On the Exponent of the All Pairs Shortest Path Problem

Noga Alon *
Department of Mathematics
Sackler Faculty of Exact Sciences
Tel–Aviv University

Zvi Galil †
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel–Aviv University and Columbia University

Oded Margalit
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel–Aviv University

Abstract

The upper bound on the exponent, ω, of matrix multiplication over a ring that was three in 1968 has decreased several times and since 1986 it has been 2.376. On the other hand, the exponent of the algorithms known for the all pairs shortest path problem has stayed at three all these years even for the very special case of directed graphs with uniform edge lengths. In this paper we give an algorithm of time $O\left(n^{\nu}\log^3 n\right)$, $\nu = (3+\omega)/2$, for the case of edge lengths in $\{-1,0,1\}$. Thus, for the current known bound on ω, we get a bound on the exponent, $\nu < 2.688$. In case of integer edge lengths with absolute value bounded above by M, the time bound is $O\left((Mn)^{\nu}\log^3 n\right)$ and the exponent is less than 3 for $M = O(n^{\alpha})$, for $\alpha < 0.116$ and the current bound on ω.

1 Introduction

Given a directed graph $G = \langle V, E\rangle$ and a length function $W : E \to I\!R$, the *All Pairs Shortest Path* problem, APSP in short, is to find, for each pair of vertices, $v_i, v_j \in V$, the length of the shortest path from v_i to v_j. (A length of a path is defined as the sum of the length of its edges.)

The shortest path is an old problem. The first algorithm is due to Dijkstra ([5]) and is 33 years old. Small improvements were made to this algorithm, mostly using sophisticated data structures (see [2]). All of them, but one, did not improve the $O(n^3)$ time bound for dense graphs. Fredman's algorithm ([6]) is $o(n^3)$, but its exponent is still 3.

This problem is equivalent to matrix multiplication over the closed semi–ring {min,+} (see [1]). We call it *"funny matrix multiplication"*. Aho, Hopcroft and Ullman suggest, in their book, that one may use similar techniques to those used by Strassen ([13]) to get a faster algorithm. Kerr ([8]) showed that one can not compute funny matrix multiplication with less than $\Omega(n^3)$ operations, when only minimum and sum are allowed. There is a simple transformation (see [14]) of the problem to a regular real matrix multiplication, thus yielding an $O(n^{\omega})$ time algorithm, where $\omega \leq 2.376$ (see [4]). The drawback of this method is the usage of infinite precision real numbers

*Research supported in part by a United states Israel BSF Grant.
†Work partially supported by NSF Grants CCR-8814977 and CCR-9014605.

arithmetic. Thus this method is not efficient at all when taking the more realistic non–uniform costs on the arithmetic operations.

In this paper we consider a simple version of the APSP problem in which the graph is directed and all edges have ± 1 or 0 length. We then extend our algorithm to the case of edge lengths which are integers with a small absolute value. Even for the case of uniform edge length (+1 length only), the best exponent has been 3 so far.

In the rest of this section we give some definitions and basic properties. In Section 2 we show how to solve some simple cases of the problem. In Section 3 we show how to solve the $\{-1, 0, 1\}$ edge lengths case in $O(n^\nu)$ time with $\nu < 3$ by using a similar method to the scaling method suggested by Gabow ([7]). The algorithm also finds negative cycles. A simple observation extends this algorithm to the case where edge lengths are small bounded integers. The fastest algorithms for matrix multiplication have only theoretical value since they are better than the naive way only when the matrices are impractically huge. Our algorithm uses the fast matrix multiplication as a black box. So if we use a practical algorithm (for matrix multiplication) we still get an $O(n^\mu)$ time practical algorithm, where $\mu < 3$. In Section 4 we give a simple solution for the uniform case (all edges have unit length). It is faster than the solution obtained from the algorithm of Section 3 by logarithmic factors. In Section 5 we briefly mention some recent results, and in Section 6 we present a few open problems.

Definitions

Given the input graph G with n vertices and an integer edge length function W, define the *length matrix* $D = \{d_{ij}\}_{i,j=1}^n$ as

$$d_{ij} = \begin{cases} w(e), & \exists\ e \in E,\ e = (v_i, v_j) \\ \infty, & \text{Otherwise.} \end{cases}$$

Let J be the funny identity matrix defined as

$$j_{ik} = \begin{cases} 0, & i = k \\ \infty, & \text{otherwise.} \end{cases}$$

For an integer $\ell > 0$, define $D^{\leq \ell} \stackrel{\text{def}}{=} \min_{1 \leq i \leq \ell} D^i$, where the matrix multiplications are funny matrix multiplication ones. Let $D^{\leq \ell} = \{d_{ij}^{\leq \ell}\}_{i,j=1}^n$. Denote by $D^* = \{d_{ij}^*\}_{i,j=1}^n$, the matrix of the shortest distances between any two vertices, i.e. the minimum over all nonempty paths of the sum of the length of the edges. Note that if empty paths are allowed we only have to change the entries d_{ii}^* which are not $-\infty$ to zero. D^* is the solution of the APSP problem.

An entry d_{ij}^* is called *δ–regular* iff there is a path from v_i to v_j in D which has minimal length (d_{ij}^*) and no more than δ edges. A matrix D is called δ–regular iff all the entries that satisfy $d_{ij}^* > -\infty$ are δ–regular and for the cases where $d_{ij}^* = -\infty$, there exists a path in D with no more than δ edges which has a nonpositive length.

Basic properties

- It is easy to see that $D^{\leq \ell} = (J+D)^{\ell-1} D$, using funny matrix multiplications in this expression.
- $d_{ij}^{\leq \ell}$ is the shortest distance from v_i to v_j which is at least 1 and at most ℓ edges long. Thus if d_{ij}^* is δ–regular then $d_{ij}^* = d_{ij}^{\leq \delta}$.
- If $\ell_1 \geq \ell_2$ then $D^{\leq \ell_1} \leq D^{\leq \ell_2}$, $D = D^{\leq 1}$ so $D^{\leq \ell} \leq D$ for any $\ell > 0$.

- $D^* \leq D^{\leq \ell}$ for any ℓ.
- If the lengths are nonnegative, or even when G does not contain negative cycles, then $D^* = D^{\leq n}$. Note that the nonnegativity is needed here since if the edge length can be negative, d^*_{ij} can be $-\infty$ but if $-\infty \notin D$ then $d^{\leq \ell}_{ij} > -\infty$ for any finite ℓ.
- $A \leq B$ implies that $A^* \leq B^*$ and that $A^{\leq \ell} \leq B^{\leq \ell}$, for any integer $\ell > 0$.
- $(A^*)^* = A^*$.

2 Solving a simple case

In this section we solve simple versions of the APSP problem. We present a way to encode the length matrix so that regular matrix multiplication over the integers will give us the answer.

Lemma 1 *Given two integer valued matrices*

$$A = \{a_{ij}\}_{i,j=1}^{n}$$

and

$$B = \{b_{ij}\}_{i,j=1}^{n}$$

such that $x < a_{ij}, b_{ij} < y$ and $y - x < M$, we can compute the matrix C which is the funny matrix multiplication of A and B in $O(Mn^{\omega} \log M)$ time.

Proof. Define the matrices A' and B' as follows.

$$a'_{ij} = (n+1)^{a_{ij}-x}.$$

$$b'_{ij} = (n+1)^{b_{ij}-x}.$$

Examine the integer matrix multiplication $C' = A'B'$:

$$c'_{ij} = \sum_{k=1}^{n} (n+1)^{a_{ik}+b_{kj}-2x}.$$

It is easy to see that c'_{ij} is divisible by $(n+1)^s$ iff for all k, $a'_{ik} + b'_{kj} - 2x \geq s$. Using binary search we find s_{ij}, the largest such s and $c_{ij} = s_{ij} + 2x$ in $O(\log M)$ time for each entry. So the total time complexity of the algorithm above is $O(n^{\omega})$ operations on integers which are $\leq n(n+1)^{2M}$. Each operation on such large numbers takes $O(M \log M)$ operations on $O(\log n)$ bits numbers, which yields the complexity stated above. □

We actually implemented the algorithm of [14] in integers. The exponent of this algorithm, even for the $\{1, \infty\}$ case exceeds 3 (M can be as large as n, resulting in an exponent $\omega + 1 > 3$). We slightly modify our way of computing the funny matrix multiplication, so that it will truncate distances which are greater than a given bound M after each multiplication, as well as truncating the input, so we get:

Theorem 1 *Given a graph G whose edge lengths matrix D, satisfies $d_{ij} \in \{-1, 0, 1, \infty\}$, we can compute $D^{\leq \ell}$ in $O(\ell n^{\omega} \log^2 \ell)$ time.*

Proof. Recall that $D^{\leq \ell} = (J + D)^{\ell-1} D$. Apply the above algorithm with $\ell + 1$–truncation (replace entries larger than ℓ (and in particular, the infinite entries in J) by $\ell + 1$) and ignore all the entries which are greater than ℓ in the output. All the matrices involved in this computation have entries with absolute value $\leq \ell + 1$, hence each funny matrix multiplication takes $O(\ell n^\omega \log \ell)$ time and there are $O(\log \ell)$ funny matrix multiplications. □

Theorem 2 *We can solve the APSP problem for graphs with edge length of* $\{1, \infty\}$ *and small* $(\leq M)$ *diameter in* $O(Mn^\omega \log n \log M)$ *time.*

Proof. If $d^*_{ij} = k$ then there is a k–edges long path from v_i to v_j, so $D^* = D^{\leq M}$. We can compute $D^{\leq M}$ as described above. We can even detect this case (small diameter) without knowing it in advance by noticing that $D^{\leq M+1}$ does not contain any $M + 1$ entry. □

There is another way to solve the positive case in which we can save the logarithmic factors.

Theorem 3 *Given a graph* G *with an edge length matrix* D*, such that* $d_{ij} \in \{1, \infty\}$*, we can find, all the entries in* D^* *which are* $\leq M$*, in* $O(Mn^\omega)$ *time.*

Proof. Define $A^{(\ell)}$ as the matrix

$$a^{(\ell)}_{ij} = \begin{cases} 1, & d^*_{ij} \leq \ell \\ 0, & \text{otherwise.} \end{cases}$$

$A^{(1)}$ can be easily derived from D and $A^{(k+1)} = A^{(k)} \cdot A^{(1)}$ where the multiplication is a regular Boolean matrix multiplication (and/or). This matrix multiplication can be computed in $O(n^\omega)$ time by computing the multiplication over the integers and then changing each non–zero entry to 1. Therefore $\{D^{(\leq i)}\}_{i=1}^{M}$ can be computed in $O(Mn^\omega)$ time. The entries of D^* not larger than M can be computed in the same time by observing for each i, j the first ℓ such that $a^{(\ell)}_{ij} = 1$. □

3 Finding the shortest distances

In this section we describe an algorithm for solving a version of the APSP problem where the edges lengths are integers with a small absolute value. Negative cycles are allowed and the algorithm finds all the $-\infty$ distances. We start with the case of edge lengths in $\{-1, 0, 1, \infty\}$.

The main difficulty with negative edge lengths is that a very small distance (small in absolute value), even zero, can have a shortest path which realizes it with many edges. For example, a path of $n/2$ edges of weight $+1$, followed by $n/2$ edges of weight -1. Our solution to that problem is to find shortcuts: we find all the zero distances and update the edge lengths accordingly. After this update, the graph satisfies an important property: every distance which has a small absolute value, is achieved on a path with a small number of edges.

We solve recursively the following problem. We are given a length matrix D such that $d_{ij} \in \{-1, 0, 1, \infty\}$ and an integer δ such that D is δ–regular. The algorithm computes D^*. The initial recursive call is with the original length matrix D and $\delta = n^2$.

Lemma 2 *Any* $\{-1, 0, 1, \infty\}$ *matrix* D *is* n^2*-regular.*

Proof. If $d^*_{ij} > -\infty$ then there is a shortest path with no cycles and therefore with no more than n edges. If $d^*_{ij} = -\infty$ then there is a path from v_i to v_j which includes a single cycle of negative

length. Repeating this cycle $n-2$ times will ensure that the total length will be nonpositive while using no more than n^2 edges. □

In order to find D^*, the algorithm first removes, in some sense which is defined later (Lemma 5), the zero length edges from D, creating the matrix E; then it computes the matrix $E^{\leq 2}$ and recursively solves the problem for the matrix $E' = \lceil E^{\leq 2}/2 \rceil$ and $\delta' = \lceil \delta/2 \rceil$. The fact that E does not contain zero length edges makes $(2E')^*$, where $(E')^*$ is the recursive solution, approximate quite well the desired solution and this approximation is enough so that $(E')^*$ will "catch" all the zeros in D^*. Having all the zero distances it is easy to fix E (by adding these shortcuts) to have the important property mentioned above and to find D^*.

Algorithm

1. This step takes care of the last recursive step when $\delta = 1$. If $\delta > 1$ proceed. If $\delta = 1$ then $D = D^{\leq 1}$ solves all the entries which are $> -\infty$. To find the $-\infty$ entries we first identify the negative cycles and then use them to find all the $-\infty$ distances.

   Lemma 3 *If the matrix D is 1–regular then the set of all v_i–s such that $d^*_{ii} < 0$ can be found in $O(n^\omega)$ time.*

   Proof. We now show that $d^*_{ii} < 0$ iff $d^3_{ii} < 0$ (as an entry of D^3 using funny matrix-multiplication). If $d^*_{ii} < 0$ then there is a negative cycle which goes through v_i. There must be a negative edge on the negative cycle, denote it by (v_j, v_k). $d^*_{ij} = d^*_{ki} = -\infty$, so from the δ–regularity ($\delta = 1$), $d_{ij}, d_{ki} \leq 0$. Since $d_{jk} < 0$, $d^3_{ii} < 0$. On the other hand, $D^* \leq D^3$ therefore $d^3_{ii} < 0$ implies that $d^*_{ii} < 0$. The time complexity is $O(n^\omega)$ since D^3 can be computed using several Boolean matrix multiplications. □

   All the $-\infty$ distances can now be easily computed.

   Lemma 4 *D^* can be computed in $O(n^\omega)$ time.*

   Proof. Build a directed graph whose vertices are the n vertices of the original problem and an edge is connecting v_i to v_j iff $d_{ij} < \infty$ and either d^*_{ii} or d^*_{jj} are $-\infty$. We now show that $d^*_{ij} = -\infty$ iff there is a path of length 2 from v_i to v_j in the new graph. Any edge in the new graph corresponds to a path of $-\infty$ length. Therefore, so does any path in the new graph. On the other hand, if $d^*_{ij} = -\infty$ then there exists a negative cycle on a path from v_i to v_j. Let v_k be any vertex on this negative cycle. $d^*_{ik} = d^*_{kj} = -\infty$ and D is 1–regular, therefore $d_{ik}, d_{kj} \leq 0 < \infty$. $d^*_{kk} < 0$, so both edges (v_i, v_k) and (v_k, v_j) are in the new graph. Hence there is a path of length 2 from v_i to v_j. Paths of length 2 can be computed using Boolean matrix multiplication in $O(n^\omega)$ time. □

2. Define D^-, D^0 and D^+ as follows:

$$d^-_{ij} = \begin{cases} 1, & d_{ij} = -1 \\ 0, & \text{Otherwise.} \end{cases}$$

$$d^0_{ij} = \begin{cases} 1, & d_{ij} = 0 \\ 0, & \text{Otherwise.} \end{cases}$$

$$d^+_{ij} = \begin{cases} 1, & d_{ij} = +1 \\ 0, & \text{Otherwise.} \end{cases}$$

Find the transitive closure of D^0 and denote it by E^0. Compute the Boolean matrix multiplications

$$E^- = (I + E^0)D^-(I + E^0),$$

and

$$E^+ = (I + E^0)D^+(I + E^0),$$

(The identity matrix I is needed iff $\exists\ i,\ d_{ii} > 0$.) Define E as follows:

$$e_{ij} = \begin{cases} -1, & e^-_{ij} = 1 \\ 0, & e^-_{ij} = 0 \text{ and } e^0_{ij} = 1 \\ 1, & e^-_{ij} = 0 \text{ and } e^0_{ij} = 0 \text{ and } e^+_{ij} = 1 \\ \infty, & \text{Otherwise.} \end{cases}$$

Note that e_{ij} is the length of a shortest path among paths with at most one non–zero length edge.

Lemma 5 *The matrix E satisfies:*

$$D^* = E^*, \tag{1}$$

E is δ–regular, furthermore, E satisfies the zero length edge property, namely for any path from v_i to v_j in D or E, there is a corresponding path in E which satisfies:

(a) its length is not greater than the length of the original path;

(b) it has no more edges than the original path; and

(c) if the path contains more than one edge, then it does not contain any zero length edge.

E can be computed in $O(n^\omega)$ time.

Proof. By definition, $E^- \geq D^-$, $E^0 \geq D^0$ and $E^+ \geq D^+$. So $E \leq D$ and hence $E^* \leq D^*$. On the other hand, since entries in E are lengths of paths in D, we have $E \geq D^*$ and $E^* \geq (D^*)^* = D^*$. The δ–regularity of E follows from the fact $E \leq D$ and the δ–regularity of D.

We now prove the zero–length edges property. If the initial path has length of $+\infty$, then the path which consists of the single edge e_{ij} satisfies the three conditions above. Otherwise, examine the path in E which has the minimum number of edges among all the paths which are no longer than the original path. $E \leq D$ so clearly this path has no more edges and it is no longer. We only have to show that if it consists of more than one edge, then it does not contain any zero length edge. Suppose to the contrary, that it contains a zero length edge $e_{ij} = 0$, so $e^0_{ij} = 1$. If e_{ij} is not the last edge on the path, examine the next edge in the path: e_{jk}. If $e_{jk} = 0$ then $e^0_{jk} = 1$ and from the transitivity of E^0, $e^0_{ik} = 1$, and $e_{ik} \leq 0$, so there is another path of not larger length with one less edge — a contradiction. If $e_{jk} = \pm 1$ then $e^\pm_{jk} = 1$ and therefore there exist j' and k' such that $e^0_{jj'} = 1,\ d_{j'k'} = \pm 1$ and $e^0_{k'k} = 1$. From the transitivity of E^0, $e^0_{ij'} = 1$, so $e^\pm_{ik} = 1$ and $e_{ik} \leq \pm 1 = e_{ij} + e_{jk}$ yielding the same contradiction. If e_{ij} is the last edge on the path, we derive the contradiction by considering the previous edge.

The transitive closure can be computed in $O(n^\omega)$ time as described in [1] and all the other computation can be computed in the same time as well. □

3. Given the matrix E, find the shortest path which are at most two edges long, $E^{\leq 2}$. The distances we get are $e_{ij}^{\leq 2} \in \{-2,-1,0,1,2,\infty\}$. This can be done in $O(n^\omega)$ time as described in Section 2, or even by several Boolean matrix multiplications.

4. Define the matrix E' as $E' \stackrel{\text{def}}{=} \lceil E^{\leq 2}/2 \rceil$.

   Lemma 6 *The matrix E' satisfies*

   $$(E')^* = \lceil E^*/2 \rceil \tag{2}$$

   and E' is $\lceil \delta/2 \rceil$–regular.

   Proof.

   (a) Consider a path in E of length λ and μ edges. By Lemma 5 there is a corresponding path in E of length at most λ and at most μ edges satisfying property (c).The corresponding path in E' has length at most $\lceil \lambda/2 \rceil$ and at most $\lceil \mu/2 \rceil$ edges. Thus $E'^* \leq \lceil E^*/2 \rceil$.

   (b) To each path in E' of length λ corresponds a path in E of length $\leq 2\lambda$. Thus $E^* \leq 2(E')^*$ or $(E')^* \geq E^*/2$.

   Now (2) follows from (a) and (b).

   We now prove the $\lceil \delta/2 \rceil$–regularity of E'. If $(e')_{ij}^* = -\infty$ then by (2), $e_{ij}^* = -\infty$; by the δ–regularity of E, $e_{ij}^{\leq \delta} \leq 0$ and from (a), $(e')_{ij}^{\leq \lceil \delta/2 \rceil} \leq 0$. Otherwise ($e_{ij}^* > -\infty$), examine the shortest path from v_i to v_j in E which has no more than δ edges (the one which exists from the δ–regularity of E). The path in (a) has no more than $\lceil \delta/2 \rceil$ edges and its length is $\leq \left\lceil e_{ij}^*/2 \right\rceil$ (by (2)). This path must have length $(e')_{ij}^*$. □

5. Solve, recursively, the problem for E' and $\delta' = \lceil \delta/2 \rceil$. Note that the δ parameter is halved at each iteration. Therefore the recursion will not be more than $\lceil \log_2 n^2 \rceil$ deep. Note that as we showed in the last lemma, the matrix E' and $\lceil \delta/2 \rceil$ satisfy the requirements of the algorithm.

   Thus by (2), $(2E')^*$ gives correctly the even entries of D^* ($= E^*$) and is smaller by 1 for the odd entries of D^*. We use E' to add some shortcuts.

6. Compute F as:

   $$f_{ij} = \begin{cases} e_{ij}, & (e')_{ij}^* > 0 \\ min\{e_{ij}, 0\}, & (e')_{ij}^* \leq 0. \end{cases}$$

   Lemma 7 $E^* = F^*$.

   Proof. First note that $F \leq E$ so $F^* \leq E^*$. For the other inequality, we will show $F \geq E^*$ and hence $F^* \geq (E^*)^* = E^*$. When $f_{ij} = e_{ij}$ we have $f_{ij} \geq e_{ij}^*$. Otherwise $f_{ij} = 0$ and $(e')_{ij}^* \leq 0$. But from (2), $e_{ij}^* \leq (2e')_{ij}^* \leq 0 = f_{ij}$. □

   We proceed by computing D^* from F.

7. Compute, using the algorithm of Section 2 (Theorem 1), $F^{\leq 2\ell+4}$, where $\ell = 1 + \left\lceil n^\Delta \right\rceil$, $\Delta = (3-\omega)/2$. It will provide some of the entries of the output matrix D^*.

   Note that

   $$F^{\leq 2\ell+4} \geq F^* = E^* = D^*. \tag{3}$$

Lemma 8 *If there exists a path from v_i to v_j in D whose length is $\leq \lambda$ where $|\lambda| \leq \ell + 1$ then $f_{ij}^{\leq 2\ell+4} \leq \lambda$.*

Proof. Let $v_i = u_1, u_2, \ldots, u_m = v_j$ be a path in D from v_i to v_j of length $\mu \leq \lambda$, where $|\lambda| \leq \ell + 1$. First assume that $|\mu| \leq \ell + 1$ and $\mu \geq 0$ ($\mu \leq 0$). For $0 \leq k \leq \mu$ ($-\mu \leq k \leq 0$), let j_k be the index of the last vertex on the path of distance k from v_i along the path. Consider the path $u_1, u_{j_0}, u_{j_0+1}, u_{j_1}, \ldots, u_{j_\mu}$ ($u_1, u_{j_0}, u_{j_0+1}, u_{j_{-1}}, \ldots, u_{j_{-\mu}}$) as a path in F of at most $2\mu + 1 = 2\ell + 3$ edges. If two consecutive vertices coincide, delete one of them and skip that edge. For the edges (v_p, v_q) in the even (in the original order, before deleting vertices) numbered positions ($p = i_{j_k}, q = i_{(j_k+1)}$), $f_{pq} \leq e_{pq} \leq d_{pq}$. Each edge (v_p, v_q) in an odd numbered positions ($p = i_{(j_k+1)}, q = i_{j_{k+1}}$) corresponds to a sub–path of length 0 if it is non–empty (otherwise it would have been deleted) and thus $d^*_{pq} \leq 0$. From (1), $e^*_{pq} = d^*_{pq} \leq 0$. Using (2), we get $(e')^*_{pq} \leq 0$, so $f_{pq} \leq 0$. It follows that the F–path is no longer than the D–path and $f_{ij}^{\leq 2\ell+4} \leq d^*_{ij} \leq \lambda$.

Figure 1 shows a typical path of 22 edges long in E and the path with only 5 edges in F.

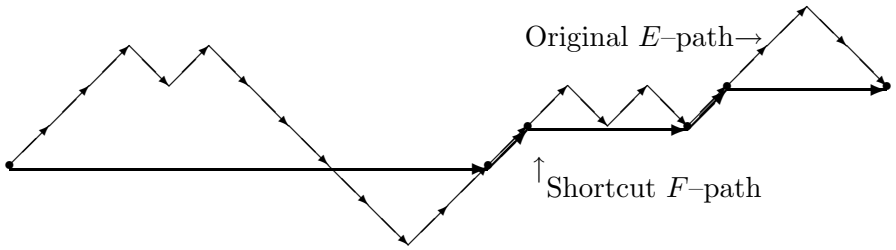


Figure 1: Shortcuts

Now suppose that $\mu < -\ell - 1$ and consider the path $u_1, u_{j_0}, u_{j_0+1}, u_{j_{-1}}, \ldots, u_{j_{-\ell-1}}, u_\mu$. The same arguments as above show that the length of this path in F without its last edge is $\leq -\ell - 1$. As for the last edge — it shortcuts a sub–path of negative length in E, therefore (by (2)), $(e')_{pq} \leq 0$ and $f_{pq} \leq 0$. So the total length of the path is $\leq -\ell - 1 \leq \lambda$. □

Lemma 9 *If $|f_{ij}^{\leq 2\ell+4}| \leq \ell$ or $|d^*_{ij}| \leq \ell$ then $d^*_{ij} = f_{ij}^{\leq 2\ell+4}$.*

Proof. First assume that $|f_{ij}^{\leq 2\ell+4}| \leq \ell$ and show that $|d^*_{ij}| \leq \ell$. Suppose that $|d^*_{ij}| > \ell$. If $d^*_{ij} > \ell$ then from (3), $|f_{ij}^{\leq 2\ell+4}| \geq f_{ij}^{\leq 2\ell+4} \geq d^*_{ij} > \ell$. If $d^*_{ij} < -\ell$ then there exists a path of length $\leq -\ell - 1$ in D. From Lemma 8, $f^{\leq 2\ell+4} \leq -\ell - 1 < -\ell$. So $|d^*_{ij}| \leq \ell$. Assume $|f_{ij}^{\leq 2\ell+4}| \leq \ell$ or $|d^*_{ij}| \leq \ell$. By the argument above $|d^*_{ij}| \leq \ell$ in both cases. Consider a path from v_i to v_j in D of length d^*_{ij}. By Lemma 8 with $\lambda = -\ell - 1$ we get $f_{ij}^{\leq 2\ell+4} \leq d^*_{ij}$. The other inequality follows from (3). □

8. As for the large positive (negative) distances: For each vertex v_i, find two distances; an odd one s_i^1 and an even one s_i^2. Both of them are in the interval $[1, n^\Delta + 1]$ ($[-1 - n^\Delta, -1]$) and both of them are small separators: there are less than $2n^{1-\Delta}$ vertices which satisfy $f_{ik}^{\leq 2\ell+4} = s_i^1$ or

$f_{\bar{ik}}^{\leq 2\ell+4} = s_i^2$. Denote this set of vertices by $S_i = S_i^1 \cup S_i^2$. This can be done in $O(n^2)$ time. Now compute the large distances using the following lemma.

Lemma 10 *Let v_i and v_j be vertices with $f_{\bar{ij}}^{\leq 2\ell+4} > \ell$ ($f_{\bar{ij}}^{\leq 2\ell+4} < -\ell$). Denote*

$$m_{ij} \stackrel{\text{def}}{=} \min_{v_k \in S_i} \{f_{\bar{ik}}^{\leq 2\ell+4} + (2e')_{kj}^*\}.$$

Then

$$e_{ij}^* = \begin{cases} (e')_{ij}^*, & (e')_{ij}^* = \pm\infty \\ m_{ij}, & \textit{otherwise.} \end{cases}$$

Proof. By (2), $e_{ij}^* = \pm\infty$ if and only if $(e')^* = \pm\infty$. So assume that e_{ij}^* is finite. Since $|f_{\bar{ij}}^{\leq 2\ell+4}| > \ell$, by Lemma 9, $|d_{ij}^*| > \ell$ and each path from v_i to v_j of finite length passes though S_i.

($\leq$)

$$e_{ij}^* = \min_{v_k}\{e_{ik}^* + e_{kj}^*\} \leq \min_{v_k \in S_i}\{e_{ik}^* + e_{kj}^*\}.$$

By (3), $e_{ik}^* \leq f_{\bar{ik}}^{\leq 2\ell+4}$, and by (2), $e_{kj}^* \leq (2e')_{kj}^*$. So

$$e_{ij}^* \leq \min_{v_k \in S_i}\{f_{\bar{ik}}^{\leq 2\ell+4} + (2e')_{kj}^*\}.$$

($\geq$) Suppose that e_{ij}^* is an odd number. Examine a shortest path in E from v_i to v_j. It has a length of e_{ij}^*. From the fact that the length matrix E contains only numbers in $\{-1, 0, 1\}$ it follows that there is a vertex $v_k \in S_i^1$ whose distance from v_i along that path is $s_i^1 = e_{ik}^* = f_{\bar{ik}}^{\leq 2\ell+4}$ (by Lemma 9) and the distance from v_k to v_j along the same path is $e_{ij}^* - s_i^1$. $e_{kj}^* = e_{ij}^* - s_i^1$ is even, therefore (by (2)) $(2e')_{kj}^* \leq e_{kj}^*$. Hence $e_{ij}^* \leq f_{\bar{ik}}^{\leq 2\ell+4} + (2e')_{kj}^*$.
If e_{ij}^* is an even number, replace S_i^1 by S_i^2 in the argument above.

□

9. Compute D^* as follows:

$$d_{ij}^* = \begin{cases} -\infty, & (e')_{ij}^* = -\infty \\ f_{\bar{ij}}^{\leq 2\ell+4}, & |f_{\bar{ij}}^{\leq 2\ell+4}| \leq \ell \\ m_{ij}, & \text{otherwise.} \end{cases}$$

(Remember that m_{ij} is defined above as $\min_{v_k \in S_i}\{f_{\bar{ik}}^{\leq 2\ell+4} + (2e')_{kj}^*\}$.)

Lemma 11 *The computation above is correct and takes $O(n^3/\ell)$ time.*

Proof. The first and third cases follow from Lemma 9; the second case from Lemma 10. The time analysis is immediate. □

Theorem 4 *The algorithm above is correct (it computes D^*) and runs in $O(n^\nu \log^3 n)$ time.*

Proof. Denote by $T(n, \delta)$ the time complexity of the algorithm on an input of n nodes and a parameter δ.

- Computing E takes $O(n^\omega)$ time.
- Computing $E^{\leq 2}$ takes $O(n^\omega)$ time too.
- E' is easily computed in $O(n^2)$ time.
- The recursive step takes $T(n, \lceil \delta/2 \rceil)$ time.
- F is easily computed in $O(n^2)$ time.
- Computing $F^{\leq 2\ell+4}$ takes $O(n^\nu \log^2 n)$ time.
- Computing D^* takes $O(n^3/\ell) = O(n^\nu)$ time.

So we get the recursive formula:

$$T(n, \delta) \leq O(n^\nu \log^2 n) + T\left(n, \lceil \delta/2 \rceil\right).$$

This solves to $T(n, \delta) = O(n^\nu \log^2 n \log \delta)$ and $T(n, n^2) = O(n^\nu \log^3 n)$. □

Small integer length

To solve problems with small (in absolute value) integer edge lengths, transform the given graph into another graph which has more vertices, but only zero or ± 1 length edges.

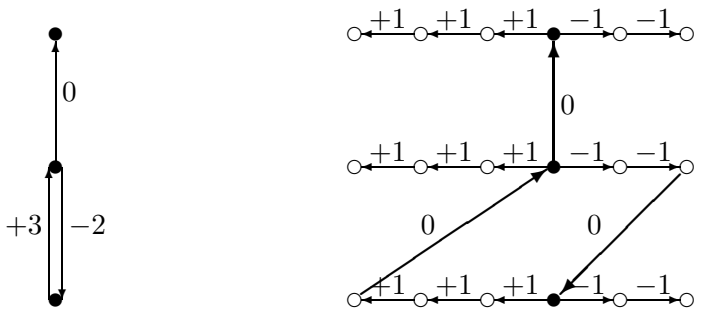
Lemma 12 *Given a graph G which has edge lengths in the interval $[x, y]$, where $x \leq 0 \leq y$ and $y - x < M$, we can, in time which is linear in the output, compute another graph G' with $M \cdot V(G)$ vertices and edge lengths in $\{-1, 0, 1\}$ only. Every shortest distance between a pair of vertices in G can be found as a shortest distance between a corresponding pair of vertices in G'.*

Proof. Transform each vertex v_i in G into M sub-vertices, $\{v_i^k\}_{k=x}^{y}$. Call the sub-vertex v_i^0 the *origin vertex* of the vertex v_i. Connect the sub-vertices of v_i in two paths: one with edges of length $+1$ connects the sub-vertices v_i^k with positive k-s and the other path with -1 length edges, connects the sub-vertices v_i^k with negative k-s. Formally, the edges are: $\{(v_i^{k-1}, v_i^k)\}_{k=1}^{y}$ with length 1, $\{(v_i^{k+1}, v_i^k)\}_{k=x}^{-1}$ with length -1. For every edge (v_i, v_j) of the original graph G which has a length ℓ, connect an edge (v_i^ℓ, v_j^0) with zero edge length. It is easy to see that the shortest distance in G between v_i and v_j, is the same as the shortest distance in G' between the corresponding origin sub–vertices v_i^0 and v_j^0. This transformation enlarges the size of the graph by a factor of M and the running time accordingly. (Figure 2 shows an example where $x = -2$ and $y = 3$.) □

4 A simple solution to the positive case

In this section we use the simple case we solved in Section 2 to get an $O(n^\nu)$ time algorithm for the APSP problem for a directed graph G which has only $\{1, \infty\}$ edge length, where $\nu = (3+\omega)/2 < 3$.

Denote by $D^{(\ell)}$ a matrix which satisfies $D^* \leq D^{(\ell)}$ and if $d_{ij}^* \leq \ell$ then $d_{ij}^* = d_{ij}^{(\ell)}$.



The original graph Transformed graph

Figure 2: The transformation

The Algorithm

1. Find, using $\left\lceil n^{\Delta} \right\rceil$ Boolean matrix multiplications (as in Theorem 3), d^* for all pairs with distance of at most $\left\lceil n^{\Delta} \right\rceil$ in the graph, where $\Delta = (3-\omega)/2$. The time complexity of this step is $O(n^{\nu})$.

2. In Step 1 we computed a $D^{(\lceil n^{\Delta} \rceil)}$ matrix. For $i = 1, \ldots, \left\lceil \log_{3/2} n^{1-\Delta} \right\rceil$, iterate the following step, each time computing a $D^{\left(\lceil (3/2)^i n^{\Delta} \rceil\right)}$ matrix. This is done by computing a $D^{(\lceil (3/2)k \rceil)}$ matrix from a $D^{(k)}$ matrix.

   (a) Find, for each vertex v_i, a distance s_i in the interval $[\lceil k/2 \rceil, k]$ such that there are no more than $2n/k$ vertices which are at this shortest distance from v_i. Denote this set by S_i. This can be done in $O(n^2)$ time by inspecting the matrix.

   (b) For every two vertices v_i and v_j compute $m_{ijk} \stackrel{\text{def}}{=} \min_{v_\ell \in S_i} \{d_{i\ell}^{(k)} + d_{\ell j}^{(k)}\}$ and

$$d_{ij}^{(\lceil \frac{3k}{2} \rceil)} = \begin{cases} d_{ij}^{(k)}, & d_{ij}^{(k)} \leq k \\ m_{ijk}, & d_{ij}^{(k)} > k \text{ and } m_{ijk} \leq \left\lceil \frac{3k}{2} \right\rceil \\ \infty, & \text{otherwise.} \end{cases}$$

   This can be done in $O\left(n^3/k\right)$ time.

Lemma 13 *The algorithm described above is correct and it runs in time $O(n^{\nu})$, where $\nu = (3+\omega)/2$.*

Proof. Correctness is obvious. Step 1 takes $O(n^{\nu})$ time by Theorem 3. Step 2 takes

$$O\left(\sum_i \left(n^3 / \left\lceil (3/2)^i n^{\Delta} \right\rceil\right)\right) = O(n^{\nu})$$

time. □

5 Recent results

Recently, the problems of All Pairs Shortest Distances/Paths were investigated by several people. In this section we sketch the recent results in the area.

1. We improved the algorithm for the undirected case [11]. The algorithm uses base 3 patterns to compute the distances and works in $\widetilde{O}(n^\omega)$ time. ($\widetilde{O}(f(n)) \stackrel{\text{def}}{=} O(f(n)\log^c n)$, where c is a constant; in this case $c = 1$.) It generalizes to $\widetilde{O}(M^2 n^\omega)$ for the weighted case (where M is the bound on the weight). Seidel [12] discovered a simpler algorithm with the same time complexity for the uniform case ($M = 1$). His algorithm does not seem to generalize to the non–uniform case even for $M = 2$. (The transformation of Figure 2 *does not* work in case of undirected graphs.)

2. We improved the solution for the case of small distances [11]: We used here a naive method (the reduction shown in Figure 2), but modifying the algorithm to take care of non–uniform edges, yields a better algorithm. The algorithms for the directed non–negative and for the undirected case, used the fact that for every path P from v_i to v_j of length ℓ and for every $0 < \ell' < \ell$, there must be a vertex v_k on the path such that $d^P_{ik} = \ell'$. Now, when large edges are present, this is not true anymore; we can "skip" some distances by a large edge. So we modify the algorithms and, for example, instead of the computation $A^{(k+1)} = A^{(k)} \cdot A^{(1)}$, we now have $A^{(k+1)} = \bigvee_{j=1}^{M} A^{(k+1-j)} A^{(j)}$.

   In the directed non–negative case, we compute L rectangular Boolean matrix multiplications, of sizes $n \times Mn$ by $Mn \times n$. These Boolean matrix multiplications can be packed into L/M^2 Boolean matrix multiplications of square matrices of size Mn. It gives us $L/M^2(Mn)^w + Mn^3/L$, which is $M^{(\omega-1)/2} n^{(3+\omega)/2}$, but this holds only when $L > M^2$; otherwise, we can pack into $M/\sqrt{L}$ multiplications of square Boolean matrices of size $\sqrt{L}n$. This yields the time $O(Mn^{(5\omega-3)/(\omega+1)})$.

   In the undirected case, we compute M rectangular Boolean matrix multiplications of sizes $n \times Mn$ by $Mn \times n$. These matrix multiplications can be packed into $\sqrt{M}$ Boolean matrix multiplications of square matrices of size $\sqrt{M}n$. This improves the dependency of the time on M mentioned above from M^2 into $M^{(\omega+1)/2}$.

3. All the algorithms we described in this paper give only the shortest distances. We can find the shortest paths as well in almost the same time [11]. Of course we cannot list all shortest paths in less than $O(n^3)$ space and time. So instead we compute for each pair (v_i, v_j), a *witness:* the first edge on a shortest path from v_i to v_j, such that following witnesses we can construct a simple shortest path for any given pair. The way to do it is to replace each Boolean matrix multiplication in the algorithm by a *witnessed* Boolean matrix multiplication: a multiplication which does not only compute $c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$, but also finds a *witness* k which "proves" that $c_{ij} = 1$. There is a simple randomized algorithm that computes witnessed Boolean matrix multiplication (discovered independently by us, by R. Seidel [12] and by D. Krager [10]), in polylog calls to a non–witnessed matrix multiplication. We can do it deterministically in $O(n^{\omega+\log^{-1/3}(n)})$ time, and recently, using derandomization [3] in $\widetilde{O}(n^\omega)$ deterministic time.

   So, the time complexity of finding the paths exceeds that of computing the distances only by a polylogarithmic factor. This result is not trivial; several steps in our algorithms have to be changed especially to avoid cycles in the paths.

6 Open problems

1. Can we improve the algorithm to $\widetilde{O}(n^\omega)$? Our algorithm computes an approximation $((2E')^*)$ for D^*, but then uses it only for the purpose of defining the shortcuts in F. Can we find a better use? Seidel's algorithm in the undirected case [12] uses a nice way to compute D^* from $(2E')^*$ directly using a single integer matrix multiplication, but the directed case seems more difficult.

2. Can we further improve the dependency on M? By solving the problem with arbitrary large integers we can solve it also with rational numbers and thus approximate the solution in case of real numbers or even find it. (Good enough approximation for the path may find the right shortest path and thus its length.)

3. Given the shortest distances, how hard it is to compute witnesses for the shortest paths? Possibly, this can be solved in $O(n^2)$ time. Our recent algorithms [3] use additional Boolean matrix multiplications and thus require at least $\Omega(n^\omega)$ time, even when the distances are given.

References

[1] A. V. Aho, J. Hopcroft and J. B. Ullman, *The Design and Analysis of Computer Algorithms*, (Addison-Wesley 1974.) pp. 201–206.

[2] K. K. Ahuja, K. Mehlhorn, J. B. Orlin and R. E. Tarjan, *Faster algorithms for the shortest path problem*, Journal of the Association for Computing Machinery 37(1990) pp. 213–223.

[3] N. Alon, Z. Galil, O. Margalit and M. Naor, *On witnesses for Boolean matrix multiplications and for shortest paths*, To appear.

[4] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, Journal of Symbolic Computation 9(1990), pp. 251–280.

[5] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik 1(1959) pp. 269-271.

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

[7] H. N. Gabow, *Scaling algorithms for network problems*, Proc. 24th Annual Symp. on Foundation Of Comp. Sci. 1983 pp. 248–257

[8] L. R. Kerr, Phd Thesis, Cornell 1970.

[9] S. C. Kleene, *Representation of events in nerve nets and finite automata*, Automata studies, Princeton University Press, 1956.

[10] D. Krager, Personal communication, 1992.

[11] O. Margalit, Phd Thesis, Tel Aviv University, to appear.

[12] R. Seidel, *On the All-Pairs-Shortest-Path problem*, Proc. 24th ACM Annual Symp. on Theory of Computing 1992, to appear.

[13] V. Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik 13(1969) pp. 354–356.

[14] G. Yuval, *An algorithm for finding all shortest paths using $N^{2.81}$ infinite–precision multiplications*, Information processing letters, 4(1976) pp. 155–156.