All-Pairs Bottleneck Paths in Vertex Weighted Graphs

Asaf Shapira * Raphael Yuster† Uri Zwick ‡

Abstract

Let $G = (V, E, w)$ be a directed graph, where $w : V \to \mathbb{R}$ is an arbitrary weight function defined on its vertices. The *bottleneck* weight, or the *capacity*, of a path is the smallest weight of a vertex on the path. For two vertices u, v the bottleneck weight, or the capacity, from u to v, denoted $c(u, v)$, is the maximum bottleneck weight of a path from u to v. In the *All-Pairs Bottleneck Paths* (APBP) problem we have to find the bottleneck weights for all ordered pairs of vertices. Our main result is an $O(n^{2.575})$ time algorithm for the APBP problem. The exponent is derived from the exponent of fast matrix multiplication. Our algorithm is the first sub-cubic algorithm for this problem. Unlike the sub-cubic algorithm for the *all-pairs shortest paths* (APSP) problem, that only applies to bounded (or relatively small) integer edge or vertex weights, the algorithm presented for APBP problem works for arbitrary large vertex weights.

The APBP problem has numerous applications, and several interesting problems that have recently attracted attention can be reduced to it, with *no asymptotic loss* in the running times of the known algorithms for these problems. Some examples are a result of Vassilevska and Williams [STOC 2006] on finding a triangle of maximum weight, a result of Bender et al. [SODA 2001] on computing least common ancestors in DAGs and a result of Kowaluk and Lingas [ICALP 2005] on finding maximum witnesses for boolean matrix multiplication. Thus, the APBP problem provides a uniform framework for these applications. For some of these problems, we can in fact show that their complexity is *equivalent* to that of the APBP problem.

A slight modification of our algorithm enables us to compute *shortest paths* of maximum bottleneck weight. Let $d(u, v)$ denote the (unweighted) distance from u to v, and let $sc(u, v)$ denote the maximum bottleneck weight of a path from u to v having length $d(u, v)$. The *All-Pairs Bottleneck Shortest Paths* (APBSP) problem is to compute $sc(u, v)$ for all ordered pairs of vertices. We present an algorithm for the APBSP problem whose running time is $O(n^{2.86})$.

1 Introduction

Finding optimal paths between pairs of vertices is one of the most fundamental algorithmic graph problems. There are various measures for optimality, depending on the problem to be solved. Commonly, one searches for *shortest* paths as in the *Single Source Shortest Paths* (SSSP) problem and the *All-Pairs Shortest Paths* (APSP) problem. In many other situations, such as in maximum flow algorithms, one searches for paths with maximum bottleneck value, or capacity. In this paper, we consider the fundamental problem of *All-Pairs Bottleneck Paths* (APBP) in vertex weighted graphs. Given a directed graph $G = (V, E, w)$ where w is an arbitrary real weight function defined on its vertices or edges, the *bottleneck* weight (or *capacity*) of a path is the smallest weight of a vertex (respectively edge) on the path. In the vertex weighted case we have the *closed* variant, in which the weights of the endpoints of a path are taken into account, and the *open* variant, in which they are not. We refer to the variants as *edge-APBP*, *open-APBP* and *closed-APBP*. In each one of these variants, we let $c(u, v)$ be the maximum bottleneck weight, or maximum capacity, of a path from u to v. In each one of the variants, the goal is to compute the bottleneck weight for all ordered pairs of vertices. As we will show shortly, many problems that have been recently studied reduce to solving an APBP problem with weighted vertices. Let us just mention here one illustrative example: consider the goal of planning routes for trucks going between different cities. One of the main constraints in planning the trucks' routes is the height of the bridges under which the trucks should pass. By solving the APBP problem on a graph in which the vertices are the cities and the bridges, and the weight of a vertex is the height of its bridge, or ∞ if it is a city, one can find the maximum height of a truck that can be sent from city A to city B, for any two cities A and B.

Let n and m denote, respectively, the number of vertices and edges of a given graph. The edge-APBP problem can be easily solved in $O(mn)$ time using, as a main procedure, a variant of Dijkstra's algorithm. For dense graphs this results in a cubic $O(n^3)$ time algorithm. A truly sub-cubic algorithm for the edge-APBP problem would imply, in particular, a truly sub-cubic algorithm for min-max products of two real valued matrices (see Section 6 for more details). Until the present paper, it was not known whether open-APBP or closed-APBP, i.e., the vertex weighted versions of the APBP problem, are easier than edge-APBP. Our main result in this paper is the first truly sub-cubic algorithm for both the closed-APBP and open-APBP problems.

*Microsoft Research. Email: asafico@tau.ac.il.

†Department of Mathematics, University of Haifa, Haifa 31905, Israel. E–mail: raphy@research.haifa.ac.il

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

We mention briefly that bottleneck versions of many other graph optimization problems were considered before. The bottleneck spanning tree problem, for example, can be easily solved, deterministically, in $O(m+n)$ time. Gabow and Tarjan [11] considered bottleneck versions of the directed spanning tree problem and the weighted matching problem.

Before presenting our main result, we need a few definitions. Let $\omega(r,s,t)$ be the minimal exponent for which the Boolean product of an $n^r \times n^s$ Boolean matrix and an $n^s \times n^t$ Boolean matrix can be computed in $O(n^{\omega(r,s,t)})$ time. The exponent $\omega = \omega(1,1,1)$ is usually called the exponent of fast Boolean matrix multiplication. Coppersmith and Winograd [6] proved that $\omega < 2.376$. Let μ be the solution to $\omega(1,\mu,1) = 1+2\mu$. The results from [5] and [12] show that $\mu < 0.575$.

1.1 The new results For a directed graph with n vertices, an APBP matrix is an $n \times n$ matrix C with rows and columns indexed by the vertices, and $C(u,v) = c(u,v)$. If there is no path from u to v we let $c(u,v) = -\infty$. In open-APBP we define $c(u,v) = \infty$ if $(u,v) \in E$ or if $u = v$. The main result of this paper is:

THEOREM 1.1. (MAIN RESULT) *Let $G = (V,E,w)$ be a directed graph with $w : V \to \mathbb{R}$. There is an algorithm that computes an open-APBP matrix and a closed-APBP matrix in $O(n^{2+\mu}) = O(n^{2.575})$ time. The algorithm also computes a data structure so that, given two vertices u, v with $c(u,v) > -\infty$, a path from u to v, having capacity $c(u,v)$ can be constructed in time proportional to its length.*

The APBP problem has several interesting applications. In Section 2 we describe three such applications, and show that they can be reduced to special cases of APBP. The first application, considered in [3], is *All-Pairs Lowest Common Ancestors* in directed acyclic graphs. The fastest algorithm for this problem, due to Kowaluk and Lingas [13, 14], runs in $O(n^{2+\mu})$ time. We show that this problem can be easily reduced to a special case of closed-APBP. The second application, first considered by Vassilevska and Williams [18], is the *Largest Weighted Triangle* problem. Given a real vertex-weighted graph, find a triangle (if one exists) with maximum total weight. The fastest algorithm for this problem, due to Vassilevska, Williams and Yuster [19], runs in $O(n^{2+\mu})$ time. Clearly, if we can find, for each pair of vertices, the maximum total weight of a 2-path, i.e., a path of length 2, connecting them, we can also find a largest weight triangle. The latter problem can be easily reduced to a special case of open-APBP. The third application, which is in fact used as a subroutine in the proof of Theorem 1.1, is *Maximum Witnesses for Boolean Matrix Multiplication* (MWBMM) (see a definition in the next section). Again, the fastest algorithm for this problem, given in [13, 14], runs in $O(n^{2+\mu})$ time. This problem turns out to be a special case of open-APBP. In fact, we will show in Section 2, that all the following four problems are computationally equivalent.

THEOREM 1.2. *The following problems are computationally equivalent (up to constant factors).*

1. *Open-APBP.*
2. *Closed-APBP.*
3. *Maximum Witnesses for Boolean Matrix Multiplication.*
4. *All-Pairs Maximum Weight 2-paths in Vertex Weighted Graphs.*

When considering the variants of APBP in *undirected* graphs, things change dramatically. It turns out that both closed-APBP and edge-APBP can be solved in optimal $\Theta(n^2)$ time, while open-APBP is probably more difficult, and is equivalent to the problems in Theorem 1.2.

THEOREM 1.3. *For undirected graphs:*

1. *The edge-APBP problem can be solved in time $\Theta(n^2)$.*
2. *The closed-APBP problem can be solved in time $\Theta(n^2)$.*
3. *The open-APBP problem is computationally equivalent to open-APBP in directed graphs, and thus to all the problems mentioned in Theorem 1.2.*

When computing shortest paths between pairs of vertices, there is, in many cases, more than one solution. Which solution (namely, which shortest path) should be preferred? Such problems have been considered in, e.g., [20]. In Section 4, we look, among all shortest paths between a pair of vertices, for the one having maximum bottleneck weight. More formally, if $G = (V,E,w)$ and $w : V \to \mathbb{R}$, let $d(u,v)$ denote the (unweighted) distance from u to v, and let $sc(u,v)$ be the maximum capacity of a path from u to v having length $d(u,v)$. The APBSP problem is to compute $sc(u,v)$ for all ordered pairs of vertices u, v (again, we have two versions: open-APBSP and closed-APBSP). Zwick [21] gave an $O(n^{2+\mu})$ time algorithm for unweighted APSP. (Note, however, that we do not claim that unweighted APSP is computationally equivalent to the problems of Theorem 1.2.) Theorem 1.1 gives an $O(n^{2+\mu})$ time algorithm for open-APBP and closed-APBP. However,

when combined together, we are unable to retain this running time, though we can still obtain a truly subcubic algorithm.

THEOREM 1.4. *Let $G = (V, E, w)$ be a directed graph with $w : V \to \mathbb{R}$. There is an algorithm for open-APBSP and closed-APBSP whose running time is $\tilde{O}(n^{(8+\mu)/3}) = O(n^{2.86})$.*

1.2 Organization and overview The rest of the paper is organized as follows: In Section 2 we prove Theorem 1.1. The main idea is to first solve a special case of the problem in which we only consider paths of constant length, and then use recursion to reduce the general case to this special case. In Section 3 we prove Theorem 1.2 using a series of reductions. In Section 1.3 we consider the undirected versions of the problem and prove Theorem 1.3. In Section 5 we consider the problem of finding shortest paths with maximum bottleneck weight. The proof of Theorem 1.4 combines the main idea of the proof of Theorem 1.1 with a sampling technique used by Zwick [21]. Section 6 contains some concluding remarks and open problems. All the algorithms presented in this paper are strongly polynomial. The only operation allowed on real numbers is comparison. Due to space limitations many details are omitted from this version of the paper and will appear in its full version.

2 All-Pairs Bottleneck Paths in Vertex Weighted Graphs

An important ingredient in the proof of Theorem 1.1 is a procedure for computing maximum witnesses for Boolean matrix multiplication. We use this procedure to solve the APBP problem in directed graphs, when restricted to paths of constant length. We then show how to solve the APBP problem for general directed graphs by using recursion together with a reduction to the case of bounded maximum path length. We shall prove Theorem 1.1 for the closed-APBP case. The proof of Theorem 1.2, appearing in the next section, supplies, in particular, two easy reductions between open-APBP and closed-APBP, showing their computational equivalence.

2.1 Maximum witnesses for Boolean matrix multiplication Let A and B be two Boolean $n \times n$ matrices. The *maximum witness matrix* of the product $C = AB$ is the $n \times n$ matrix W defined as follows. If $C(i, j) = 0$, then $W(i, j) = 0$. Otherwise, $W(i, j)$ is the largest index k such that $A(i, k) = B(k, j) = 1$.

A simple randomized algorithm for computing (not necessarily maximum) witnesses for Boolean matrix multiplication, in essentially the same time required to perform the product, is given by Seidel [17]. Alon and Naor [2] gave a deterministic algorithm for the problem. An alternative, slightly slower, deterministic algorithm was given by Galil and Margalit [10]. However, computing the maximum witness matrix seems to be a more difficult problem. Kowaluk and Lingas [13, 14] proved the following.

THEOREM 2.1. (KOWALUK AND LINGAS [13, 14]) *A maximum witness matrix for the product of two $n \times n$ Boolean matrices can be computed in $O(n^{2+\mu}) = O(n^{2.575})$ time.*

2.2 Constant length paths Throughout the paper we use $A \wedge B$ to denote the bitwise logical and of two Boolean matrices A and B of the same dimensions. Let $G = (V, E, w)$ be a vertex-weighted directed graph. Since sorting is not a bottleneck for our algorithm, we may assume, w.l.o.g., that $V = \{1, \ldots, n\}$ and that $w(i) \leq w(i + 1)$ for $i = 1, \ldots, n - 1$. Let $c_s(u, v)$ be the maximum bottleneck weight of a path from u to v in G whose length is at most s. For completeness, $c_s(u, u) = w(u)$ and $c_s(u, v) = -\infty$ if there is no path from u to v whose length is at most s. In this section we show that for any constant positive integer t, we can compute $c_t(u, v)$ for all ordered pairs of vertices $u, v \in V$ in $O(n^{2+\mu})$ time.

For each $s = 0, \ldots, t$ we define two $n \times n$ Boolean matrices, P_s and Q_s as follows. $P_s(u, v) = 1$ if there is a path from u to v, of length *at most* s, in which v has minimum weight. Otherwise, $P_s(u, v) = 0$. $Q_s(u, v) = 1$ if there is a path from u to v, of length at most s, in which u has minimum weight. Otherwise, $Q_s(u, v) = 0$.

Let A be the Boolean adjacency matrix of G, with 1's on the diagonal. Let B be the Boolean matrix with $B(u, v) = 1$ if and only if $w(u) \geq w(v)$. Clearly, $P_0 = Q_0 = I$. P_1 and Q_1 are easily determined from A and B by setting $P_1 = A \wedge B$ and $Q_1 = A \wedge B^T$. Suppose we have computed P_{s-1} and Q_{s-1}. We show how to compute P_s and Q_s. We claim that $P_s = AP_{s-1} \wedge B$. Indeed, let $p = (u, x, \ldots, v)$ be a path of length at most s from u to v in which v has minimum weight. Then, $A(u, x) = 1$ because (u, x) is an edge of G, $P_{s-1}(x, v) = 1$ because $p = (x, \ldots, v)$ is a path of length at most $s-1$ from x to v in which v has minimum weight, and finally $B(u, v) = 1$. It is easy to see that the converse holds as well. Similarly, we have $Q_s = Q_{s-1}A \wedge B^T$.

We have shown how to compute P_s and Q_s for $s = 0, \ldots, t$ in $O(tn^\omega)$ time, and hence in $O(n^\omega)$ time when t is a constant. Now, for $s = 0, \ldots, t$ let W_s be a maximum witness matrix for the Boolean product P_sQ_{t-s}. By Theorem 2.1, all the matrices $W_0, \ldots, W_t$ can be computed in $O(tn^{2+\mu})$ time, and

hence in $O(n^{2+\mu})$ time when t is a constant. Having computed them, we claim that for each ordered pair u, v we can determine $c_t(u, v)$ in $O(t)$ time by setting $w(0) = -\infty$ and

$$c_t(u, v) = w(\max_{0 \le s \le t} W_s(u, v)) \ . \tag{2.1}$$

Indeed, there is no path of length at most t from u to v if and only if $W_s(u, v) = 0$ for all $s = 0, \ldots, t$. In this case, we correctly define $c_t(u, v) = -\infty$. Otherwise, suppose $c_t(u, v) = w^*$, let p be a path from u to v of length at most t having capacity w^*, and suppose y is a vertex on p with $w(y) = w^*$. The path p is a concatenation of two paths $p_1 = (u, \ldots, y)$ and $p_2 = (y, \ldots, v)$ where p_1 has length s and p_2 has length at most $t - s$, where $0 \le s \le t$ (possibly $u = y$ or $y = v$). Hence, $P_s(u, y) = 1$ and $Q_{t-s}(y, v) = 1$. This means that $W_s(u, v) \ge y$, and therefore our choice for the value of $c_t(u, v)$ in (2.1) yields a capacity of at most $w(y) = w^*$. For the other direction, if our choice in (2.1) yields value w^*, then there is a vertex y with $w(y) = w^*$ and two paths p_1, which connects u to y, and p_2, which connects y to v, of total length at most t such that y has minimum weight on p_1 and p_2. This means that indeed $c_t(u, v) \le w^*$. We have thus proved the following.

COROLLARY 2.1. *For every integer t there is an $O(tn^{2+\mu})$ time algorithm for computing the values $c_t(u, v)$ for All-Pairs of vertices in a graph on n vertices.*

2.3 Proof of Theorem 1.1 Let $G = (V, E, w)$ be a vertex-weighted directed graph. As in the previous sub-section, we assume that $V = \{1, \ldots, n\}$ and that $w(i) \le w(i + 1)$ for $i = 1, \ldots, n - 1$. We may and will assume that n is a power of 2, as this does not affect the asymptotic nature of our results. We need to compute the bottleneck weight $c(u, v)$ for all ordered pairs u, v. Let $A = \{1, \ldots, n/2\}$ and $B = V \setminus A$. Let us denote by $c_{G'}(u, v)$ the bottleneck weight of the pair (u, v) in a subgraph G' of G. Let G_1 be the subgraph of G induced by B. We recursively solve the closed-APBP problem in G_1. Note that if for $u, v \in B$ we have $c_{G_1}(u, v) > -\infty$, then this is the correct value of $c(u, v) = c_{G_1}(u, v)$. Let $G_2 = (A, E_2)$, where $(u, v) \in E_2$ if and only if $(u, v) \in E$ or if there is path from u to v in G all whose internal vertices are in B. We can clearly construct G_2 in $O(n^\omega)$ time using Boolean matrix multiplication, as in the classical transitive closure problem (the weights play no role here). We then recursively solve the closed-APBP problem in G_2. It is clear that for any $u, v \in A$ we have $c(u, v) = c_{G_2}(u, v)$.

It remains to compute $c(u, v)$ for $(u, v) \in A \times B$, $(u, v) \in B \times A$, and for $(u, v) \in B \times B$ for which $c_{G_1}(u, v) = -\infty$. To do that, we create a graph $G' = (V', E')$ as follows. The vertex set V' is the disjoint union of five vertex sets $V' = B_1 \cup A_2 \cup A_3 \cup A_4 \cup B_5$ where B_1, B_5 are copies of B and A_2, A_3, A_4 are copies of A. Notice that $|V'| = 2.5n$. For every $u \in V$, we let u_i be the copy of u in A_i or B_i. We let $w(u_i) = w(u)$.

For every $u \in B$ and $v \in A$, we have $(u_1, v_2) \in E'$ if and only if there is a path from u to v in G whose internal vertices are all from B. Again, using Boolean matrix multiplication we can determine the edges from B_1 to A_2 in $O(n^\omega)$ time. For every $u, v \in A$, we have $(u_2, v_3) \in E'$ if and only if $c(u, v) = w(v)$ (recall that we already know $c(u, v)$ at this point). For every $u, v \in A$, we have $(u_3, v_4) \in E'$ if and only if $c(u, v) = w(u)$. Finally, for every $u \in A$ and $v \in B$ we have $(u_4, v_5) \in E'$ if and only if there is a path from u to v in G whose internal vertices are all from B. Note that G' can be constructed in $O(n^\omega)$ time.

All the paths in G' have length at most 4. By Corollary 2.1 we can compute a closed-APBP matrix for G' in $O(n^{2+\mu})$ time. Let $u \in B$ and $v \in A$ and consider a path from u to v with maximum capacity, whose first vertex in A is x and whose vertex with minimum weight is y. Note that $y \in A$. By definition, we have $(u_1, x_2), (x_2, y_3), (y_3, v_4) \in E'$. Therefore $c_{G'}(u_1, v_4) \le w(y_3) = w(y) = c(u, v)$. It is also easy to see that $c_{G'}(u_1, v_4) \ge c(u, v)$ and hence $c(u, v) = c_{G'}(u_1, v_4)$. Let $u, v \in B$ be two vertices for which $c_{G_1}(u, v) = \infty$. By the same reasoning we get that $c(u, v) = c_{G'}(u_1, v_5)$. Similarly, if $u \in A$ and $v \in B$, then $c(u, v) = c_{G'}(u_2, v_5)$.

The running time of the algorithm satisfies the recursion $f(n) \le O(n^{2+\mu}) + 2f(n/2)$. Therefore $f(n) = O(n^{2+\mu})$, as required. ∎

It is not difficult to modify the algorithm so as to obtain a data structure representing the maximum capacity paths between all pairs of vertices, without affecting the running time. The details will appear in the full version of this paper.

3 Applications and Computational Equivalence

In this section we prove Theorem 1.2 using a sequence of reductions. Problem A reduces to problem B if a solution of B in $f(n)$ time implies a solution of A in $O(f(n))$ time.

LEMMA 3.1. *Closed-APBP reduces to open-APBP.*

Proof: Suppose $G = (V, E, w)$ is a directed graph with $w : V \to \mathbb{R}$. We reduce closed-APBP to open-APBP in $O(n^2)$ time by creating a graph $G' = (V \cup V_{in} \cup V_{out}, E \cup E', w')$ as follows. Associate with each $v \in V$ two new vertices $v_{in} \in V_{in}$ and $v_{out} \in V_{out}$. Let $E' = \{(v, v_{in}), (v_{out}, v) \mid v \in V\}$. For every $v \in V$, let

$w'(v) = w(v)$, and let $w'(v_{in})$ and $w'(v_{out})$ be arbitrary. If C' is the open-APBP matrix of G' and C is the closed-APBP matrix of G then clearly $C(u,v) = C'(u_{out}, v_{in})$. ■

LEMMA 3.2. *Open-APBP reduces to closed-APBP.*

Proof: We reduce open-APBP to closed-APBP in $O(n^2)$ time by creating a graph $G' = (V \cup V_{in} \cup V_{out}, E \cup E', w')$ as follows. Associate with each $v \in V$ two new vertices $v_{in} \in V_{in}$ and $v_{out} \in V_{out}$. Let $E' = \{(u, v_{in}), (u_{out}, v) \mid (u,v) \in E\}$. For every $v \in V$, let $w'(v) = w(v)$ and $w'(v_{in}) = w'(v_{out}) = \infty$. If C' is the closed-APBP matrix of G' and C is the open-APBP matrix of G then for every $(u,v) \notin E$ we have $C(u,v) = C'(u_{out}, v_{in})$. ■

Let $f(n) = O(n^\omega)$ be the complexity of Boolean matrix multiplication, and of transitive closure; the two are known to be equivalent [7, 9, 15]. Let $g(n) = O(n^{2+\mu})$ be the complexity of computing maximum Witnesses for Boolean matrix multiplication. By definition, $f(n) \leq g(n)$, and the proof of Theorem 1.1 gives the following corollary.

LEMMA 3.3. *Closed-APBP reduces to MWBMM*

We now show that the converse is also true.

LEMMA 3.4. *MWBMM reduces to Open-APBP.*

Proof: Let A and B be two Boolean $n \times n$ matrices. We create a vertex weighted 3-layered graph $G = (V, E, w)$ with $V = V_1 \cup V_2 \cup V_3$, where $V_1 = \{x_1, \ldots, x_n\}$, $V_2 = \{y_1, \ldots, y_n\}$, and $V_3 = \{z_1, \ldots, z_n\}$. We let $E = \{(x_i, y_k) \mid A(i,k) = 1\} \cup \{(y_k, z_j) \mid B(k,j) = 1\}$. For every $1 \leq k \leq n$, we let $w(y_k) = k$. The weights $w(x_i)$ and $w(z_j)$, for $1 \leq i, j \leq n$ are arbitrary. If C is the open-APBP of G, then for every $1 \leq i, j \leq n$, if $C(x_i, z_j) = -\infty$, let $W(i,j) = 0$, otherwise, let $W(i,j) = C(i,j)$. It is then easy to see that W is the maximum witness matrix for AB. ■

The above claims imply that the first three problems in Theorem 1.2 are equivalent. Let us conclude with the last problem. For a real vertex-weighted graph $G = (V, E, w)$, let $w_2(u,v)$ be the maximum total weight of a path of length 2 from u to v (the weights of u and v being included in the total). We conclude the proof of Theorem 1.2 with the following claim. As we have discussed in Section 1, this claim implies that we can use Theorem 1.1 to derive the main result of Vassilevska and Williams [18] on finding triangles of maximum weight.

LEMMA 3.5. *Finding all-pairs maximum weight 2-paths is equivalent to open-APBP.*

Proof: We start with a simple reduction from the all-pairs maximum weight 2-paths problem to the open-APBP. Let $G = (V, E, w)$ be a vertex weighted directed graph. Let $w_2(u,v)$ denote the maximum weight of a 2-path from u to v in G, or $-\infty$ if there is no such path. Let $G' = (V', E', w')$, where $V' = \{v_1, v_2, v_3 \mid v \in V\}$ and $E' = \{(u_1, v_2), (u_2, v_3) \mid (u,v) \in E\}$, and $w'(v_2) = w(v)$, for every $v \in V$. The weights $w'(v_1)$ and $w'(v_3)$ are arbitrary. Let C be the open-APBP matrix for G'. Clearly, $w_2(u,v) = w(u) + w(v) + C(u_1, v_3)$, for every $u, v \in V$. A reduction similar to the one given from MWBMM to open-APBP shows the converse. ■

Let $G = (V, E)$ be a directed acyclic graph (DAG). Vertex u is said to be an *ancestor* of vertex v in G if and only if there is a directed path from u to v in G. If u is an ancestor of v, then v is said to be a *descendant* of u in G. A *lowest common ancestor* (LCA) of two vertices u, v in a DAG is a vertex w that is an ancestor of both u and v, and such that no proper descendant of w is an ancestor of both u and v. Finding LCA's for all pairs of vertices in a tree, or, more generally, in a DAG, has many interesting applications (see, e.g., [3]). Kowaluk and Lingas [13] gave a reduction from the LCA problem in DAGs to the problem of finding maximum witnesses for Boolean matrix multiplication. The following is a simple reduction from the LCA problem to computing closed-APBP.

LEMMA 3.6. *Finding all-pairs LCA in DAGs reduces to computing closed-APBP (and thus to computing witnesses for boolean matrix multiplication).*

Proof: Let $G = (V, E)$ be a DAG. Assume that $V = \{1, \ldots, n\}$ and that $(i,j) \in E$ implies $i < j$ (topological sort can be done in linear time). Construct a vertex weighted graph $G' = (V', E', w')$ as follows: $V' = \{v_1, v_2 \mid v \in V\}$ and $E' = \{(v_1, u_1), (u_2, v_2) \mid (u,v) \in E\} \cup \{(v_1, v_2) \mid v \in V\}$. Note that G' has $2|V|$ vertices and $2|E| + |V|$ edges. Let $w'(v_1) = w'(v_2) = v$, for every $v \in V$. (Recall that $V = \{1, \ldots, n\}$.) Let C be a closed-APBP matrix for G'. If $C(v_1, u_2) = -\infty$, then u, v have no LCA in G. Otherwise $C(v_1, u_2)$ is an LCA of u, v in G, as required. ■

4 All Pairs Bottleneck Paths in Undirected Graphs

In this section we prove Theorem 1.3. We begin by showing that the open-APBP problem in undirected graphs is not easier than the open-APBP problem in directed graphs. By Theorem 1.2, it suffices to reduce MWBMM to open-APBP in undirected graphs. The reduction is very similar to the one used in the directed case, as shown in the previous section. The only

difference is that the weights of the vertices in V_1 and in V_3 are now defined to be 0, and that the edges are considered to be undirected. The solution of open-APBP for G implies the maximum witness matrix for AB. Indeed, $c(x_i, z_j) = 0$ or $c(x_i, z_j) = -\infty$ implies $W(i,j) = 0$ and $c(x_i, z_j) = k > 0$ implies $W(i,j) = k$.

Next, we prove that closed-APBP in undirected graphs is not harder than edge-APBP in undirected graphs. Suppose $G = (V, E, w)$ is an undirected graph and that $w : V \to \mathbb{R}$. We make G an edge-weighted graph by assigning $w(u,v) = \min\{w(u), w(v)\}$. It is easy to see that an edge-APBP matrix for the edge-weighted graph is also a closed-APBP matrix for the vertex-weighted graph.

In order to complete the proof of Theorem 1.3 we present a $\Theta(n^2)$ time algorithm for edge-APBP.

LEMMA 4.1. *The edge-APBP problem for undirected graphs can be solved in $\Theta(n^2)$ time.*

Proof: Let $G = (V, E, w)$ is an undirected graph, where $w : E \to \mathbb{R}$. We may assume, without loss of generality, that G is connected and that all edge weights are distinct. Let $T = (V, E')$ be a spanning tree of G of maximum weight. The tree T can be easily computed in $O(m + n \log n) = O(n^2)$ time using, say, Prim's algorithm [16, 8]. We claim that $c(u,v) = c_T(u,v)$, for every $u, v \in V$. Clearly, $c(u,v) \geq c_T(u,v)$, for every $u, v \in V$, as T is a subgraph of G. Let p be a path from u to v in G. Each edge e on p closes a cycle with some edge of T. As T is a maximum spanning tree, the weights of all the edges on this cycle are larger than the weight of e. (Recall that all edge weights are distinct.) Thus, every edge of p not in T can be replaced by a path in T of larger capacity. Thus, $c(u,v) \leq c_T(u,v)$, as required. Now, for every $u \in V$, the capacities $c_T(u,v)$, for all $v \in V$ can be easily computed in $O(n)$ time by running BFS. Thus, all the capacities $c_T(u,v)$, and hence $c(u,v)$, can be computed in $O(n^2)$ time, as claimed. ■

5 All-Pairs Bottleneck Shortest Paths

In this section we prove Theorem 1.4. We describe an algorithm for the closed-APBSP problem. An algorithm for the open-APBSP problem is obtained using the same reduction used to reduce the open-APBP problem to the closed-APBP problem. (See Lemma 3.2.) We start with a simple lemma showing that a variant of BFS solves the *single source* version of the closed-APBSP.

LEMMA 5.1. *Let $G = (V, E, w)$ be a directed graph, with $w : V \to \mathbb{R}$, and let $u \in V$. Then, there is an $O(m+n)$ time algorithm that computes $sc(u,v)$ for all $v \in V$.*

Proof: We start by setting $d(u,u) = 0$, $sc(u,u) = w(u)$, and place u in the queue. All other vertices are initialized with $d(u,v) = \infty$.

When x leaves the queue, for each $(x,y) \in E$ the following is performed. If $d(u,x) \geq d(u,y)$ we do nothing. Otherwise we have $d(u,x) < d(u,y)$. If $d(u,y) = \infty$ we set $d(u,y) = d(u,x) + 1$, place y in the queue and set $sc(u,y) = \min\{sc(u,x), w(y)\}$. Otherwise, we already have $d(u,y) = d(u,x) + 1$, so we do not place y in the queue, but we still update $sc(u,y) = \max\{sc(u,y), \min\{w(y), sc(u,x)\}\}$. The correctness and the claimed running time are straightforward to verify. ■

Let $G = (V, E, w)$ be a directed graph, with $w : V \to \mathbb{R}$, and $|V| = n$. Let $0 < t < n - 1$ be an integer parameter to be chosen later. Suppose $T \subset V$ has the property that for any pair of vertices u, v with $t < d(u,v) < n$, there is a path from u to v of length $d(u,v)$, containing a vertex from T. A set T having this property is called a *t-bridging set*. It was shown in [21], using a simple probabilistic argument, that a random subset of $\min\{n, 9n \log n/t\}$ vertices is a t-bridging set, with high probability. It is also shown in [21] that a t-bridging set of size $O(n \log n/t)$ can be found, deterministically, in $\tilde{O}(n^2 t)$ time. We need here a slight strengthening of the notion of t-bridging sets, as we would like the set T to hit specific shortest paths, namely shortest paths that have maximum capacity:

LEMMA 5.2. *Let $G = (V, E)$ be a directed graph on n vertices and let S be a set of paths connecting distinct pairs of vertices, where each path is of length at least t. Then, a randomly chosen set of vertices $B \subseteq V$, where each vertex of V is put in B, independently, with probability $\min\{(9 \log n)/t, 1\}$, contains, with high probability, at least one vertex from each path in S.*

Proof: If $(9 \log n)/t \geq 1$, then $B = V$ and the claim is obvious. Suppose, therefore, that $(9 \log n)/t < 1$. For a fixed path in S of length $k \geq t$, the probability that we do not choose any of the vertices of the path is at most $(1 - (9 \log n)/t)^t \ll n^{-2}$. As S contains at most n^2 paths, we conclude by the union bound that with high probability we "hit" all the paths in S. ■

Note that Lemma 5.2 implies that the set B can be chosen *without* knowing S explicitly. It is shown in [21] how to obtain a set B as in Lemma 5.2 *deterministically* in $\tilde{O}(tn^2)$ time, given a (partial) representation of S. Such a representation is available in our setting. The details of the deterministic procedure for constructing B will appear in the full version of this paper.

Proof of Theorem 1.4: We first show how to compute $sc(u,v)$ for all pairs u, v with $d(u,v) \leq t$. As in Section 2 we may assume that $V = \{1, \ldots, n\}$ and

that $w(i) \leq w(i+1)$ for $i = 1, \ldots, n-1$. Slightly modifying the definitions from Section 2, let $c_s(u,v)$ be the maximum capacity of a path from u to v whose length is *precisely* s. Clearly, $c_0(u,u) = w(u)$, and $c_0(u,v) = -\infty$ for $v \neq u$. For completeness, $c_s(u,v) = -\infty$ if there is no path from u to v whose length is precisely s.

For each $s = 0, \ldots, t$ we define two $n \times n$ Boolean matrices, P_s and Q_s as follows. $P_s(u,v) = 1$ if there is a path from u to v, of length precisely s, in which v has minimum weight. Otherwise, $P_s(u,v) = 0$. $Q_s(u,v) = 1$ if there is a path from u to v, of length precisely s, in which u has minimum weight. Otherwise, $Q_s(u,v) = 0$. Let A be the Boolean adjacency matrix of G (with 0's on the diagonal). Let B be the Boolean matrix with $B(u,v) = 1$ if and only if $w(u) \geq w(v)$. Clearly, $P_0 = Q_0 = I$. As in the proof in Section 2, we have $P_s = AP_{s-1} \wedge B$ and $Q_s = Q_{s-1}A \wedge B^T$. We compute P_s and Q_s, for $s = 0, \ldots, t$, in $O(tn^\omega)$ time. Now, for two nonnegative integers r, s with $r + s \leq t$, let $W_{r,s}$ be a maximum witness matrix for the Boolean product $P_r Q_s$. By Theorem 2.1, all the matrices $W_{r,s}$ can be computed in $O(t^2 n^{2+\mu})$ time. Having computed them, we claim that for each ordered pair u, v with $d(u,v) \leq t$, we can determine $sc(u,v)$ in $O(t^2)$ time by setting

$$sc(u,v) = w\Big(\max_{\substack{0 \leq r,s \leq t \\ r+s = d(u,v)}} W_{r,s}(u,v)\Big) . \tag{5.2}$$

Indeed, as in the proof of Theorem 1.1, suppose there is a path p from u to v of length $d(u,v)$ having capacity $sc(u,v)$. Let y be a vertex on p with $w(y) = sc(u,v)$. The path p is a concatenation of two paths $p_1 = (u, \ldots, y)$ and $p_2 = (y, \ldots, v)$ with p_1 having length $r \geq 0$ and p_2 having length $s \geq 0$, so that $r + s = d(u,v) \leq t$. Thus, $P_r(u,y) = 1$ and $Q_s(y,v) = 1$. Thus, $W_{r,s}(u,v) \geq y$. Hence, in (5.2) we get value bounded by w^*, and it is easy to see that we in fact get the correct value.

It remains to compute $sc(u,v)$ for pairs u, v with $d(u,v) > t$. Let S be the set of shortest paths with maximum capacity between all the pairs of vertices $(u,v) \in V \times V$ for which $d(u,v) \geq t$. Let B be a set of vertices as guaranteed by Lemma 5.2. Suppose that we have computed $sc(y,v)$ and $sc(v,y)$ for each $y \in B$, in $O(n^2|B|)$ time using Lemma 5.1. Now suppose that u, v have $d(u,v) > t$. In that case there is a path p from u to v of length $d(u,v)$ that has capacity $sc(u,v)$ and that contains a vertex $y \in B$. Notice that $sc(u,v) = \min\{sc(u,y), sc(y,v)\}$. Hence, setting

$$sc(u,v) = \max_{\substack{y \in B \\ d(u,y) + d(y,v) = d(u,v)}} \min\{sc(u,y), sc(y,v)\}$$

yields the correct value of $sc(u,v)$. The overall running time of the algorithm is therefore

$$\tilde{O}\left(t^2 n^{2+\mu} + n^2 \frac{n}{t}\right).$$

Setting $t = n^{(1-\mu)/3}$ we obtain an overall running time of $\tilde{O}(n^{(8+\mu)/3}) = O(n^{2.86})$. ■

The above algorithm solves the case were the edges of the graph are unweighted. The algorithm can be extended to the case of small edge weights as well, that is edge weights from the set $\{1, \ldots, M\}$. For $M = n^{o(1)}$ we get essentially the same running time. The full details will appear in the full version.

6 Concluding Remarks and Open Problems

- Let A and B be two $n \times n$ real-valued matrices. The *MIN-MAX product* $C = A * B$ is defined by $C(i,j) = \max_{k=1}^{n} \min\{A(i,k), B(k,j)\}$. Clearly computing min-max product is a special case of edge-APBP on a 3-layer graph constructed as in the proof of Theorem 1.2. The converse can be shown as well using a method from Aho et al. [1], Section 5.9, Corollary 2. The min-max product problem resembles the *min-plus product* problem where one defines $C(i,j) = \min_{k=1}^{n} A(i,k) + B(k,j)$. The fastest published algorithm for the latter, due to Chan [4], runs in $O(n^3/\log n)$ time (a mild improvement having running time $O(n^3(\log\log n/\log n)^{5/4})$ was recently announced by Y. Han).

- Our algorithm for the APBP problem runs in time $O(n^{2+\mu})$, which is equivalent to the fastest algorithm [21] for the all-pairs shortest paths problem on directed unweighed graphs. It will be interesting to see if it can be shown that one of the problems is not harder then the other, or if they can be shown to be equivalent.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. **The Design and Analysis of Computer Algorithms**. Addison-Wesley, Reading, 1974.

[2] N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[3] M.A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms*, 57(2):75–94, 2005. Also, Proc. of SODA 2001.

[4] T.M. Chan. All-Pairs Shortest Paths with Real Weights in $O(n^3/\log n)$ Time. In *Proc. of the 9th WADS*, Lecture Notes in Computer Science 3608, Springer (2005), 318–324.

[5] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13:42–49, 1997.

[6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbol. Comput.*, 9:251–280, 1990.

[7] M.J. Fischer and A.R. Meyer. Boolean matrix multiplication and transitive closure. In *Proc. of the 12th Symposium on Switching and Automata Theory*, East Lansing, Mich., (1971), 129–131.

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

[9] M.E. Furman. Application of a method of fast multiplication of matrices in the problem of finding the transitive closure of a graph. Dokl. Akad. Nauk SSSR, 11 (1970), no. 5, p. 1252.

[10] Z. Galil and O. Margalit. Witnesses for Boolean matrix multiplication and for transitive closure. *Journal of Complexity*, 9(2):201–221, 1993.

[11] H.N. Gabow and R.E. Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417, 1988.

[12] X. Huang and V.Y. Pan. Fast rectangular matrix multiplications and applications. *Journal of Complexity*, 14:257–299, 1998.

[13] M. Kowaluk and A. Lingas. LCA Queries in Directed Acyclic Graphs. In *Proc. of the 32nd ICALP*, Lecture Notes in Computer Science 3580, Springer (2005), 241–248.

[14] A. Lingas. Result announced at ICALP 2005 presentation of [13].

[15] I. Munro. Efficient determination of the strongly connected components and the transitive closure of a graph. Unpublished manuscript, Univ. of Toronto, Toronto, Canada, 1971.

[16] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389-1401, 1957.

[17] R. Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.

[18] V. Vassilevska and R. Williams. Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In *Proc. of the 38th STOC*, 225–231, 2006.

[19] V. Vassilevska, R. Williams, and R. Yuster. Finding the smallest H-subgraph in real weighted graphs and related problems. In *Proc. of the 33rd ICALP*, Lecture Notes in Computer Science, Springer (2006), 262-273.

[20] U. Zwick. All Pairs Lightest Shortest Paths. In *Proc. of the 31st STOC*, 61–69, 1999.

[21] U. Zwick. All-pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49:289–317, 2002.