and Shiva Chaudhuri helped to clarify our ideas.

This work was supported by the ESPRIT II Basic Research Actions Program of the EC under contract number 7141 (Project ALCOM II) and the BMFT-Project "Softwareökonomie und Softwaresicherheit" ITS 9103. A preliminary version of this paper was presented at the Third Annual European Symposium on Algorithms, Corfu, Greece, 1995, [12]. Volker Priebe's research was supported by a Graduiertenkolleg graduate fellowship of the Deutsche Forschungsgemeinschaft.

REFERENCES

[1] P. A. Bloniarz, A shortest-path algorithm with expected time $O(n^2 \log n \log^* n)$, *SIAM J. Comput.*, **12**, 588–600 (1983).

[2] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.*, **1**, 269–271 (1959).

[3] D. Dubhashi, V. Priebe, and D. Ranjan, Negative dependence through the FKG inequality, Research Report MPI-I-96-1-020, Max-Planck-Institut für Informatik, Saarbrücken, August 1996.

[4] R. W. Floyd, Algorithm 97: shortest path, *Commun. ACM*, **5**, 345 (1962).

[5] M. L. Fredman, New bounds on the complexity of the shortest path problem, *SIAM J. Comput.*, **5**, 83–89 (1976).

[6] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. ACM*, **34**, 596–615 (1987).

[7] A. M. Frieze and G. R. Grimmett, The shortest-path problem for graphs with random arc-lengths, *Discrete Appl. Math.*, **10**, 57–77 (1985).

[8] W. Hoeffding, Probability inequalities for sums of bounded random variables, *J. Am. Stat. Assoc.*, **58**, 13–30 (1963).

[9] M. Hofri, *Analysis of Algorithms: Computational Methods and Mathematical Tools*, Oxford University Press, Oxford, 1995.

[10] D. R. Karger, D. Koller, and S. J. Phillips, Finding the hidden path: time bounds for all-pairs shortest paths, *SIAM J. Comput.*, **22**, 1199–1217 (1993).

[11] C. McDiarmid, On the method of bounded differences, in *Surveys in Combinatorics, 1989*, (J. Siemons, Ed.), London Mathematical Society Lecture Note Series 141, Cambridge University Press, Cambridge, 1989, pp. 148–188.

[12] K. Mehlhorn and V. Priebe, On the all-pairs shortest path algorithm of Moffat and Takaoka, in *Algorithms—ESA '95* (P. Spirakis, Ed.), Lecture Notes in Computer Science 979, Springer-Verlag, Berlin, 1995, pp. 185–198.

[13] A. Moffat and T. Takaoka, An all pairs shortest path algorithm with expected running time $O(n^2 \log n)$, *Proc. 26th Annual Symposium on Foundations of Computer Science*, Portland, OR, 1985, pp. 101–105.

[14] A. Moffat and T. Takaoka, An all pairs shortest path algorithm with expected time $O(n^2 \log n)$, *SIAM J. Comput.*, **16**, 1023–1031 (1987).

[15] R. Raman, The power of collision: randomized parallel algorithms for chaining and integer sorting, in *Foundations of Software Technology and Theoretical Computer Science*, (K. V. Nori and C. E. Veni Madhavan, Eds.), Lecture Notes in Computer Science 472, Springer-Verlag, Berlin, 1990, pp. 161–175.

[16] P. M. Spira, A new algorithm for finding all shortest paths in a graph of positive arcs in average time $O(n^2 \log^2 n)$, *SIAM J. Comput.*, **2**, 28–32 (1973).

[17] T. Takaoka, A new upper bound on the complexity of the all pairs shortest path problem, *Inf. Process. Lett.*, **43**, 195–199 (1992).

[18] H. Thorisson, Coupling methods in probability theory, *Scand. J. Stat.*, **22,** 159–182 (1995).