Finding an induced path that is not a shortest path

Eli Berger
University of Haifa

Paul Seymour[1]
Princeton University, Princeton, NJ 08544

Sophie Spirkl[2]
Rutgers University, Piscataway, NJ 08854

July 18, 2019; revised March 28, 2020

[1]Supported by AFOSR grant A9550-19-1-0187 and NSF grant DMS-1800053.
[2]Current address: Princeton University, Princeton, NJ08544. This material is based upon work supported by the National Science Foundation under Award No. DMS-1802201.

Abstract

We give a polynomial-time algorithm that, with input a graph G and two vertices u, v of G, decides whether there is an induced uv-path that is longer than the shortest uv-path.

1 Introduction

All graphs in this paper are finite and simple. For a graph G and $u, v \in V(G)$, the *G-distance* $d_G(u, v)$ ($d(u, v)$ when there is no danger of confusion) is the number of edges in a shortest uv-path in G; let $d(u, v) = \infty$ if there is no such path. Let P be an induced uv-path. The *length* of P is the number of edges of P. We call P a *non-shortest uv-path (uv-NSP)* if the length of P is more than $d(u, v)$.

Given a graph G and $u, v \in V(G)$ we consider the question of whether there are two induced uv-paths of different lengths, or equivalently, whether there is a uv-NSP. Deciding this in polynomial time is surprisingly non-trivial. (It is important that we want induced paths; if we just want paths of different lengths, the question is much easier.) Our main result is the following:

1.1. *There is an algorithm that, given a graph G and $u, v \in V(G)$, decides whether there is a uv-NSP in time $O(|G|^{16})$.*

A step in the proof has the following consequence which may also be of interest:

1.2. *For fixed k, there is a polynomial-time algorithm that, given a graph G and $u, v \in V(G)$, decides whether there is an induced path between u and v in G of length exactly $d(u, v) + k$.*

We prove 1.2 in section 2, and 1.1 in section 3. Many variants of finding pairs of induced paths have been considered previously; for instance

1.3 (Bienstock [1])**.** *The following problems are NP-hard:*

- *Given $u, v \in V(G)$, decide whether there is an induced uv-path of odd (even) length.*
- *Given $u, v \in V(G)$, decide whether there are two induced uv-paths P_1 and P_2 with no edges between $V(P_1) \setminus \{u, v\}$ and $V(P_2) \setminus \{u, v\}$.*

Here are two more NP-hardness results, that are new as far as we know, but for reasons of space we omit the proofs:

1.4. *The following problem is NP-hard:*

- *Input: A graph G and $u, v \in V(G)$.*
- *Output: "Yes" if there exist two induced uv-paths P and Q such that there are no edges between $V(P) \setminus \{u, v\}$ and $V(Q) \setminus \{u, v\}$, and P is a shortest uv-path; and "No" otherwise.*

This is in contrast with 2.4, which implies that the problem is polynomial-time solvable if both P and Q are both required to be shortest paths (or at most a fixed constant amount longer than a shortest path). In view of 1.1, it is natural to ask:

1.5. *For fixed $k > 1$, is there a polynomial-time algorithm that, given a graph G and $u, v \in V(G)$, decides whether there is an induced uv-path P in G of length at least $d(u, v) + k$?*

This remains open, even for $k = 3$ (the algorithm of this paper does the case $k = 1$, and can be adjusted to do the case $k = 2$). It is necessary to fix k, because of the following:

1.6. *The following problem is NP-hard:*

- *Input: A graph G and $u, v \in V(G)$.*
- *Output: "Yes" if there exists a uv-NSP of length at least $2d_G(u, v)$ and "No" if there is no such path.*

2 Dynamic programming

A *path forest* means a graph in which every component is a path (possibly of length zero); and a *path forest in* G means an induced subgraph of G that is a path forest. (Thus it consists of a set of induced paths of G, pairwise vertex-disjoint and with no edges of G joining them.)

Let $V_1, \ldots, V_n$ be pairwise disjoint subsets of $V(G)$, with union $V(G)$, such that for all $i, j \in \{1, \ldots, n\}$, if $j \geq i+2$ then there are no edges between V_i and V_j. We call $(V_1, \ldots, V_n)$ an *altitude*. We are given a graph G and an altitude $(V_1, \ldots, V_n)$ in G, and we need to test whether there is a path forest in G with certain properties, that contains only a bounded number of vertices from each V_i. We shall see that this can easily be solved with dynamic programming.

Let $X \subseteq V(G)$, and let H, H' be path forests in G. We say they are X*-equivalent* if

- $V(H) \cap X = V(H') \cap X$;
- H, H' have the same number of components; and
- for each component P of H, there is a component P' of H' with the same ends and same length as P.

This is an equivalence relation.

Again, let $X \subseteq V(G)$. A path forest H is h*-restricted* in G relative to X if $|V(H) \cap X| \leq h$, and there are at most h components of H that have no end in X. Now let $(V_1, \ldots, V_n)$ be an altitude in G. A path forest H is h*-narrow* (with respect to $(V_1, \ldots, V_n)$) if for $1 \leq i \leq n$, $H[V_i \cup \cdots \cup V_n]$ is h-restricted in $G[V_i \cup \cdots \cup V_n]$ with respect to V_i.

Let $1 \leq i \leq n$. Let $\mathcal{C}_i$ be the set of all equivalence classes, under V_i-equivalence, that contain a path forest in $G[V_i \cup \cdots \cup V_n]$ that is h-narrow with respect to $(V_i, \ldots, V_n)$. Algorithmically, we may describe $\mathcal{C}_i$ by explicitly storing such a path forest.

We observe:

2.1. *If h is fixed, with G, $V_1, \ldots, V_n$ as above, for $1 \leq i < n$ we can compute $\mathcal{C}_i$ from a knowledge of $\mathcal{C}_{i+1}$ in polynomial time.*

Proof. There are only polynomially many equivalence classes in $\mathcal{C}_{i+1}$. (This is where we use the condition that at most h components of H have no end in X, in the definition of "h-restricted".) For each one, take a representive member H' say. There are only polynomially many induced subgraphs J of the graph $G[V_i \cup V_{i+1}]$ such that $V(J) \cap V_{i+1} = V(H') \cap V_{i+1}$ and $|V(J) \cap V_i| \leq h$. For each such J, check whether $H' \cup J$ is h-narrow in $G[V_i \cup \cdots \cup V_n]$ with respect to $(V_1, \ldots, V_n)$, and if so record its equivalence class under V_i-equivalence. To see that every member of $\mathcal{C}_i$ is recorded, observe that if H is a path forest in $G[V_i \cup \cdots \cup V_n]$ that is h-narrow with respect to $(V_i, \ldots, V_n)$, then $H \setminus V_i$ is a path forest in $G[V_{i+1} \cup \cdots \cup V_n]$ that is h-narrow with respect to $(V_{i+1}, \ldots, V_n)$; and if H' is another member of the equivalence class in $\mathcal{C}_{i+1}$ that contains $H \setminus V_i$, then its union with $J = H[V_i \cup V_{i+1}]$ is h-narrow with respect to $(V_1, \ldots, V_n)$ and V_i-equivalent to H. This proves 2.1. ∎

We deduce:

2.2. *For all fixed $h \geq k \geq 0$, there is a polynomial-time algorithm that, given pairs $(s_1, t_1), \ldots, (s_r, t_r)$ of a graph G, and integers $n_1, \ldots, n_r \geq 0$, and an altitude $(V_1, \ldots, V_n)$ in G, computes whether there is a path forest in G, h-restricted with respect to $(V_1, \ldots, V_n)$, with r components, where the ith component has ends s_i, t_i and has length n_i.*

Proof. First compute $\mathcal{C}_n$; then $n-1$ applications of 2.1 allow us to compute $\mathcal{C}_1$, and from $\mathcal{C}_1$ we can read off the answer. ∎

This implies 2.3, which we restate:

2.3. *For fixed k, there is a polynomial time algorithm that, given a graph G and $u, v \in V(G)$, decides whether there is an induced path between u and v in G of length exactly $d(u,v)+k$.*

We may assume that G is connected. For each $i \geq 0$, let V_i be the set of vertices with distance exactly i from u. Then $(V_1, \ldots, V_n)$ is an altitude, where n is the largest i with $V_i \neq \emptyset$. Let P be an induced uv-path of length $d(u,v)+k$. Then, for all $i \in \{1, \ldots, d(u,v)\}$, P contains a vertex x with $d(x,v) = i$. Consequently, for all $i \in \mathbb{N}_0$, P contains at most $k+1$ vertices with distance exactly i from v. So P is $(k+1)$-narrow with respect to $(V_1, \ldots, V_n)$, where n is the largest i with $V_i \neq \emptyset$. Hence 2.2, with $r = 1$ and $n_1 = d(u,v)+k$, will detect a path in the same V_1-equivalence class. ∎

Similarly, by trying all possibilities for $n_1, \ldots, n_r$, we obtain

2.4. *For fixed h and r, there is a polynomial-time algorithm with the following specifications, where V_i is the set of vertices with distance exactly i from v:*

- *Input: A graph G, $v \in V(G)$ and r pairs $(s_1, t_1), \ldots, (s_r, t_r) \in V(G)$.*
- *Output: A path forest H of G with r components $P_1, \ldots, P_r$, such that for each i, P_i has ends s_i, t_i and $|V(H) \cap V_j| \leq h$ for all $j \in \mathbb{N}$, or a determination that no such path forest exists.*

3 Finding an induced non-shortest path

In this section, we prove 1.1. We start with some definitions. A vertex $x \in V(G)$ is *uv-straight* if $d(u,x) + d(x,v) = d(u,v)$. Let G be a graph, and $u, v \in V(G)$. Let F be the set of uv-straight vertices. For $i \in \{0, \ldots, d(u,v)\}$, let $V_i = \{x \in F : d(u,x) = i\}$; we call V_i the *uv-layer of height i*, and we say its elements have *height i*; and we call the sequence $V_0, \ldots, V_{d(u,v)}$ the *uv-layering* of G. It follows that for $i, j \in \{0, \ldots, d(u,v)\}$ with $|i-j| \geq 2$, there are no edges between V_i and V_j, and moreover, for $i \in \{1, \ldots, d(u,v)-1\}$, every vertex in V_i has a neighbour in V_{i-1} and in V_{i+1}.

We call a path Q with $V(Q) \subseteq F$ *monotone* (leaving the dependence on u, v to be understood) if $|V(Q) \cap V_i| \leq 1$ for all $i \in \{0, \ldots, d(u,v)\}$ (and therefore Q is induced); and it follows that the vertices of Q are in $|V(Q)|$ uv-layers of consecutive heights. For every vertex $x \in F$, there is a monotone xu-path intersecting precisely $V_0, \ldots, V_{d(u,x)}$ and a monotone xv-path intersecting precisely $V_{d(u,x)}, \ldots, V_{d(u,v)}$, and from the definition of uv-monotonicity, it follows that both of these paths are shortest paths. If $K \subseteq V(G)$, $N(K)$ or $N_G(K)$ denotes the set of all vertices in $V(G) \setminus K$ that have a neighbour in K.

Conveniently, in order to solve 1.1 it is enough to handle the case when all vertices are uv-straight, because of the next result.

3.1. *There is a polynomial-time algorithm with the following specifications:*

- *Input: A graph G and $u, v \in V(G)$.*

- *Output: Either a uv-NSP, or a graph G' with $u, v \in V(G') \subseteq V(G)$ such that G' has a uv-NSP if and only if G has a uv-NSP, and such that every vertex of G' is uv-straight in G'.*

Proof. Let G be a graph, and $u, v \in V(G)$. We compute the set F of uv-straight vertices, and the uv-layering $V_0, \ldots, V_{d(u,v)}$ of G. We may assume that $V(G) \setminus F \neq \emptyset$, for otherwise G, u, v is the desired output.

Compute the vertex set K of a connected component of $G \setminus F$. Suppose first that $N(K)$ contains non-adjacent vertices x, y with $d(u,x) < d(u,y)$, and choose x, y such that $d(u,y) - d(u,x)$ is maximum. Let $i = d(u,x)$ and $j = d(u,y)$. It follows that no vertex in $V_0, \ldots, V_{i-1}$ has a neighbour in K (for otherwise such a vertex contradicts the choice of x); and similarly, no vertex in $V_{j+1}, \ldots, V_{d(u,v)}$ has a neighbour in K. Now let P_1 be a monotone xu-path, let P_2 be a monotone yv-path, and let Q be an induced xy-path with interior in K. It follows that the concatenation P_1-Q-P_2 is an induced uv-path; and since $V(Q) \cap K \neq \emptyset$, it follows from the definition of K and F that P_1-Q-P_2 is a uv-NSP, and we can find such a path in polynomial time.

Thus we may assume that $N(K)$ is contained in $V_i \cup V_{i+1}$ for some $i \in \{0, \ldots, d(u,v) - 1\}$, and $N(K) \cap V_i$ is complete to $N(K) \cap V_{i+1}$. Let H be obtained from G by deleting K and adding edges to make $N(K)$ a clique. We claim that H has a uv-NSP if and only if G does.

Suppose first that P is a uv-NSP of G. Since $N(K)$ is a clique of H, there is a uv-path of H with vertex set a subset of $V(P)$; let Q be the shortest such path. We claim that Q is a uv-NSP of H. If $V(P) = V(Q)$, this follows from the choice of P. Otherwise, Q contains an edge e in $E(H) \setminus E(G)$. Since e connects two vertices at the same distance from u, it follows that every induced uv-path containing e is a uv-NSP of H, as claimed, and so H has a uv-NSP.

Now suppose that Q is a uv-NSP of H. If Q does not contain an edge in $E(H) \setminus E(G)$, then Q is a uv-NSP of G, so we assume that Q contains such an edge. Since $N(K)$ is a clique of H, it follows that Q contains exactly two vertices $x, y \in N(K)$, and $xy \notin E(G)$. Let P be obtained from Q by replacing xy by an induced xy-path with interior in K. Then P is a uv-NSP of G, since P contains a vertex of K. This proves that H has a uv-NSP if and only if G does.

By repeating this procedure for all components of $G \setminus F$, we either find a uv-NSP, or the desired graph G'. ∎

3.2. *There is a polynomial-time algorithm with the following specifications:*

- *Input: A graph G and $u, v \in V(G)$ such that every vertex of G is uv-straight.*
- *Output: A uv-NSP in G, or a determination that none exists.*
- *Running time: $O(|G|^{16})$.*

Proof. For $i \in \{0, \ldots, d(u,v)\}$, let $V_i = \{x \in V(G) : d(x,u) = i\}$, and for each vertex x, let $h(x)$ be its height. Let P be a shortest uv-NSP in G (if one exists). We will prove some properties of P that will make it easier to find P.

Let P_u be the longest monotone subpath of P containing u, and let P_v be the longest monotone subpath of P containing v. Let s denote the endpoint of P_u that is not u, and let t denote the endpoint of P_v that is not v. It follows that P_u and P_v are disjoint, for otherwise P is monotone, contrary to the choice of P.

(1) *$V(P) \setminus V(P_v)$ does not contain a vertex x with $h(x) > h(s)$, and $V(P) \setminus V(P_u)$ does not contain a vertex x with $h(x) < h(t)$.*

Let $x \in V(P) \setminus V(P_v)$ be chosen with $h(x)$ maximum, breaking ties by choosing the vertex closest to u along P. Let Q be a monotone xv-path, and let P' be the subpath of P from u to x. Let Q' denote the concatenation of P' and Q. We claim that Q' is shorter than P. This follows since the subpath of P from x to v is not monotone (because $x \notin V(P_v)$), and the subpath of Q' from x to v is monotone. Since P is a shortest uv-NSP, it follows that Q' is not a uv-NSP, and hence Q' is monotone. In particular, P' is monotone. Thus $V(P') \subseteq V(P_u)$. From the choice of x, it follows that $P' = P_u$; and so $u = s$. From the choice of x, and from the symmetry between u and v, this proves (1).

Since P is not monotone, (1) immediately implies that $h(s) \geq h(t)$.

(2) *For fixed k, if $h(s) - h(t) \leq k$, then we can find a uv-NSP in polynomial time (depending on k).*

It suffices to prove (2) when $h(s) - h(t) = k$; then we obtain the desired algorithm by applying the statement for $k' = 0, \dots, k$.

Let $xy \in E(G)$ with $h(y) = h(x) + 1$, and let $v_1\text{-}\cdots\text{-}v_{k+2}$ be a $(k+2)$-vertex path with $h(v_i) = h(y) + i - 1$ for $1 \leq i \leq k+2$, such that v_1 is nonadjacent to x, and v_i is nonadjacent to x, y for $2 \leq i \leq k+2$. For all such choices of $x, y, v_1, \dots, v_{k+2}$, we proceed as follows:

- Let Q_u be a monotone path from x to u, and let Q_v be a monotone path from v_{k+2} to v.
- We delete all vertices and neighbours of $V(Q_u) \cup V(Q_v) \cup \{x\} \cup \{v_2, \dots, v_{k+2}\}$ except for y and v_1 from G. Let H denote the graph we obtain by these deletions.
- We check if H contains an induced path Q from v_1 to y. If so, we return the concatenated path

$$Q' = u\text{-}Q_u\text{-}x\text{-}y\text{-}Q\text{-}v_1\text{-}v_2\text{-}\cdots\text{-}v_{k+2}\text{-}Q_v\text{-}v.$$

First, we claim that if this returns a path Q', then Q' is a uv-NSP. From the construction of H, it follows that Q' is an induced path. Moreover, since Q' contains v_1 and y, and since $h(v_1) = h(y)$, it follows that Q' is a uv-NSP.

Now we need to show that if $h(t) = h(s) - k$, then the algorithm above always returns a path. We consider the iteration of the algorithm in which $x, y \in V(P_u)$, and $t = v_1$, and $v_1, \dots, v_{k+2} \in V(P_v)$. We claim that the subpath P' of P from v_1 to y is contained in H. Since every vertex z in $V(Q_u) \setminus \{x\}$ satisfies $h(z) \leq h(t) - 2$, it follows from (1) that z has no neighbours in P'. Similarly, no vertex in $V(Q_v)$ has a neighbour in P'. Since $x, y \in V(P_u)$, it follows that the only neighbour of x in P' is y. Since $v_1, \dots, v_{k+2} \in V(P_v)$, it follows that the only possible neighbour of $v_2, \dots, v_{k+2}$ in P' is v_1. This proves our claim. Since P' is a path from v_1 to y in H, it follows that the algorithm returns a path Q'. This proves (2).

By (2), we may assume that $h(s) - h(t) \geq 6$. Let $s_0, s_1, \dots, s_6, t_1, \dots, t_6, t_7 \in V(G)$ be distinct, such that:

- s_0-s_1-s_2-s_3, s_4-s_5-s_6, t_1-t_2-t_3, and t_4-t_5-t_6-t_7 are paths;
- $h(s_i) = h(t_i)$ for $1 \leq i \leq 6$;
- $h(s_0) + 3 = h(t_1) + 2 = h(t_2) + 1 = h(t_3) \leq h(t_4) = h(t_5) - 1 = h(t_6) - 2 = h(t_7) - 3$;
- s_i is non-adjacent to t_j for all $i \in \{0, \ldots, 6\}$ and $j \in \{1, \ldots, 7\}$.

For each such 14-tuple $s_0, s_1, \ldots, s_6, t_1, \ldots, t_6, t_7$, we do the following:

- We pick a monotone path Q_u from s_0 to u, and a monotone path Q_v from t_7 to v.
- We check using 2.4 whether there are monotone paths R_u, R_v such that R_u is an s_3s_4-path, R_v is a t_3t_4-path, and there are no edges between R_u and R_v; if not, we move on to the next 14-tuple.
- Let P'_u and P'_v be respectively the concatenations

$$u\text{-}Q_u\text{-}s_0\text{-}s_1\text{-}s_2\text{-}s_3\text{-}R_u\text{-}s_4\text{-}s_5\text{-}s_6$$

$$t_1\text{-}t_2\text{-}t_3\text{-}R_v\text{-}t_4\text{-}t_5\text{-}t_6\text{-}t_7\text{-}Q_v\text{-}v.$$

  Let H be obtained from G by deleting all vertices of $P'_u \setminus \{s_6\}$ and all their neighbours except s_6, and deleting all vertices of $P'_v \setminus \{t_1\}$ and all their neighbours except t_1. We check if there is an induced path Q from t_1 to s_6 in H, and if so, we return the concatenated path u-P'_u-s_6-Q-t_1-P'_v-v.

If this returns a path Q', then the construction implies that Q' is an induced path; and since Q' contains s_1, t_1 with $h(s_1) = h(t_1)$, it follows that Q' is a uv-NSP. It remains to show that if a shortest uv-NSP P exists with $h(s) - h(t) \geq 6$, then this algorithm returns a path. We consider the 14-tuple such that $s_6 = s$, and $t_1 = t$, $\{s_0, \ldots, s_6\} \subseteq V(P_u)$, and $\{t_1, \ldots, t_7\} \subseteq V(P_v)$. This 14-tuple exists since $h(s) - h(t) \geq 6$, and so there are at least six vertices in P_u that each have the same height as some vertex in P_v.

Now we need to show that the last bullet above returns a path. Let P' be the subpath of P from s to t. It follows from (1) that there are no edges from $V(Q_u)$ or $V(Q_v)$ to $V(P')$. Since $\{s_0, \ldots, s_6\} \subseteq V(P_u)$ and $\{t_1, \ldots, t_7\} \subseteq V(P_v)$, it follows that the only edges from $\{s_0, \ldots, s_6, t_1, \ldots, t_7\}$ to $V(P')$ are the edge from $s = s_6$ to its neighbour in $V(P')$, and the edge from $t = t_1$ to its neighbour in $V(P')$. If neither $V(R_u)$ nor $V(R_v)$ intersects or has edges to $V(P')$, then P' is present in H, and a path is returned. By symmetry, we may assume (for a contradiction) that $V(R_u)$ intersects or has edges to $V(P')$. Let z be the vertex closest to s_3 in R_u such that z has a neighbour in $V(P')$.

Let $x \in V(P')$ be the neighbour of z closest to $t = t_1$ in P'. Let R be the induced uv-path that begins with a subpath of P'_u from u to z and the edge zx, and whose remaining vertices are contained in the vertex set of the subpath of P' from x to t, and P'_v. Then R is shorter than P, since the subpath of R from u to x has length $h(z) + 1$, but in P, the subpath from u to x contains s, and thus it has length at least $h(s) + 1 > h(z) + 1$. Since R is induced, it follows that R is monotone, and therefore $h(x) > h(z)$ (and x has a neighbour in $V(P'_v)$, but we will not need this).

The concatenation Q'' of the subpath of P'_u from u to z, the edge zx, and the subpath of P from x to v is not monotone, since it contains s_1 and t_1; and as before, it is shorter than P. Therefore

Q'' is not an induced path. This implies that some vertex y of P_v has a neighbour in the subpath of R_u between s_3 and z; choose y with $h(y)$ maximum, and let z' be a neighbour of y in the subpath of R_u between s_3 and z, chosen with $h(z')$ maximum (possibly $z' = z$). It follows that y lies in the subpath of P_v between t_3, t_4.

Let t' be a vertex of the subpath of P' between x and t, such that $h(t') = h(t)$, and subject to that, the subpath of P' between x, t' is minimal. Now let R' be the concatenation of a monotone path from u to t', the subpath of P' from t' to x, the edge xz, the subpath of R_u between z and z', the edge $z'y$, and the subpath of P_v from y to v. Then R' is an induced path because of (1); and its length is at most the length of P' plus $d(u,t) + 2 + d(y,v)$; but the length of P is at least the length of P' plus $d(u,t) + 6 + d(t,v)$, and $d(t,v) \geq d(y,v)$ since $y \in V(P_v)$. This implies that R' is monotone. Since z is closer to v than x in R', it follows that $h(x) < h(z)$, a contradiction. Hence the last bullet above does indeed return a path. (We omit the analysis of running time, which is straightforward.) This proves 3.2. ∎

Now 1.1 follows from 3.1 and 3.2.

References

[1] Bienstock, Daniel. *On the complexity of testing for odd holes and induced odd paths.* Discrete Mathematics 90 (1991) 85–92. (Corrigendum, Discrete Mathematics 102 (1992) 109.)