Finding and counting small induced subgraphs efficiently

Angelos Assos

assos@mit.edu

June 21, 2023

Overview

1. Listing all simplicial vertices
2. Subgraphs on four vertices
   - Diamond-free Graphs
   - Counting the number of 4-cliques
   - Counting all subgraphs on 4 vertices

Simplicial vertices

In this section we are interested in listing all **simplicial** vertices of a simple connected graph $G(V, E)$ on n vertices and m edges.

Definition

A vertex $x \in G$ is **simplicial** if its neighborhood $N(x)$ is complete.

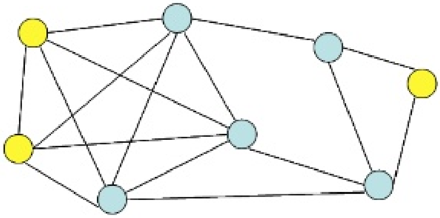

Figure: Example Graph; yellow vertices are simplicial

Simplicial vertices

Lemma 1

A vertex $x \in G$ is simplicial if and only if for every neighbor y it holds that $N[x] \subseteq N[y]$

Proof: By the definition of the simplicial vertex, if x is simplicial then it must be obvious that $N[x] \subseteq N[y]$ for every neighboring y.

For the other direction, assume that $N[x] \subseteq N[y]$ for all neighbors y of x. If x is not simplicial, then there are two neighbors of x, y and z , that are not connected. Then $z \in N[x]$ and $z \notin N[y]$ which contradicts $N[x] \subseteq N[y]$.

Simplicial vertices

Lemma 1

A vertex $x \in G$ is simplicial if and only if for every neighbor y of x it holds that $N(x) \subseteq N(y)$

Corollary 1

A vertex $x \in G$ is simplicial if and only if for every neighbor y of x it holds that $|N(x) \cap N(y)| = |N(x)|$

Take the $0/1$ adjacency matrix A of the graph G with 1s on the diagonals. Consider A^2. We will have:

$$(A^2)_{x,y} = |N[x] \cap N[y]|$$

Simple algorithm for Simplicial vertices

Corollary 1

A vertex $x \in G$ is simplicial if and only if for every neighbor y of x it holds that $|N(x) \cap N(y)| = |N(x)|$

We present a simple algorithm for listing all simplicial vertices in $O(n^\omega)$:

- Construct A as described
- Compute A^2 in $O(n^\omega)$
- Check for every vertex x if it is simplicial in $O(d(x))$ (for every neighbor y of x it must be $A^2_{x,y} = A^2_{x,x}$ from corollary 1).

$\implies$ total running time $O(n^\omega)$

Another approach

We present another algorithm for listing all simplicial vertices that uses the low degree high degree technique.

Whenever we use this technique, suppose the low degree vertices L are vertices that have degree at most D and high degree vertices have degree at least $D + 1$.

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Proof: Follows from Lemma 1.

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Proof: Follows from Lemma 1.

Key Observation 2: We can compute if a vertex x of low degree is simplicial in $O(d(x)^2)$

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Proof: Follows from Lemma 1.

Key Observation 2: We can compute if a vertex x of low degree is simplicial in $O(d(x)^2)$

Proof: We can check every pair of neighbors of x to see if they are connected which takes $O(d(x)^2)$.

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Proof: Follows from Lemma 1.

Key Observation 2: We can compute if a vertex x of low degree is simplicial in $O(d(x)^2)$

Proof: We can check every pair of neighbors of x to see if they are connected which takes $O(d(x)^2)$.

Key Observation 3: We can compute if there are simplicial high degree vertices in $O((\frac{m}{D})^\omega)$

Another approach

Key Observation 1: If a vertex of high degree is simplicial, then all of its neighbors are of high degree as well.

Proof: Follows from Lemma 1.

Key Observation 2: We can compute if a vertex x of low degree is simplicial in $O(d(x)^2)$

Proof: We can check every pair of neighbors of x to see if they are connected which takes $O(d(x)^2)$.

Key Observation 3: We can compute if a vertex x of high degree is simplicial in $O((\frac{m}{D})^\omega)$

Proof: We can disregard all high degree vertices that have neighbors in L. For the high degree vertices that remain, use the previous approach of $O(n^\omega)$ to find all simplicial vertices.

Another approach

$\Rightarrow (1),(2),(3)$ there is an $O(m^{\frac{2\omega}{\omega+1}})$ time algorithm that can list all simplicial vertices(by choosing D to be $m^{\frac{\omega-1}{\omega+1}}$)

Subgraphs on four vertices

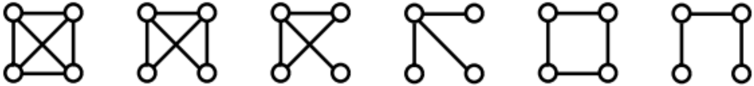

Figure: All non-isomorphic graphs on 4 vertices

Diamond-free Graphs

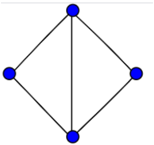

Figure: The diamond graph

Diamond graph $= K_4 - e$

Create an algorithm to check if a graph G is diamond-free.

Diamond-free Graphs

Lemma 2

A graph G is diamond-free if for every vertex x, $G[N(x)]$ is a cluster graph(a disjoint union of cliques)

Diamond-free Graphs

Lemma 2

A graph G is diamond-free if for every vertex x, $G[N(x)]$ is a cluster graph(a disjoint union of cliques)

Proof: Consider a vertex x of degree 3 in a certain diamond, the neighborhood of x contains a P_3, therefore $G[N(x)]$ is not a cluster graph. Conversely, if for some $x \in G$, $G[N(x)]$ is not a cluster graph then it must contain a P_3.

Diamond-free Graphs

Lemma 2

A graph G is diamond-free if for every vertex x, $G[N(x)]$ is a cluster graph(a disjoint union of cliques)

Proof: Consider a vertex x of degree 3 in a certain diamond, the neighborhood of x contains a P_3, therefore $G[N(x)]$ is not a cluster graph. Conversely, if for some $x \in G$, $G[N(x)]$ is not a cluster graph then it must contain a P_3.

Corollary 2

MAXIMUM CLIQUE is solvable in polynomial time($O(n(n+m))$) for diamond free graphs.

Diamond-free Graphs

Algorithm for checking if a graph G is diamond-free:

Diamond-free Graphs

Algorithm for checking if a graph G is diamond-free:

- Firstly, check if there exists a diamond with a low degree vertex of degree 3 in the diamond in $O(d(x)^2)$ for every vertex thus $O(Dm)$ in total. (Check for every $x \in L$ if $N(x)$ is a cluster graph).

Diamond-free Graphs

Algorithm for checking if a graph G is diamond-free:

- Firstly, check if there exists a diamond with a low degree vertex of degree 3 in the diamond in $O(d(x)^2)$ for every vertex thus $O(Dm)$ in total. (Check for every $x \in L$ if $N(x)$ is a cluster graph).
- Secondly, check if there exists a diamond with a low degree vertex of degree 2 in the diamond in $O(n^\omega + Dm)$ (We have the cliques from before, so for every clique C of $N(x)$ check if $y, z \in C$ have a common neighbor outside C, i.e. if $A^2_{y,z} > |C| - 1$).

Diamond-free Graphs

Algorithm for checking if a graph G is diamond-free:

- Firstly, check if there exists a diamond with a low degree vertex of degree 3 in the diamond in $O(d(x)^2)$ for every vertex thus $O(Dm)$ in total. (Check for every $x \in L$ if $N(x)$ is a cluster graph).
- Secondly, check if there exists a diamond with a low degree vertex of degree 2 in the diamond in $O(n^\omega + Dm)$ (We have the cliques from before, so for every clique C of $N(x)$ check if $y, z \in C$ have a common neighbor outside C, i.e. if $A^2_{y,z} > |C| - 1$).
- Lastly, disregard all low degree vertices and repeat the method we used in the first bullet
  $O(\sum_{x \in H} d(x)^2) = O(m|H|) = O(m^2/D)$

Diamond-free Graphs

$\implies$ we have an algorithm running in $O(n^{\omega} + m^{1.5})$ (for $D = \sqrt{m}$) that detects if a graph G is diamond free.

4-cliques

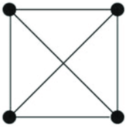

Figure: A 4-clique

We will try to count how the number of 4 cliques in G.

There are 5 different types of 4-cliques, depending on how many low degree vertices the clique has(0-4).

$L_i =$ set of cliques of size 4 that have i low degree vertices in it.
$l_i = |L_i|$

We want to compute: $K = l_0 + l_1 + l_2 + l_3 + l_4$

1^{st} equation

Computing 4-cliques containing vertices only in H

Let A be the adjacency matrix of $G[N(x) \cap H]$, $x \in H$ with $d_H(x)$ vertices

Number of cliques containing x = number of triangles in A

Triangle detection is equivalent to BMM $\implies$ we can find all triangles containing x in $O(d_H(x)^\omega)$

Running time: $O(\sum_{x \in H} d_H(x)^\omega) = O(\frac{m^\omega}{D^{\omega-1}})$

because $\sum_{x \in H} d_H(x)^\omega \leq (\sum_{x \in H} d_H(x)) \frac{(2m)^{\omega-1}}{D^{\omega-1}} = O(\frac{m^\omega}{D^{\omega-1}})$
and $d_H(x) \leq \frac{2m}{D}$

The above will give us exactly $4I_0$

2^{nd} equation

Similarly, we can compute 4-cliques containing vertices only in L

Let A be the adjacency matrix of $G[N(x) \cap L]$, $x \in L$,, with $d_L(x)$ vertices.

Number of cliques containing x = number of triangles in A

Triangle detection is equivalent to BMM $\implies$ we can find all triangles containing x in $O(d_L(x)^\omega)$

Running time: $O(\sum_{x \in H} d_H(x)^\omega) = O(mD^{\omega-1})$
because $\sum_{x \in L} d_L(x)^\omega \leq (\sum_{x \in L} d_L(x))D^{\omega-1} = O(mD^{\omega-1})$
and $d_L(x) \leq D$

The above will give us exactly $4l_4$

$3^{rd}, 4^{th}, 5^{th}$ equations

Other 3 equations found similarly:

- Number of triangles of $G[N(x) \cap H]$ for $x \in L$ will give l_1 in total time of $O(mD^{\omega-1})$
- Number of triangles of $G[N(x)]$ for $x \in L$ will give $l_1 + 2l_2 + 3l_3 + 4l_4$ in total time of $O(mD^{\omega-1})$
- Number of triangles of $G[N(x)]$ for $x \in L$ will give $2l_2 + 3l_1$ in total time of $O(mD^{\omega-1})$ - when counting triangles, count only the triangles that at least two of x neighbors are in H.

Combining results

Finally, we have 5 equations for 5 variables $\implies$ we can compute $I_0 + I_1 + I_2 + I_3 + I_4$

Running time: For $D = \sqrt{m}$ we get $O(m^{\frac{\omega+1}{2}})$

$\implies$ we can compute K in $O(m^{\frac{\omega+1}{2}})$

Finding all subgraphs on 4 vertices

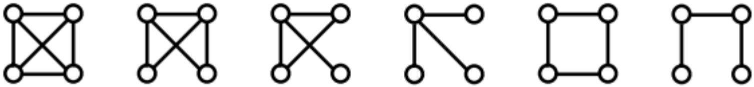

Figure: 4-clique(K),Diamond(D), Paw(Q), Claw(Y), Square(S), 4-path(P)

For K, D, Q, Y, S, P we have the following equations, where A is the adjacency matrix of G and $C = \bar{A}$:

$$\sum_{(x,y)\in E} \binom{(A^2)_{x,y}}{2} = 6K + D, \quad \sum_{(x,y)\notin E} \binom{(A^2)_{x,y}}{2} = D + 2S$$

$$\sum_{(x,y)\in E} (AC)_{x,y}(CA)_{x,y} = 4S + P, \quad \sum_{x\in V} (A^3)_{x,x} = 4D + 2P + 4Q$$

$$\sum_{(x,y)\in E} \binom{(AC)_{x,y}}{2} = Q + 3Y$$

Finding all subgraphs on 4 vertices

From the above five equations, the LHS's can be computed in BMM time, i.e. $O(n^{\omega})$

We are missing ,however, one equation ...what can we do?

Finding all subgraphs on 4 vertices

From the above five equations, the LHS's can be computed in BMM time, i.e. $O(n^\omega)$

We are missing ,however, one equation ...what can we do?

We can use the previous algorithm in which we computed K!

$\implies$ **Running time:** $O(n^\omega + m^{\frac{\omega+1}{2}})$

The end