Triangle Detection in H-Free Graphs

Amir Abboud, Ron Safier, *Nathan Wallheimer*

WEIZMANN INSTITUTE OF SCIENCE

Funded by the European Union

European Research Council

Triangle Detection

Triangle Detection

Given an n-vertex graph G, does G contain a triangle?

Triangle Detection

Triangle Detection

Given an n-vertex graph G, does G contain a triangle?

Algorithms:

- **Trivial:** Brute force over all triplets of vertices. $O(n^3)$ time.

Triangle Detection

Triangle Detection

Given an n-vertex graph G, does G contain a triangle?

Algorithms:

- **Trivial:** Brute force over all triplets of vertices. $O(n^3)$ time.
- **Fast Matrix Multiplication:** Compute the third power of the adjacency matrix A_G. $A_G^3[u, u] = \#\text{triangles incident at } u$. $O(n^{2.371\cdots})$ time.

Triangle Detection

Triangle Detection

Given an n-vertex graph G, does G contain a triangle?

Algorithms:

- **Trivial:** Brute force over all triplets of vertices. $O(n^3)$ time.
- **Fast Matrix Multiplication:** Compute the third power of the adjacency matrix A_G. $A_G^3[u, u] = \#$triangles incident at u. $O(n^{2.371\ldots})$ time.

No other $n^{3-\varepsilon}$ algorithm is known!

Combinatorial Algorithms

Combinatorial Algorithms

A combinatorial algorithm is an algorithm that avoids Fast Matrix Multiplication (FMM).

Combinatorial Algorithms

Combinatorial Algorithms

A combinatorial algorithm is an algorithm that avoids Fast Matrix Multiplication (FMM).

Abboud et al. [STOC'24]: A combinatorial Triangle Detection algorithm running in time: $n^3 \cdot 2^{-\Omega(\sqrt[7]{\log n})}$.

Combinatorial Algorithms

Combinatorial Algorithms

A combinatorial algorithm is an algorithm that avoids Fast Matrix Multiplication (FMM).

Abboud et al. [STOC'24]: A combinatorial Triangle Detection algorithm running in time: $n^3 \cdot 2^{-\Omega(\sqrt[7]{\log n})}$.

The Combinatorial Triangle Detection Conjecture

There is no combinatorial algorithm for Triangle Detection running in time $n^{3-\varepsilon}$, where $\varepsilon > 0$.

The Combinatorial Triangle Detection Conjecture

Falsifying the conjecture would give us **new hopes**:

The Combinatorial Triangle Detection Conjecture

Falsifying the conjecture would give us **new hopes**:

- A new approach to Triangle Detection in $n^{2+o(1)}$ time.

The Combinatorial Triangle Detection Conjecture

Falsifying the conjecture would give us **new hopes**:

- A new approach to Triangle Detection in $n^{2+o(1)}$ time.
- New approaches to APSP in $n^{3-\varepsilon}$ time, 3SUM in $n^{2-\varepsilon}$ time, OMv in $n^{3-\varepsilon}$ time...
  - APSP $\Rightarrow$ Minimum-Weight Triangle
  - 3SUM $\Rightarrow$ Triangle Listing

The Combinatorial Triangle Detection Conjecture

Falsifying the conjecture would give us **new hopes**:

- A new approach to Triangle Detection in $n^{2+o(1)}$ time.
- New approaches to APSP in $n^{3-\varepsilon}$ time, 3SUM in $n^{2-\varepsilon}$ time, OMv in $n^{3-\varepsilon}$ time...
  - APSP $\Rightarrow$ Minimum-Weight Triangle
  - 3SUM $\Rightarrow$ Triangle Listing

FMM does not address **weighted**, **listing**, and **online** variants of Triangle Detection.

The Combinatorial Triangle Detection Conjecture

Falsifying the conjecture would give us **new hopes**:

- A new approach to Triangle Detection in $n^{2+o(1)}$ time.
- New approaches to APSP in $n^{3-\varepsilon}$ time, 3SUM in $n^{2-\varepsilon}$ time, OMv in $n^{3-\varepsilon}$ time...
  - APSP $\Rightarrow$ Minimum-Weight Triangle
  - 3SUM $\Rightarrow$ Triangle Listing

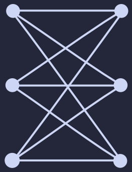
FMM does not address **weighted**, **listing**, and **online** variants of Triangle Detection.

Question
Does the conjecture hold when G has more structure?

Triangle Detection in H-Free Graphs

Definition (H-Free Graph)

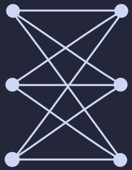
A graph G is H-free if it does not contain H as a subgraph.



Example: A complete bipartite graph is triangle-free.

Triangle Detection in H-Free Graphs

Definition (H-Free Graph)
A graph G is H-free if it does not contain H as a subgraph.



Example: A complete bipartite graph is triangle-free.

- Contains $C_4, C_6, \ldots$ as subgraphs.
- **Not** to be confused with *induced* subgraphs!

Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time | **As hard as general graphs**

Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time

- ,

As hard as general graphs

Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time

As hard as general graphs

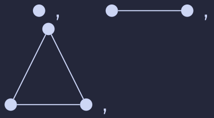
Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time



As hard as general graphs

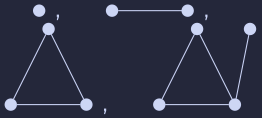
Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time



As hard as general graphs

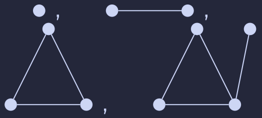
Triangle Detection in H-Free Graphs

The Objective

Classify each pattern H into one of the following classes:

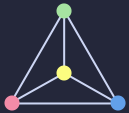
1. Triangle Detection in H-free graphs is solvable in $n^{3-\varepsilon}$ time via a combinatorial algorithm.
2. Triangle Detection in H-free graphs is as hard as in general graphs.

Combinatorially in $n^{3-\varepsilon}$ time



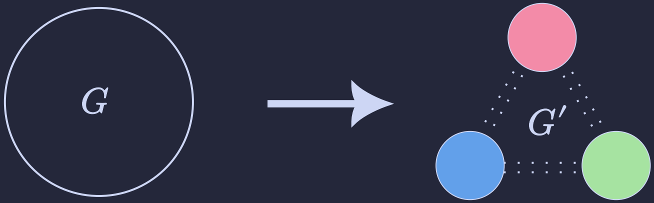
As hard as general graphs

Not 3-colorable



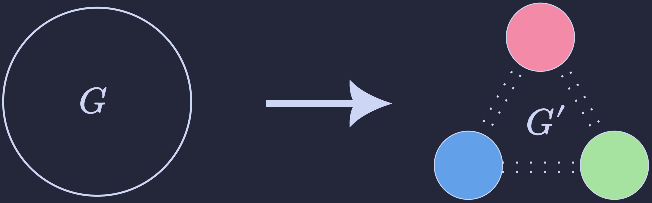
E.g., K_4

Color-Coding



Partition the vertices into 3 buckets randomly. Delete edges within each bucket.

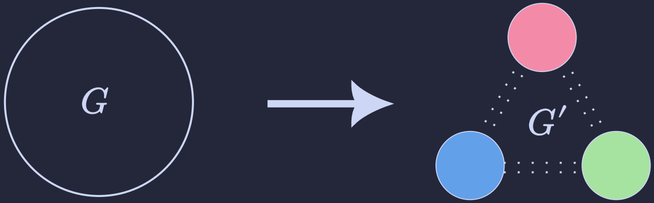
Color-Coding



Partition the vertices into 3 buckets randomly. Delete edges within each bucket.

- If G contains a triangle, G' contains a triangle with probability $\Omega(1)$.

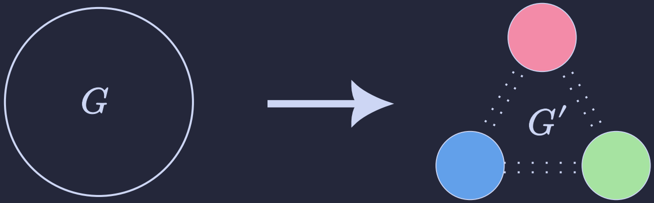
Color-Coding



Partition the vertices into 3 buckets randomly. Delete edges within each bucket.

- If G contains a triangle, G' contains a triangle with probability $\Omega(1)$.
- G' is H-free for every H which is **not** 3-colorable (e.g., K_4).

Color-Coding



Partition the vertices into 3 buckets randomly. Delete edges within each bucket.

- If G contains a triangle, G' contains a triangle with probability $\Omega(1)$.
- G' is H-free for every H which is **not** 3-colorable (e.g., K_4).

Corollary

*If H is **not** 3-colorable, then Triangle Detection in H-free graphs is as hard as Triangle Detection in general graphs.*

Bipartite Patterns

Theorem (Kovári-Sós-Turán)

If H is bipartite with color classes of size $s \leq t$, then an H-free graph has $O(n^{2-1/s})$ edges, and Triangle Detection can be solved in $O(n^{3-1/s})$ time.

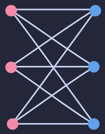
Bipartite Patterns

Theorem (Kovári-Sós-Turán)

If H is bipartite with color classes of size $s \leq t$, then an H-free graph has $O(n^{2-1/s})$ edges, and Triangle Detection can be solved in $O(n^{3-1/s})$ time.

Combinatorially in $n^{3-\varepsilon}$ time

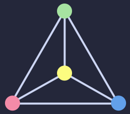
- Trivial patterns
- Bipartite patterns



E.g., Trees, even cycles, . . .

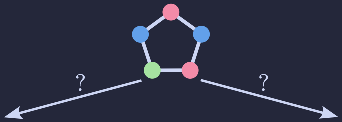
As hard as general graphs

- Not 3-colorable



E.g., K_4

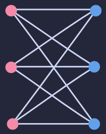
Bipartite Patterns



3-Chromatic graphs?

Combinatorially in $n^{3-\varepsilon}$ time

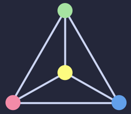
- Trivial patterns
- Bipartite patterns



E.g., Trees, even cycles, ...

As hard as general graphs

- Not 3-colorable



E.g., K_4

Our Results

Theorem
There is an $O(n^{3-1/2^k})$*-time algorithm* ($k = |V(H)|$) *for a large class of patterns that admit a special type of* 3*-coloring.*

Our Results

Theorem
There is an $O(n^{3-1/2^k})$-time algorithm ($k = |V(H)|$) for a large class of patterns that admit a special type of 3-coloring.

Corollary
A complete classification for all patterns of size < 9.

Our Results

Theorem

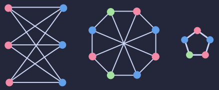
There is an $O(n^{3-1/2^k})$-time algorithm ($k = |V(H)|$) for a large class of patterns that admit a special type of 3-coloring.

Corollary

A complete classification for all patterns of size < 9.

Combinatorially in $n^{3-\varepsilon}$ time

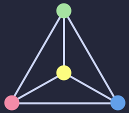
- "Nicely" 3-colored



E.g., Bipartite graphs, Odd cycles, ...

As hard as general graphs

- Not 3-colorable



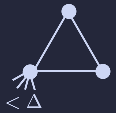
E.g., K_4

Working Example: An Algorithm for C_5-Free Graphs

Let $1 \leq \Delta \leq n$ be a threshold parameter to be set later.

Low-degree case

The triangle is incident to a vertex of degree $< \Delta$.

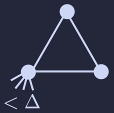


Working Example: An Algorithm for C_5-Free Graphs

Let $1 \leq \Delta \leq n$ be a threshold parameter to be set later.

Low-degree case

The triangle is incident to a vertex of degree $< \Delta$.



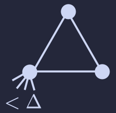
(1) For every $v \in V(G)$, if $\deg(v) < \Delta$, check if v is in a triangle in $O(\Delta^2)$ time.

Working Example: An Algorithm for C_5-Free Graphs

Let $1 \leq \Delta \leq n$ be a threshold parameter to be set later.

Low-degree case

The triangle is incident to a vertex of degree $< \Delta$.



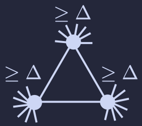
(1) For every $v \in V(G)$, if $\deg(v) < \Delta$, check if v is in a triangle in $O(\Delta^2)$ time.

(2) Remove v from G.

Working Example: An Algorithm for C_5-Free Graphs

High-degree case

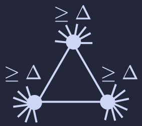
The vertices of the triangle in G have degrees $\geq \Delta$.



Working Example: An Algorithm for C_5-Free Graphs

High-degree case

The vertices of the triangle in G have degrees $\geq \Delta$.

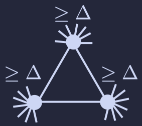


(1) Sample $\tilde{O}(n/\Delta)$ vertices from G.

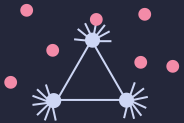
Working Example: An Algorithm for C_5-Free Graphs

High-degree case

The vertices of the triangle in G have degrees $\geq \Delta$.



(1) Sample $\tilde{O}(n/\Delta)$ vertices from G.



With high probability, a sampled vertex hits the neighborhood of the triangle.

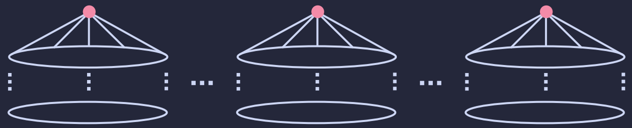
Working Example: An Algorithm for C_5-Free Graphs

(2) Check if any of the sampled vertices is part of a triangle.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

Working Example: An Algorithm for C_5-Free Graphs

(2) Check if any of the sampled vertices is part of a triangle.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

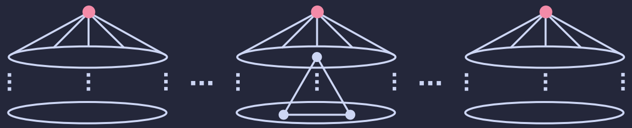
(3) Construct the radius-2 ball around each sampled vertex.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

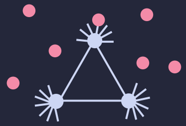


Working Example: An Algorithm for C_5-Free Graphs

(2) Check if any of the sampled vertices is part of a triangle.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

(3) Construct the radius-2 ball around each sampled vertex.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.



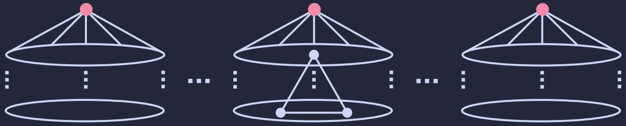


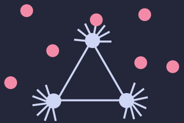
With high probability, one of the balls contains the triangle!

Working Example: An Algorithm for C_5-Free Graphs

(2) Check if any of the sampled vertices is part of a triangle.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

(3) Construct the radius-2 ball around each sampled vertex.
$\tilde{O}(n/\Delta \cdot n^2) = O(n^3/\Delta)$ time.

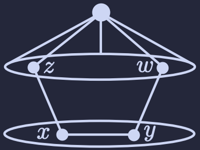




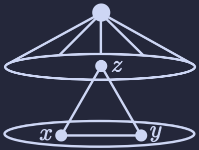
With high probability, one of the balls contains the triangle!

(5) Search for the triangle in each ball: $O(n^3)$ time.

Exploiting C_5-Freeness



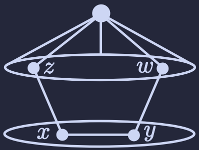
$z \neq w \implies C_5$



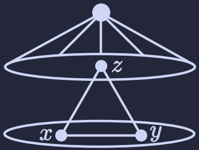
$z = w \implies$ triangle.

For every edge xy at distance 2, x and y have parents z and w at distance 1.

Exploiting C_5-Freeness



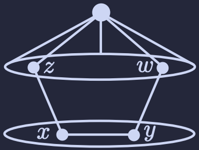
$z \neq w \implies C_5$



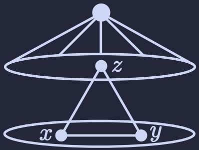
$z = w \implies$ triangle.

For every edge xy at distance 2, x and y have parents z and w at distance 1.
Key idea: An edge at distance 2 implies a triangle in C_5-free graphs!
$O(n^2)$ for finding an edge.

Exploiting C_5-Freeness



$z \neq w \implies C_5$

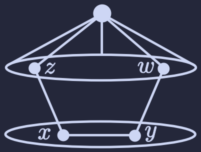


$z = w \implies$ triangle.

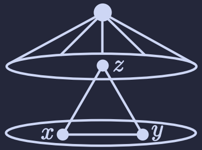
For every edge xy at distance 2, x and y have parents z and w at distance 1.
Key idea: An edge at distance 2 implies a triangle in C_5-free graphs!
$O(n^2)$ for finding an edge.

$$\text{Total time} = \underbrace{\tilde{O}(n\Delta^2)}_{\text{Low-degree case}} + \underbrace{\tilde{O}\left(\frac{n^3}{\Delta}\right)}_{\text{High-degree case}}$$

Exploiting C_5-Freeness



$z \neq w \implies C_5$



$z = w \implies$ triangle.

For every edge xy at distance 2, x and y have parents z and w at distance 1.

Key idea: An edge at distance 2 implies a triangle in C_5-free graphs!

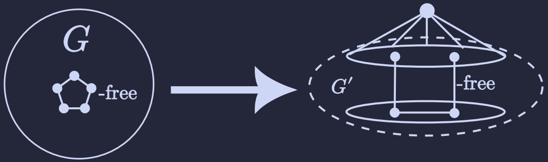
$O(n^2)$ for finding an edge.

$$\text{Total time} = \underbrace{\tilde{O}(n\Delta^2)}_{\text{Low-degree case}} + \underbrace{\tilde{O}\left(\frac{n^3}{\Delta}\right)}_{\text{High-degree case}}$$

Choose $\Delta = n^{2/3}$ to balance terms:

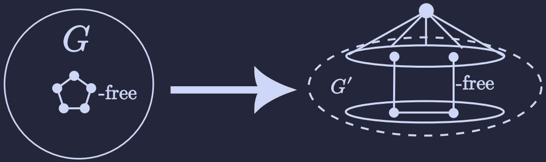
$$= \tilde{O}(n^{7/3})$$

Recursive View



Starting with C_5-free graphs, we ended up with P_4-free graphs.

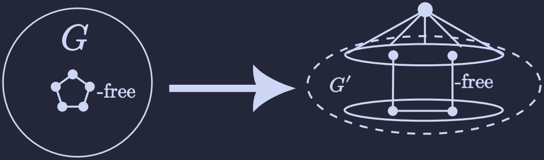
Recursive View



Starting with C_5-free graphs, we ended up with P_4-free graphs.

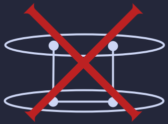
*Only P_4's whose endpoints are at distance 1, and midpoints are at distance 2.

Recursive View

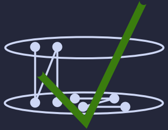


Starting with C_5-free graphs, we ended up with P_4-free graphs.

*Only P_4's whose endpoints are at distance 1, and midpoints are at distance 2.



Forbidden type of P_4's.



Permissible types of P_4's.

Colored H-Free Graphs

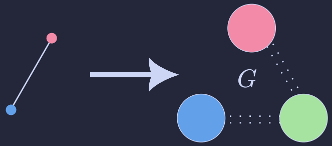
Colored H-free graph

Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.

Colored H-Free Graphs

Colored H-free graph

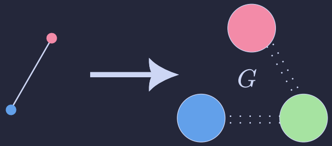
Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.



Colored H-Free Graphs

Colored H-free graph

Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.

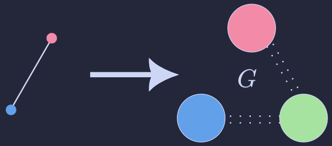


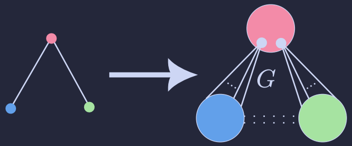
$O(1)$ time for Triangle Detection in colored H-free graph.

Colored H-Free Graphs

Colored H-free graph

Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.



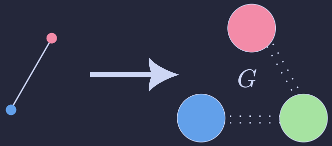


$O(1)$ time for Triangle Detection in colored H-free graph.

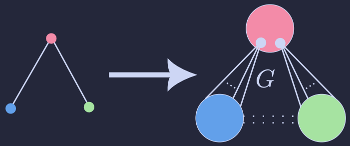
Colored H-Free Graphs

Colored H-free graph

Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.



$O(1)$ time for Triangle Detection in colored H-free graph.

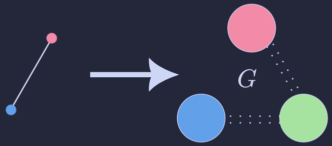


$O(1)$ time for Triangle Detection in colored H-free graph.

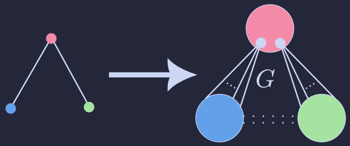
Colored H-Free Graphs

Colored H-free graph

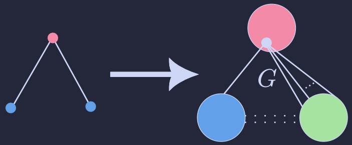
Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.



$O(1)$ time for Triangle Detection in colored H-free graph.



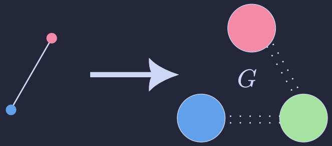
$O(1)$ time for Triangle Detection in colored H-free graph.



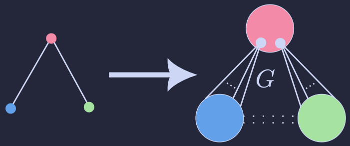
Colored H-Free Graphs

Colored H-free graph

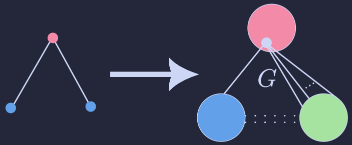
Both G and H are **colored**, and G does not contain a copy of H that preserves its colors.



$O(1)$ time for Triangle Detection in colored H-free graph.



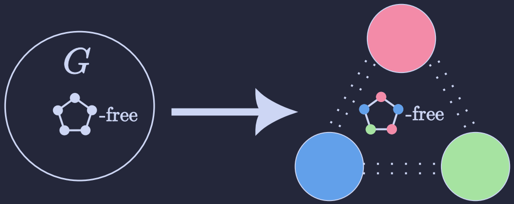
$O(1)$ time for Triangle Detection in colored H-free graph.



$O(n^2)$ time for Triangle Detection in colored H-free graph.

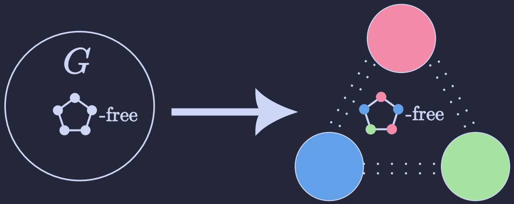
Recursive Approach for C_5-Free Graphs

(1) Color-code via random partitioning.



Recursive Approach for C_5-Free Graphs

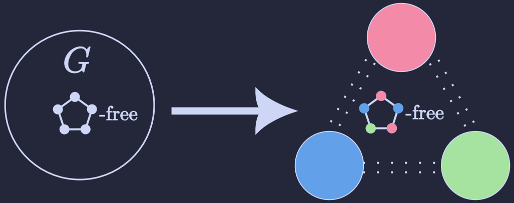
(1) Color-code via random partitioning.



WLOG, all vertices have high degrees to each color class.

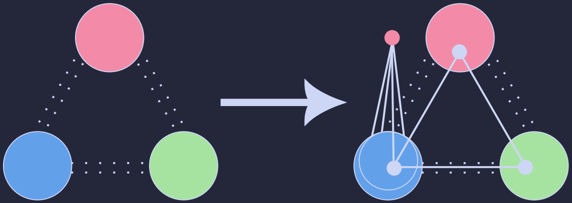
Recursive Approach for C_5-Free Graphs

(1) Color-code via random partitioning.



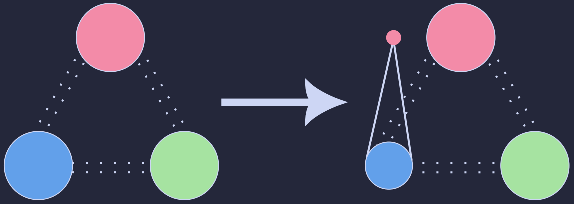
WLOG, all vertices have high degrees to each color class.

(2) Sample a red neighbor of the blue triangle vertex.



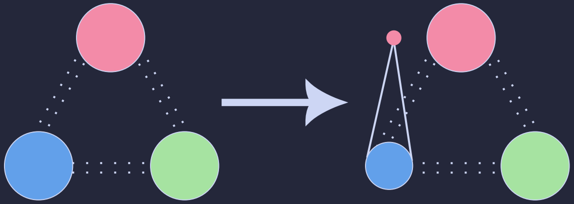
Recursive Approach for C_5-Free Graphs

(3) Restrict the blue vertices to the neighborhood of the sampled vertex.

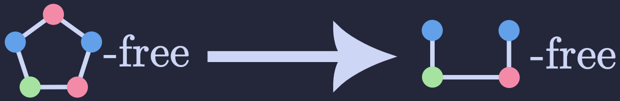


Recursive Approach for C_5-Free Graphs

(3) Restrict the blue vertices to the neighborhood of the sampled vertex.



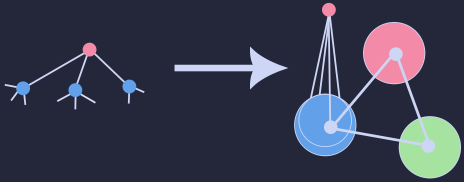
From **colored $\mathbf{C_5}$-free** graph, to **colored $\mathbf{P_4}$-free** graph.



General Recursive Approach

The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.

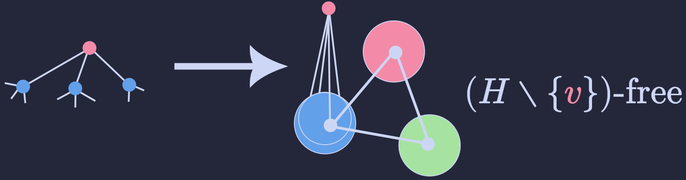


$(H \setminus \{v\})$-free

General Recursive Approach

The General Approach

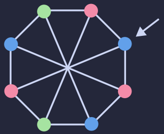
In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.



Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

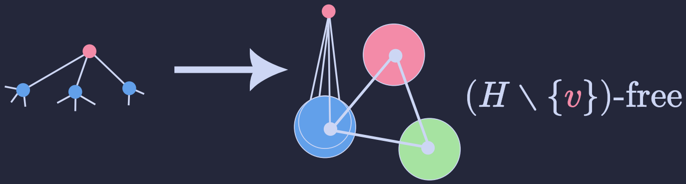
Example:



General Recursive Approach

The General Approach

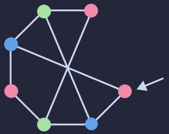
In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.



Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

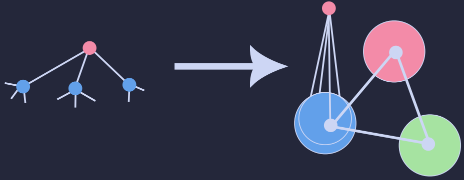
Example:



General Recursive Approach

The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.

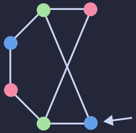


$(H \setminus \{v\})$-free

Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

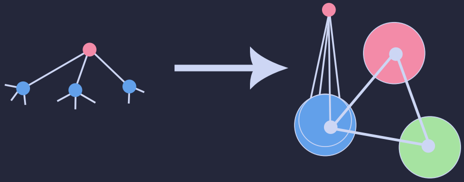
Example:



General Recursive Approach

The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.

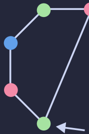


$(H \setminus \{v\})$-free

Degenerate Coloring

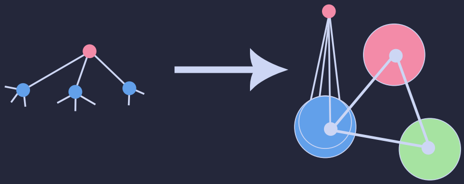
Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

Example:



The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.

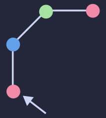


$(H \setminus \{v\})$-free

Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

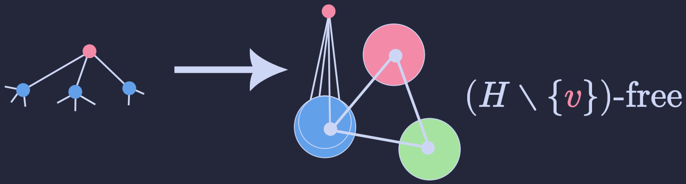
Example:



General Recursive Approach

The General Approach

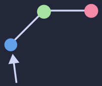
In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.



Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

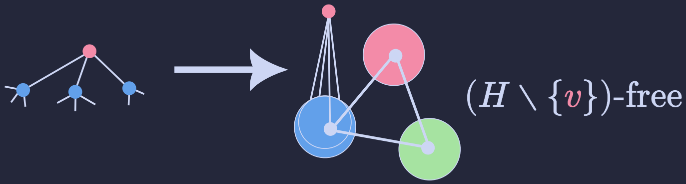
Example:



General Recursive Approach

The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.



Degenerate Coloring

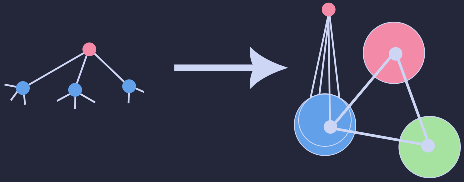
Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

Example:

General Recursive Approach

The General Approach

In each iteration, embed a vertex of H with a monochromatic neighborhood as a neighbor of the triangle.



$(H \setminus \{v\})$-free

Degenerate Coloring

Pattern H is *degenerately colored* if every induced subgraph of H contains a vertex with a monochromatic neighborhood.

Example:

Degenerately Colored Graphs

Theorem

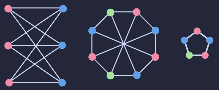
If H admits a degenerate 3-coloring, then there is an $O(n^{3-1/2^k})$ time algorithm for H-free graphs.

Degenerately Colored Graphs

Theorem
If H admits a degenerate 3-coloring, then there is an $O(n^{3-1/2^k})$ time algorithm for H-free graphs.

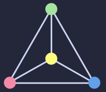
Combinatorially in $n^{3-\varepsilon}$ time

- Degenerately 3-colored



As hard as general graphs

- Not 3-colorable

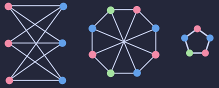


Degenerately Colored Graphs

Theorem
If H admits a degenerate 3-coloring, then there is an $O(n^{3-1/2^k})$ time algorithm for H-free graphs.

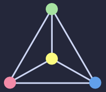
Combinatorially in $n^{3-\varepsilon}$ time

- Degenerately 3-colored

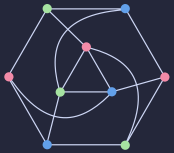


As hard as general graphs

- Not 3-colorable



The smallest unclassified pattern:



Summary

Hypothesis

The Triangle Detection problem in H-free graphs is:

- **Easy** (solvable combinatorially in $n^{3-\varepsilon}$ time) if H is 3-colorable & has ≤ 1 triangle, and
- **Hard** (as general graphs) otherwise.

Summary

Hypothesis

The Triangle Detection problem in H-free graphs is:

- **Easy** (solvable combinatorially in $n^{3-\varepsilon}$ time) if H is 3-colorable & has ≤ 1 triangle, and
- **Hard** (as general graphs) otherwise.

Open Questions:

- **Other models:** Do similar results hold in CONGEST?

Summary

Hypothesis

The Triangle Detection problem in H-free graphs is:

- **Easy** (solvable combinatorially in $n^{3-\varepsilon}$ time) if H is 3-colorable & has ≤ 1 triangle, and
- **Hard** (as general graphs) otherwise.

Open Questions:

- **Other models:** Do similar results hold in CONGEST?
- **Complexity:** Can we improve the runtime on degenerately colored graphs to $n^{3-1/k^{O(1)}}$? Prove a conditional lower bound of $n^{2+\delta}$ for some pattern H and $0 < \delta < 1$?

Summary

Hypothesis

The Triangle Detection problem in H-free graphs is:

- **Easy** (solvable combinatorially in $n^{3-\varepsilon}$ time) if H is 3-colorable & has ≤ 1 triangle, and
- **Hard** (as general graphs) otherwise.

Open Questions:

- **Other models:** Do similar results hold in CONGEST?
- **Complexity:** Can we improve the runtime on degenerately colored graphs to $n^{3-1/k^{O(1)}}$? Prove a conditional lower bound of $n^{2+\delta}$ for some pattern H and $0 < \delta < 1$?
- **Non-combinatorial algorithms:** What is the runtime if we allow Fast Matrix Multiplication?

Summary

Hypothesis

The Triangle Detection problem in H-free graphs is:

- **Easy** (solvable combinatorially in $n^{3-\varepsilon}$ time) if H is 3-colorable & has ≤ 1 triangle, and
- **Hard** (as general graphs) otherwise.

Open Questions:

- **Other models:** Do similar results hold in CONGEST?
- **Complexity:** Can we improve the runtime on degenerately colored graphs to $n^{3-1/k^{O(1)}}$? Prove a conditional lower bound of $n^{2+\delta}$ for some pattern H and $0 < \delta < 1$?
- **Non-combinatorial algorithms:** What is the runtime if we allow Fast Matrix Multiplication?

Questions?

amir.abboud@weizmann.ac.il

ron.safier@weizmann.ac.il

nathan.wallheimer@weizmann.ac.il