ARTICLE

Detecting an induced subdivision of K_4

Ngoc Khang Le (iD)

LIP, ENS de Lyon, Lyon, France

Correspondence
Ngoc Khang Le, LIP, ENS
de Lyon, Lyon, France.
Email: ngoc-khang.le@ens-lyon.fr

Abstract
In this article, we propose a polynomial-time algorithm to
test whether a given graph contains a subdivision of K_4 as
an induced subgraph. This continues the study of detect-
ing an induced subdivision of H for some fixed graph H,
which is still far from being complete. Our result answers a
question by Chudnovsky *et al.* and Lévêque *et al.*

KEYWORDS
detecting, induced subgraph, ISK4

1 | INTRODUCTION

We say that a graph G *contains* some graph H if there exists an induced subgraph of G isomorphic to
H. A graph G is H*-free* if G does not contain H. For $n \geq 1$, denote by K_n the complete graph on n
vertices. A *triangle* is a graph isomorphic to K_3. A *subdivision* of a graph G is obtained by subdividing
its edges into paths of arbitrary length (at least one). For $n \geq 3$, we say that H is an *ISKn of a graph*
G if H is an induced subgraph of G and H is a subdivision of K_n. A graph that does not contain any
induced subdivision of K_n is *ISKn-free*. A *twin wheel* is a graph consisting of a chordless cycle C of
length at least four and a vertex with exactly three consecutive neighbors in C. Note that K_4 and twin
wheels are two special kinds of ISK4.

The class of ISK4-free graphs has recently been studied. In [8], a decomposition theorem for this
class is given. However, it does not lead to a recognition algorithm. The chromatic number of this class
is also proved to be bounded by 24 in [6], while it is conjectured to be bounded by 4 [8]. Given a
graph H, the *line graph* of H is the graph $L(H)$ with vertex set $E(G)$ and edge set $\{ef : e \cap f \neq \emptyset\}$.
Since the class of ISK4-free graphs contains the line graph of every cubic graph, where finding the
edge chromatic number is known to be NP-hard [4], we know that finding the chromatic number of
ISK4-free graphs is also NP-hard.

For a fixed graph H, the question of detecting an induced subdivision of H in a given graph has
been studied in [7]. There are certain graphs H where the problem is known to be NP-hard and graphs
H where there exists a polynomial algorithm. For example, detecting an induced subdivision of K_3 is
trivial since a graph is ISK3-free if and only if it is a forest. On the other hand, detecting an induced

subdivision of K_5 has been shown to be NP-hard [7]. So far, apart from the trivial cases, the only two subcubic graphs for which we have a polynomial-time algorithm to detect their induced subdivision are $K_{2,3}$ [1] and net [2]. Here, we answer the question of detecting ISK4, which was asked in [7] and [2], by proving the following:

1.1 |

There is an algorithm with the following specifications:

- **Input**: *Graph G.*
- **Output**:

  - *An ISK4 in G, or*
  - *Conclude that G is ISK4-free.*

- **Running time**: $O(n^9)$.

2 | PRELIMINARIES

We first introduce some notions that we use in this article. Let $G(V, E)$ be a graph with n vertices. For $X \subseteq V(G)$, we denote by $G \setminus X$ the subgraph of G induced by $V(G) \setminus X$. For $u \in V(G)$, let $N_G(u)$ denote the *neighborhood* of u in G - the set of neighbors of u in G and $N_G[u] = N_G(u) \cup \{u\}$. We also extend that notion for a subset $X \subseteq V(G)$, let $N_G(X) = \cup_{x \in X} N_G(x) \setminus X$ and $N_G[X] = N_G(x) \cup X$. If the context is clear, we write $N(u)$ and $N(X)$ instead of $N_G(u)$ and $N_G(X)$. For $k \geq 1$, a graph P on $\{x_1, \ldots, x_k\}$ is a *path* if $x_i x_j \in E(P)$ if and only if $|i - j| = 1$ (this is often referred to as *induced* or *chordless path* in literature). The *length* of a path is the number of its edges. The two *ends* of P are x_1 and x_k. The *interior* of P is $\{x_2, \ldots, x_{k-1}\}$. We denote by $x_i P x_j$ the subpath of P from x_i to x_j and denote by P^* the subpath of P from x_2 to x_{k-1} ($x_2 P x_{k-1}$). A *claw* is a graph on four vertices $\{u, x, y, z\}$ that has exactly three edges: ux, uy, uz. Vertex u is called the *center* of that claw.

Let x, y, z be three distinct pairwise nonadjacent vertices in G. A graph H is an (x, y, z)-*radar* in G if it is an induced subgraph of G and:

- $V(H) = V(C) \cup V(P_x) \cup V(P_y) \cup V(P_z)$.
- C is an induced cycle of length ≥ 3 containing three distinct vertices x', y', z'.
- P_x is a path from x to x', P_y is a path from y to y', P_z is a path from z to z'.
- P_x, P_y, P_z are vertex-disjoint and x', y', z' are the only common vertices between them and C.
- These are the only edges in H.

Note that the length of each path P_x, P_y, P_z could be 0, therefore an induced cycle in G passing through x, y, z is considered as an (x, y, z)-radar. We will see that 1.1 is a direct consequence of the following algorithm.

2.1 |

There is an algorithm with the following specifications:

- **Input**: *A graph G, four vertices $u, x, y, z \in V(G)$ such that $\{u, x, y, z\}$ induces a claw with center u in G.*

- **Output**: *One of the following:*

  - *An ISK4 in G, or*
  - *Conclude that there is no (x, y, z)-radar in $G' = G \setminus (N[u] \setminus \{x, y, z\})$.*

- **Running time**: $O(n^5)$.

The following is trivial:

2.2 |

An ISK4 is either K_4, a twin wheel or contains a claw.

Proof of 1.1 by 2.1. We describe an algorithm to detect an ISK4 in G as follows. First, we check if there is a K_4 or a twin wheel in G. Checking if there exists a K_4 takes $O(n^4)$. Checking if there is a twin wheel in G can be done as follows: list all 4-tuples (a, b, c, d) of vertices in G such that they induce a $K_4 \setminus e$ (a graph obtained from K_4 by removing one edge, usually called a *diamond*) where $ad \notin E(G)$; for each tuple, check if a and d are connected in $G \setminus ((N[b] \cup N[c]) \setminus \{a, d\})$. Since we have $O(n^4)$ such tuples, this can be done in $O(n^6)$. If there exists a K_4 or a twin wheel in G, then output that ISK4 in G. Otherwise, move on to next step.

Now we may assume that G is $\{K_4, \text{twin wheel}\}$-free. The following claim is true thanks to 2.2: G contains an ISK4 if and only if there exists some 4-tuple (u, x, y, z) of vertices in G such that they induce a claw with center u and there is an (x, y, z)-radar in G'. The last step in our algorithm is the following: generate every 4-tuple (u, x, y, z) of vertices in G such that they induce a claw with center u and run Algorithm 2.1 for each tuple. If for some tuple (u, x, y, z), we detect an ISK4 in G then output that ISK4 and stop. If for all the tuples, we conclude that there is no (x, y, z)-radar in G' then we can conclude that G contains no ISK4. Since we have $O(n^4)$ such tuples, and it takes $O(n^5)$ for each tuple by Algorithm 2.1 then the running time of our algorithm is $O(n^9)$. ■

The rest of our article is therefore devoted to the proof of 2.1. In the next section, we introduce some useful structures and the main proof is presented in Section 4.

3 | ANTENNAS AND CABLES

First we introduce two useful structures in our algorithm.

Let x, y, z be three distinct pairwise nonadjacent vertices in G. An (x, y, z)-*antenna* in G is an induced subgraph H of G such that:

- $V(H) = \{c\} \cup V(P_x) \cup V(P_y) \cup V(P_z)$.
- $c \notin \{x, y, z\} \cup V(P_x) \cup V(P_y) \cup V(P_z)$.
- P_x is a path from x to x', P_y is a path from y to y', P_z is a path from z to z'.
- P_x, P_y, P_z are vertex-disjoint and at least one of them has length ≥ 1.
- $cx', cy', cz' \in E(H)$.
- These are the only edges in H.
- For any vertex v in $G \setminus H$:

  - v has either at most one neighbor in H, or

- v has exactly two neighbors v_1, v_2 in H such that for some $t \in \{x, y, z\}$, $v_1, v_2 \in P_t \cup \{c\}$ and their distance in H is 1 (so they are adjacent) or 2.

We also define a *cable* given three distinct pairwise nonadjacent vertices x, y, z in G. An (x, y, z)-*cable* in G is an induced subgraph H of G such that:

- H is a path from x' to z' going through y' for some permutation (x', y', z') of $\{x, y, z\}$.
- For any vertex v in $G \setminus H$:

  - v has no neighbor in H or exactly one neighbor in H, or
  - v has exactly two neighbors v_1, v_2 in H such that for some $t \in \{x', z'\}$, v_1, v_2 are in the path $y' H t$ and their distance in H is 1 or 2, or
  - v has exactly three neighbors in H, which are y' and the two neighbors of y' in H.

Note that the existence of an (x, y, z)-antenna or an (x, y, z)-cable in G implies that there is no vertex in G adjacent to all three vertices x, y, z.

The Steiner problem is the following. Given a graph $G = (V, E)$ with nonnegative edge weights and let $S \subseteq V$ be a subset of the vertices, called *terminals*. The task is to find a minimum-weight Steiner tree—a tree in G that spans S. This problem is well-known to be NP-hard in general [5]. We will use the following algorithm, which is a direct consequence of Steiner problem in graphs for fixed number of terminals:

3.1 |

There is an algorithm with the following specifications:

- **Input**: *A graph G, a subset $X \subseteq V(G)$ of size k (k is fixed).*
- **Output**: *A minimum subgraph of G connecting every vertex in X (minimum with respect to the number of vertices).*
- **Running time**: $O(n^3)$.

Proof. By considering X as the set of terminals and the weight of every edge is 1, the solution for Steiner problem in G with k terminals gives a tree T (a subgraph of G) connecting X with minimum number of edges. Since T is a tree, the number of its vertices differs exactly one from the number of its edges, therefore graph G induced by $V(T)$ is also a solution for 3.1. An $O(n^3)$ algorithm for Steiner problem in graphs with fixed number of terminals is given in [3]. ∎

3.2 |

Given a connected graph G and three vertices $x, y, z \in V(G)$, a minimum subgraph H of G connecting x, y, z induces either:

1. *A path, or*
2. *A tree containing exactly one claw, or*
3. *The line graph of a tree containing exactly one claw.*

Proof. If there are more edges, we would find a smaller subgraph in G connecting x, y, z, a contradiction. ∎

From now on, we always denote by G, u, x, y, z the input of Algorithm 2.1 and denote by G' the graph $G \setminus (N[u] \setminus \{x, y, z\})$. The following algorithm shows that we can detect some nice structures in G' in polynomial time.

3.3 |

There is an algorithm with the following specifications:

- **Input**: G, u, x, y, z.
- **Output**: *One of the following:*

    - *An ISK4 in G, or*
    - *Conclude that there is no (x, y, z)-radar in G', or*
    - *A vertex $v \in G'$ adjacent to all three vertices x, y, z, or*
    - *An (x, y, z)-antenna H in G', or*
    - *An (x, y, z)-cable H in G'.*

- **Running time**: $O(n^3)$.

Proof. First, we check if x, y, z are connected in G' in $O(n^2)$. If they are not connected, conclude that there is no (x, y, z)-radar in G'. Now suppose that they are connected, we can find a minimum induced subgraph H of G connecting x, y, z by Algorithm 3.1. By 3.2, if H does not induce a path or a tree, output $H \cup \{u\}$ as an ISK4 in G. Therefore, we may assume that H induces a path or a tree. If H contains a vertex adjacent to both x, y, z, output that vertex and stop. Otherwise, we will prove that H must be an (x, y, z)-antenna or an (x, y, z)-cable in G', or G contains an ISK4. It is clear that now H must have the same induced structure as an antenna or a cable. We are left to prove that the attachment of a vertex $v \in G' \setminus H$ also satisfies the conditions in both cases:

- Case 1: H has the same induced structure as an (x, y, z)-antenna. Let c be the center of the only claw in H. Let x', y', z' be three neighbors of c such that x' (y', z') is the one closest to x (y, z, respectively) in H. Denote by P_x, P_y, P_z the paths from x to x', y to y', z to z' in H, respectively. Let $v \in G' \setminus H$. The following is true:

    - v cannot have neighbors in all of P_x, P_y, P_z.
    If v does, $N_H(v) = \{x', y', z', c\}$ or $N_H(v) = \{x', y', z'\}$, otherwise $(H \setminus \{c, t\}) \cup \{v\}$ is a graph connecting x, y, z that is smaller than H, where t is one of $\{x', y', z'\}$, a contradiction. If $N_H(v) = \{x', y', z', c\}$, $\{u, v, c\} \cup P_x \cup P_y$ induces an ISK4 in G. If $N_H(v) = \{x', y', z'\}$, suppose that $z' \neq z$ (since v is not adjacent to all of x, y, z), then $\{u, v, c, z'\} \cup P_x \cup P_y$ induces an ISK4 in G.

    - v has at most two neighbors in $P_x \cup \{c\}$ (this holds for P_y, P_z also).
    If v has at least four neighbors in $P_x \cup \{c\}$, let P be a shortest path from x to c in $H \cup \{v\}$, then $P \cup P_y \cup P_z$ induces a graph connecting x, y, z that is smaller than H, a contradiction. If v has exactly three neighbors in $P_x \cup \{c\}$, suppose that v has no neighbor in P_z (since v cannot have neighbors in all of P_x, P_y, P_z), then $\{u, v, c\} \cup P_x \cup P_z$ induces an ISK4 in G.

    - v cannot have neighbors in two paths among P_x, P_y, P_z.
    Without loss of generality, suppose that v has neighbors in both P_x and P_y, we might assume that v has no neighbor in P_z. If v has two neighbors in one of $P_x \cup \{c\}$ and $P_y \cup \{c\}$, suppose

that is $P_x \cup \{c\}$, let t be the neighbor of v in P_y that is closest to c. In this case, $\{u, v\} \cup P_x \cup P_z \cup tP_y y'$ induces an ISK4 in G. Therefore, v has exactly one neighbor in P_x and one neighbor in P_y and $H \cup \{u, v\}$ induces an ISK4 in G.

- If v has exactly two neighbors in $P_x \cup \{c\}$, then they must be of distance 1 or 2 in H. Otherwise, we find a graph connecting x, y, z smaller than H, a contradiction.

- Case 2: H has the same induced structure as an (x, y, z)-cable. Suppose that H is a path from x to z going through y. Let x', z' be the two neighbors of y in H such that x' is closer to x in H. Denote by P_x, P_z the paths from x to x', z to z' in H, respectively. Let $v \in G' \setminus H$. The following is true:

  - v has at most two neighbors in $P_x \cup \{y\}$.
    If v has four neighbors in $P_x \cup \{y\}$, let P be a shortest path from x to y in $H \cup \{v\}$, then $P \cup P_z$ is a subgraph connecting x, y, z that is smaller than H, a contradiction. If v has exactly three neighbors in $P_x \cup \{y\}$, then $\{u, v, y\} \cup P_x$ induces an ISK4 in G.

  - If v has neighbors in both P_x, P_z, then $N_H(v) = \{x', y, z'\}$.
    We first show that v is adjacent to y. Suppose that v is not adjacent to y. If v has two neighbors in P_x, let t be the neighbor of v in P_z that is closest to y. In this case, $\{u, v, y\} \cup P_x \cup tP_z z'$ induces an ISK4 in G. Therefore, v has exactly one neighbor in P_x and one neighbor in P_y. But now, $\{u, v\} \cup H$ induces an ISK4 in G.
    Now, v is adjacent to y. Since v has at most two neighbors in $P_x \cup \{y\}$ and two neighbors in $P_z \cup \{y\}$, v has exactly one neighbor in P_x and one neighbor in P_z. If v is not adjacent to x', let t be the neighbor of v in P_x. Now $\{u, v, y\} \cup P_z \cup xP_x t$ induces an ISK4 in G. Therefore, $N_H(v) = \{x', y, z'\}$.

  - If v has exactly two neighbors in P_x, then they must be of distance 1 or 2 in H. Otherwise, we find a graph connecting x, y, z smaller than H, a contradiction. ∎

Actually, there is an alternative way to implement Algorithm 3.3 more efficiently by not using 3.1. Basically, we only have to consider a shortest path P_{xy} from x to y, then find a shortest path from z to P_{xy}. By that we would obtain immediately an (x, y, z)-antenna or an (x, y, z)-cable. However, we use 3.1 since it gives us a more convenient proof. The first case we need to handle in Algorithm 3.3 is when there is some vertex v adjacent to all of x, y, and z.

3.4 ┃

There is an algorithm with the following specifications:

- **Input**: *G, u, x, y, z, some vertex $v \in G'$ adjacent to x, y, z.*
- **Output**: *One of the following:*

  - *An ISK4 in G, or*
  - *Conclude that v is not contained in any (x, y, z)-radar in G'.*

- **Running time**: *$O(n^2)$.*

Proof. It is not hard to see the following: v is contained in some (x, y, z)-radar in G' if and only if there exists a path from y to z in $G_x = G' \setminus ((N[x] \cup N[v]) \setminus \{y, z\})$ (up to a relabeling of x, y, z). Therefore, we only have to test if y and z are connected in G_x (and symmetries). If we find some path P from y to z in G_x, output $\{u, x, v\} \cup P$ as an ISK4. If no such path exists, we can conclude that

v is not contained in any (x, y, z)-radar in G'. Since we only have to test the connection three times (between y and z in G_x, and symmetries), the running time of this algorithm is $O(n^2)$. ∎

We also have the following algorithm to handle the antenna.

3.5 |

There is an algorithm with the following specifications:

- **Input**: *G, u, x, y, z, an (x, y, z)-antenna H in G'.*
- **Output**: *One of the following:*

    - *An ISK4 in G, or*
    - *Conclude that there is no (x, y, z)-radar in G', or*
    - *Some vertex $c \in G'$ that is not contained in any (x, y, z)-radar in G'.*

- **Running time**: *$O(n^4)$.*

Proof. Denote by c, x', y', z', P_x, P_y, P_z the elements of H as in the definition of an antenna. In this proof, we always denote by $N(X)$ the neighborhood of X in G'. First, we prove that any path connecting any pair of $\{x, y, z\}$ in $G' \setminus c$ that contains at most two neighbors of c certifies the existence of an ISK4 in G. Such a path can be found by generating every pair (v, t) of neighbors of c in G', and for each pair, finding a shortest path between each pair of $\{x, y, z\}$ in $G' \setminus (N[c] \setminus \{v, t\})$. It is clear that if such a path is found by this algorithm, then it has at most two neighbors of c and if no path is reported, we can conclude that it does not exist. Since we have $O(n^2)$ pairs (v, t) and finding a shortest path between some pair of vertices in a graph takes $O(n^2)$, this algorithm runs in $O(n^4)$. Now we prove that such a path certifies the existence of an ISK4. Let P be a path between some pair in $\{x, y, z\}$ that contains at most two neighbors of c, without loss of generality, assume that P is from x to y. We say that a path Q is a (P_x, P_y)-*connection* if one end of Q is in $N(P_x)$, the other end is in $N(P_y)$ and $Q^* \cap (N(P_x) \cup N(P_y)) = \emptyset$ (we make symmetric definitions for (x, z) and (y, z)). We also say that a path Q is S-*independent* for some $S \subseteq V(G')$ if $Q \cap N[S] = \emptyset$. We consider following cases:

1. *P contains no neighbor of c.*
   It is clear that there exists a subpath P' of P such that P' is a (P_x, P_y)-connection. Furthermore, we may assume that P' is P_z-independent since otherwise there exists some subpath P'' of P' that is a (P_x, P_z)-connection and is P_y-independent. Let x'', y'' be two ends of P' that are in $N(P_x)$ and $N(P_y)$, respectively. We see that $x'' \neq y''$ because H is an antenna. Moreover, x'' and y'' are not adjacent to c since P' contains no neighbor of c. We have the following cases based on the attachment on an antenna:

   1. x'' and y'', each has exactly one neighbor in P_x and P_y, respectively. Then $\{u\} \cup P' \cup H$ induces an ISK4 in G.
   2. x'' has exactly one neighbor in P_x and y'' has exactly two neighbors in P_y (or symmetric). Then $\{u, c\} \cup P' \cup P_x \cup P_y$ induces an ISK4 in G.
   3. x'' and y'', each has exactly two neighbors in P_x and P_y, respectively. Let t be the neighbor of x in P_x that is closer to c. Then $\{u, c\} \cup P' \cup t P_x x' \cup P_y \cup P_z$ induces an ISK4 in G.

2. *P contains exactly one neighbor of c.*

Similar to the argument of the previous case, there exists a path P' with two ends x'' and y'' such that P' is a (P_x, P_y)-connection and is P_z-independent. We may assume that c has exactly one neighbor c' in P', otherwise we are back to previous case. In this case, at most one vertex in $\{x'', y''\}$ can be adjacent to c (in other words, at most one vertex in $\{x'', y''\}$ can be identical to c'). We consider the following cases:

1. x'' has exactly one neighbor in $P_x \cup \{c\}$ (therefore exactly one neighbor in P_x) and y'' has exactly one neighbor in $P_y \cup \{c\}$ (therefore exactly one neighbor in P_y). Then $\{u, c\} \cup P' \cup P_x \cup P_y$ induces an ISK4 in G.

2. x'' has exactly one neighbor in $P_x \cup \{c\}$ (this neighbor must be in P_x) and y'' has exactly two neighbors in $P_y \cup \{c\}$ (or symmetric). If y'' is adjacent to c, then $\{u, c\} \cup P' \cup P_x \cup P_y$ induces an ISK4 in G. Otherwise y'' has two neighbors in P_y and $\{u, c\} \cup c'P'y'' \cup P_x \cup P_y$ induces an ISK4 in G.

3. x'' has exactly two neighbors in $P_x \cup \{c\}$ and y'' has exactly two neighbors in $P_y \cup \{c\}$. Since at most one of them is adjacent to c, we might assume that y'' is not adjacent to c. Then $\{u, c\} \cup c'P'y'' \cup P_y \cup P_z$ induces an ISK4 in G.

3. P contains exactly two neighbors of c.
We may assume that P is P_z-independent since otherwise we have some subpath of P that is a (P_z, P_x)-connection or (P_z, P_y)-connection and contains at most one neighbor of c and we can argue like in preceding cases. Therefore, $\{u, c\} \cup P \cup P_z$ induces an ISK4.

It is easy to see that above argument can be turned into an algorithm to output an ISK4 in each case. Now we can describe our algorithm for 3.5. First, test if there exists a path in $G' \setminus c$ between some pair of $\{x, y, z\}$ that contains at most two neighbors of c:

1. If such a path exists, output the corresponding ISK4 in G.

2. If no such path exists, test the connection between each pair of $\{x, y, z\}$ in $G' \setminus c$:

   (a) If $\{c\}$ is a cutset in G' disconnecting some pair of $\{x, y, z\}$, then conclude that there is no (x, y, z)-radar in G'.

   (b) Otherwise, conclude that c is the vertex not contained in any (x, y, z)-radar in G'.

Now we explain why this algorithm is correct. If Case 1 happens, it outputs correctly the ISK4 by the argument above. If Case 2 happens, we know that there are only two possible cases for the connection between each pair of $\{x, y, z\}$ in $G' \setminus c$, for example for (x, y):

- x and y are not connected in $G' \setminus c$, or
- Every path from x to y in $G' \setminus c$ contains at least three neighbors of c.

Therefore, Case 2a corresponds to one of the following cases, both lead to the conclusion that there is no (x, y, z)-radar in G':

- Each pair of $\{x, y, z\}$ is not connected in $G' \setminus c$.
- x is not connected to $\{y, z\}$, while y and z are still connected in $G' \setminus c$ (or symmetric). In this case, every path from y to z in $G' \setminus c$ contains at least three neighbors of c.

If Case 2b happens, we know that each pair of $\{x, y, z\}$ is still connected in $G' \setminus c$ and furthermore every path between them contains at least three neighbors of c. This implies that c is not contained in

any (x, y, z)-radar, since if c is in some (x, y, z)-radar, we can easily find a path between some pair of $\{x, y, z\}$ in that radar containing at most two neighbors of c, a contradiction.

The complexity of the whole algorithm is still $O(n^4)$ since we can find an ISK4 in Case 1 in $O(n^2)$ and test the connection in Case 2 in $O(n^2)$. ∎

The next algorithm deals with cable.

3.6 |

There is an algorithm with the following specifications:

- **Input**: G, u, x, y, z, an (x, y, z)-cable H in G'.
- **Output**: *One of the following:*

  - *An ISK4 in G, or*
  - *Conclude that there is no (x, y, z)-radar in G', or*
  - *Some vertex $c \in G'$ that is not contained in any (x, y, z)-radar in G'.*

- **Running time**: $O(n^4)$.

Proof. Without loss of generality, we may assume that the cable H is a path from x to z containing y. Let x' be the neighbor of y in H that is closer to x and z' be the other neighbor of y in H. Let $P_x = xHx'$ and $P_z = zHz'$. In this proof, we denote by $N(X)$ the neighborhood of X in G'. We also say that a path Q is a (P_x, P_z)-*connection* if one end of Q is in $N(P_x)$, the other end is in $N(P_z)$ and $Q^* \cap (N(P_x) \cup N(P_z)) = \emptyset$. Before the algorithm, we first prove the following:

(1) Every path P from x to z in $G' \setminus y$ containing no neighbor of y certifies an ISK4 in G.

   Let P' be a subpath of P such that P' is a (P_x, P_z)-connection. Let x'' and z'' be two ends of P' such that $x'' \in N(P_x)$ and $z'' \in N(P_z)$. Since P' has no neighbor of y, both x'' and z'' are not adjacent to y. We consider the following cases based on the attachment on a cable:

   (a) x'' and z'', each has exactly one neighbor in P_x and P_z, respectively. Then $\{u\} \cup H \cup P'$ induces an ISK4 in G.

   (b) x'' has exactly two neighbors in P_x and z'' has exactly one neighbor in P_z (or symmetric). Let t be the neighbor of z'' in P_z.
      - If $t \neq z$ then $\{u, y\} \cup P' \cup P_x \cup tP_zz'$ induces an ISK4 in G.
      - If $t = z$ then $\{u, y, z\} \cup P' \cup P_x$ induces an ISK4 in G.

   (c) x'' and z'', each has exactly two neighbors in P_x and P_z, respectively. Let t be one of the two neighbors of z'' that is closer to y. Then $\{u, y\} \cup P' \cup P_x \cup tP_zz'$ induces an ISK4 in G.

(2) Every path P from x to z in $G' \setminus y$ containing exactly two neighbors of y certifies an ISK4 in G.

   It is clear since $\{u, y\} \cup P$ induces an ISK4 in G.

(3) Assume that every path from x to z in $G' \setminus y$ contains at least one neighbor of y. If there exists some path from x to z in $G' \setminus y$ containing exactly one neighbor of y, then a shortest such path P satisfies that $P \cup \{y\}$ is an (x, y, z)-antenna in G', or G contains an ISK4.

   Let us prove (3). It is clear that $P \cup \{y\}$ has the same induced structure as an antenna, we only have to prove the attachment on it. Let c be the only neighbor of y on P and x', z' be the two

neighbors of c different from y such that x' is the one closer to x in P. Denote $P_x = xPx'$, $P_z = zPz'$. Let v be a vertex in $G' \setminus (P \cup \{y\})$, we consider the following cases:

- v is not adjacent to y. The following is true:
  - v cannot have neighbors on both P_x and P_z.
    If v does, there exists a path in G' from x to z (passing through v) containing no neighbor of y, a contradiction.
  - v has at most two neighbors in $P_x \cup \{c\}$.
    If v has at least three neighbors in $P_x \cup \{c\}$, they must be exactly three consecutive neighbors in P, otherwise there exists a shorter path than P satisfying the assumption. But if v has three consecutive neighbors in $P_x \cup \{c\}$, then $\{u, v\} \cup P$ induces an ISK4.
  - If v has exactly two neighbors in $P_x \cup \{c\}$, they must be of distance 1 or 2 in P.
    Otherwise, we have a shorter path than P (passing through v) satisfying the assumption.
- v is adjacent to y. The following is true:
  - v cannot have neighbors on both P_x and P_z.
    If v does, then $N(v) \cap P_x = \{x'\}$ and $N(v) \cap P_z = \{z'\}$, otherwise there exists a shorter path than P (passing through v) satisfying the assumption. If v is adjacent to c, then $\{u, v\} \cup P$ induces an ISK4. If v is not adjacent to c, since v cannot be adjacent to both x and z (by definition of a cable), we may assume that v is not adjacent to x (or equivalently $x \neq x'$). In this case, $\{u, v, y, c, x'\} \cup P_z$ induces an ISK4 in G.
  - v cannot have at least three neighbors in $P_x \cup \{c\}$.
    If v does, there exists a path (passing through v) from x to z containing exactly two neighbors of y (which are v and c). This path certifies an ISK4 by (2).
  - v cannot have exactly two neighbors in $P_x \cup \{c\}$.
    If v does, $\{u, v, y, c\} \cup P_x$ induces an ISK4.
  - If v has exactly one neighbor in $P_x \cup \{c\}$, it must be c.
    If v has exactly one neighbor in $P_x \cup \{c\}$ that is not c, then $\{u, v, y\} \cup P$ induces an ISK4.

The above discussion shows that either G contains an ISK4 (and we can detect in $O(n^2)$), or $P \cup \{y\}$ is an (x, y, z)-antenna in G'. This proves (3).

Now we describe our algorithm for 3.6:

1. Test if there exists a path P from x to z in $G' \setminus y$ containing no neighbor of y.
   1. If such a path exists, output an ISK4 by the argument in (1).
   2. If no such path exists, move to the next step.
2. Find a shortest path P from x to z in $G' \setminus y$ containing exactly one neighbor of y if such a path exists.
   1. If such a path P exists, by the argument in (3), either we detect an ISK4 in G, output it and stop, or we find an (x, y, z)-antenna $P \cup \{y\}$ in G', run Algorithm 3.5 given this antenna as input, output the corresponding conclusion.
   2. If no such path exists, move to the next step.
3. Test if there exists a path P from x to z in $G' \setminus y$ containing exactly two neighbors of y.
   1. If such a path P exists, output an ISK4 in G by the argument in (2).

2. If no such path exists, conclude that there is no (x, y, z)-radar in G' (since at this point, every path from x to z in $G' \setminus y$ contains at least three neighbors of y).

Step 1 can be done in $O(n^2)$ by checking the connection between x and z in $G' \setminus N[y]$. Step 2 runs in $O(n^3)$ by generating every neighbor t of y and for each t, find a shortest path between x and z in $G' \setminus (N[y] \setminus \{t\})$. Then pick the shortest one over all such paths. Since we call the Algorithm 3.5, step 2a takes $O(n^4)$. Step 3 can be done in $O(n^4)$ by generating every pair (t, w) of neighbors of y and for each pair (t, w), check the connection between x and z in $G' \setminus (N[y] \setminus \{t, w\})$. Therefore, the total running time of Algorithm 3.6 is $O(n^4)$. ∎

4 | PROOF OF 2.1

Now we sum up everything in previous section and describe the algorithm for 2.1:

1. Run Algorithm 3.3. Output is one of the following:

   i. An ISK4 in G: output it and stop.

   ii. Conclude that there is no (x, y, z)-radar in G' and stop.

   iii. A vertex v adjacent to x, y, z: run Algorithm 3.4 with v as input. Output is one of the following:

   (i) An ISK4 in G: output it and stop.

   (ii) Conclude that v is not contained in any (x, y, z)-radar in G': Run Algorithm 2.1 recursively for $(G \setminus v, u, x, y, z)$.

   iv. An (x, y, z)-antenna H in G': run Algorithm 3.5 with H as input. Output is one of the following:

   (1) An ISK4 in G: output it and stop.

   (2) Conclude that there is no (x, y, z)-radar in G' and stop.

   (3) Some vertex $c \in G'$ that is not contained in any (x, y, z)-radar in G': Run Algorithm 2.1 recursively for $(G \setminus c, u, x, y, z)$.

   v. An (x, y, z)-cable H in G': run Algorithm 3.6 with H as input. Output is one of the following:

   (1) An ISK4 in G: output it and stop.

   (2) Conclude that there is no (x, y, z)-radar in G' and stop.

   (3) Some vertex $c \in G'$ that is not contained in any (x, y, z)-radar in G': Run Algorithm 2.1 recursively for $(G \setminus c, u, x, y, z)$.

The correctness of this algorithm is based on the correctness of the Algorithms 3.3, 3.4, 3.5, and 3.6. Now we analyse its complexity. Let $f(n)$ be the complexity of this algorithm. Since we have five cases, each case takes $O(n^4)$ and at most a recursive call with the complexity $f(n-1)$. Therefore $f(n) \leq O(n^4) + f(n-1)$ and $f(n) = O(n^5)$.

5 | CONCLUSION

In this article, we give an $O(n^9)$ algorithm to detect an induced subdivision of K_4 in a given graph. We believe that the complexity might be improved to $O(n^7)$ by first decomposing the graph by clique cutset until there is no $K_{3,3}$ (using decomposition theorem in [8]). Now every (ISK4, $K_{3,3}$)-free graph

has a linear number of edges since it is c-degenerate by some constant c as shown in [8]. Therefore, testing the connection in this graph takes only $O(n)$, instead of $O(n^2)$ as in the algorithm. Also, we only have to consider $O(n^3)$ triples of three independent vertices and test every possible center of that claw at the same time instead of generating all $O(n^4)$ claws. But we prefer to keep our algorithm as $O(n^9)$ since it is simple and does not rely on decomposition theorem. We leave the following open question as conclusion:

Open question. Given a graph H of maximum degree 3, can we detect an induced subdivision of H in polynomial time?

ORCID

Ngoc Khang Le http://orcid.org/0000-0001-9458-3638

REFERENCES

[1] M. Chudnovsky and P. Seymour, *The three-in-a-tree problem*, Combinatorica **30** (2010), 387–417.

[2] M. Chudnovsky, P. Seymour, and N. Trotignon, *Detecting an induced net subdivision*, Journal of Combinatorial Theory Series B **103** (2013), 630–641.

[3] S. E. Dreyfus and R. A. Wagner, *The Steiner problem in graphs*, Networks **1** (1971), 195–207.

[4] I. Holyer, *The NP-completeness of edge-coloring*, SIAM Journal on Computing **10** (1981), 718–720.

[5] R. M. Karp, Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, Berlin, 1972, pp. 85–103.

[6] N. K. Le, *Chromatic number of ISK4-free graphs*, Graphs and Combinatorics **33** (2017), 1635–1646.

[7] B. Lévêque et al., *Detecting induced subgraphs*, Discrete Applied Mathematics **157** (2009), 3540–3551.

[8] B. Lévêque, F. Maffray, and N. Trotignon, *On graphs with no induced subdivision of K_4*, Journal of Combinatorial Theory Series B **102** (2012), 924–947.