Motif Counting in Preferential Attachment Graphs

Jan Dreier, Peter Rossmanith

December 11, 2019

Theoretical Computer Science
RWTH Aachen University, Germany

FSTTCS 2019
How often does $H$ with $k$ edges occur in $G$ of size $n$?
How often does $H$ with $k$ edges occur in $G$ of size $n$?

- Important in network analysis and biology
How often does $H$ with $k$ edges occur in $G$ of size $n$?

- important in network analysis and biology
- many tools that work on real data for small $k$
How often does $H$ with $k$ edges occur in $G$ of size $n$?

- important in network analysis and biology
- many tools that work on real data for small $k$
- cannot be solved in time $f(k)n^{o(k)}$ under #ETH
How often does $H$ with $k$ edges occur in $G$ of size $n$?

- important in network analysis and biology
- many tools that work on real data for small $k$
- cannot be solved in time $f(k)n^{o(k)}$ under $\#\text{ETH}$
- Curticapean, Dell, Marx: in time $k^{O(k)}n^{0.174k+o(k)}$
Average Case

Analyze runtime on typical instances.
Average Case

Analyze runtime on *typical* instances.

*theoretical computer science:*
- efficient algorithms

*network science:*
- models for the real world
Average Case

Analyze runtime on *typical* instances.

*theoretical computer science*: efficient algorithms

*network science*: models for the real world

↓

efficient algorithms for the real world?
Some properties of complex networks:

- **Degree distribution**
  Most people have little but a few have a lot.

- **Small world**
  Everyone is close to everyone.

- **Clustering**
  If we have a common friend we are likely friends as well.
Some properties of complex networks:

- **Degree distribution**: Most people have little but a few have a lot.

- **Small world**: Everyone is close to everyone.

- **Clustering**: If we have a common friend we are likely friends as well.
- $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G_2^1$:
- $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G_2^1$: 

[Diagram of a graph with vertices $v_1$ and $v_2$]
Preferential Attachment

- $G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^1_2$: 

![Diagram](image-url)
Preferential Attachment

- $G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^1_2$:
- $G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^1_2$: 

  ![Diagram](v1_graph.png)
- $G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^2_2$: 

![Diagram showing a graph with vertices $v_1$ and $v_2$ connected by an edge]
$G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

$G^2_2$: 

- Graph with vertices $v_1, v_2, v_3$ and edges connecting $v_1$ to $v_2$ and $v_3$.
○ $G^n_m$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

○ $G^2_2$: 

Diagram: [Diagram of a graph with vertices $v_1$, $v_2$, and $v_3$ connected in a specific pattern]
○ $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

○ $G_2^2$:
G^n_m graph with vertices v_1, \ldots, v_n and m edges per vertex

G^3_2:
- $G^m_n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^3_2$: 

[Diagram showing a graph with vertices $v_1, v_2, v_3, v_4$ and edges connecting them]
○ $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

○ $G_2^3$: 

![Graph Diagram]
Preferential Attachment

○ $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

○ $G_2^3$: 

![Diagram of $G_2^3$]
○ $G_m^n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

○ $G_2^4$: 

![Graph Diagram](image-url)
Preferential Attachment

- $G^m_n$ graph with vertices $v_1, \ldots, v_n$ and $m$ edges per vertex

- $G^4_2$: 

![Diagram of $G^4_2$]
Our Result

**Theorem**

One can compute how often $H$ occurs in $G^n_m$ in expected FPT time $f(|H|)m^{O(|H|^6)} \log(n)^{O(|H|^{12})}n$.

In particular, for fixed $m \in \mathbb{N}$ and $H$ the run time is $\log(n)^{O(1)}n$. 


Simple Algorithm

Bound on number of certain subgraphs

Bound on degrees of individual vertices
It is sufficient to count connected subgraphs.
Counting Disconnected Subgraphs

It is sufficient to count connected subgraphs.

\[ \# \square \triangleright = \# \square \times \# \triangle \]
It is sufficient to count connected subgraphs.
Example: Counting the number of squares
Example: Counting the number of squares
Example: Counting the number of squares
Example: Counting the number of squares

\[ \text{\ldots} \]
Example: Counting the number of squares
Example: Counting the number of squares

\[ \text{\ldots} = 4 \]
Example: Counting the number of squares

\[
\begin{align*}
\quad & = \quad + \\
\quad & = \quad + \quad \ldots \\
\quad & \quad \quad 4
\end{align*}
\]
Example: Counting the number of squares

\[ \begin{align*}
&= + + \\
&= + + \ldots \\
&= 4
\end{align*} \]
Example: Counting the number of squares
Example: Counting the number of squares

\[ \begin{align*}
\text{Counting the number of squares} & = 4 \\
\end{align*} \]
Example: Counting the number of squares

\[
\begin{align*}
\text{Example: Counting the number of squares} \\
4 &= + + +
\end{align*}
\]
Example: Counting the number of squares
Example: Counting the number of squares

\[ \begin{align*}
\text{=} & \quad + \\
& \quad + \\
\end{align*} \]
Example: Counting the number of squares

\[ \begin{align*}
\quad & = 1 + 4 + \ldots \\
\quad & = 4
\end{align*} \]
Example: Counting the number of squares

\[
\begin{array}{c}
= 1 + 4 + \ldots
\end{array}
\]

\[
= 4
\]
Example: Counting the number of squares

\[ = 1 + \ldots + 4 \]
Example: Counting the number of squares

\[
\begin{align*}
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\
= & + + \\n
4

\end{align*}
\]
Example: Counting the number of squares
Example: Counting the number of squares

\[ 4 = 1 + 1 + \ldots + 4 \]
Example: Counting the number of squares

\[
\begin{align*}
1 & \quad + \quad 1 \\
\quad & \quad + \quad \ldots
\end{align*}
\]

\[= 4\]
Example: Counting the number of squares

\[ = 1 + 1 + \ldots + 4 \]
Example: Counting the number of squares

\[
4 = 1 + 1 + \ldots + 4
\]
Example: Counting the number of squares

\[= 1 + 1 + 1 + \ldots + 1 \]

\[= 4\]
Example: Counting the number of squares

\[= + +
\]

\[
\]

\[
\]

\[
\text{4}
\]
Example: Counting the number of squares
We need to bound the number of subroutine calls per neighborhood.
We need to bound the number of subroutine calls per neighborhood.

- For every subroutine call there is exactly one small leaf-free subgraph.
We need to bound the number of subroutine calls per neighborhood.

- For every subroutine call there is exactly one small leaf-free subgraph.

- Number of subroutine calls bounded by number of small leaf-free subgraphs.
Bounding Number of Leaf-Free Subgraphs

expectation of $\#\square$ in $G^n_m$
Bounding Number of Leaf-Free Subgraphs

$$= \Pr\left[ \text{expectation of } \# \square \text{ in } G^n_m \right]$$
Bounding Number of Leaf-Free Subgraphs

\[ \text{expectation of } \# \square \text{ in } G_m^n \]

\[ = \Pr \left[ \begin{array}{l}
1 \quad a \quad b \quad d \quad c \\
\end{array} \right] \]

\[ + \Pr \left[ \begin{array}{l}
1 \quad d \quad b \quad a \quad c \\
\end{array} \right] \]
Bounding Number of Leaf-Free Subgraphs

\[ \text{expectation of } \# \square \text{ in } G^n_m \]

\[ = \Pr \left[ \begin{array}{c}
1 & a & b & d & c & n \\
\end{array} \right] \]

\[ + \Pr \left[ \begin{array}{c}
1 & d & b & a & c & n \\
\end{array} \right] \]

\[ + \Pr \left[ \begin{array}{c}
1 & a & d & b & c & n \\
\end{array} \right] \]
Bounding Number of Leaf-Free Subgraphs

Expectation of \( \# \square \) in \( G_m^n \)

\[
\Pr[1 \quad \begin{array}{c} a \\ b \\ d \\ c \\ n \end{array} ] + \Pr[1 \quad \begin{array}{c} b \\ a \\ d \\ c \\ n \end{array} ] + \Pr[1 \quad \begin{array}{c} a \\ d \\ b \\ c \\ n \end{array} ] + \Pr[1 \quad \begin{array}{c} a \\ d \\ c \\ b \\ n \end{array} ] + \Pr[1 \quad \begin{array}{c} a \\ c \\ d \\ b \\ n \end{array} ]
\]
Bounding Number of Leaf-Free Subgraphs

\text{expectation of } \# \square \text{ in } G_m^n

\begin{align*}
&= \Pr \left[ \\
&+ \Pr \left[ \\
&+ \Pr \left[ \\
&+ \Pr \left[ \\
&+ \ldots \right. \\
\right. \\
\right. \\
\right. \\
\right]
\end{align*}
Bounding Number of Leaf-Free Subgraphs

\[ \Pr \left[ \text{abcd} \right] \]
Bounding Number of Leaf-Free Subgraphs

\[
Pr[ab] = \frac{1}{\sqrt{ab}}
\]
Bounding Number of Leaf-Free Subgraphs

\[
\Pr[\text{abcd}] = \log(n) \cdot \Pr[ab] \cdot \Pr[bc] \cdot \Pr[cd] \cdot \Pr[da]
\]
Bounding Number of Leaf-Free Subgraphs

\[
\Pr \left[ \frac{1}{\sqrt{ab}} \cdot \Pr[bc] \cdot \Pr[cd] \cdot \Pr[da] \right] = \frac{1}{\sqrt{bc}}
\]

\[
\Pr[ab] = \frac{1}{\sqrt{ab}} \\
\Pr[bc] = \frac{1}{\sqrt{bc}} \\
\Pr[da] = \frac{1}{\sqrt{da}}
\]
Bounding Number of Leaf-Free Subgraphs

\[ \Pr[\text{abcd}] = \frac{1}{\sqrt{ab}} \cdot \frac{1}{\sqrt{bc}} \cdot \frac{1}{\sqrt{cd}} = \Pr[ab] \cdot \Pr[bc] \cdot \Pr[cd] \]

Diagram:
- \( \Pr[ab] = \frac{1}{\sqrt{ab}} \)
- \( \Pr[bc] = \frac{1}{\sqrt{bc}} \)
- \( \Pr[cd] = \frac{1}{\sqrt{cd}} \)

Nodes: 1, a, b, d, c, n
Bounding Number of Leaf-Free Subgraphs

\[
\Pr[abc] = \frac{1}{\sqrt{ab}} \cdot \Pr[bc] \cdot \Pr[cd] \cdot \Pr[da] = \frac{1}{abcd}
\]
Bounding Number of Leaf-Free Subgraphs

\[
\Pr\left[ \text{abcd} \right] = \log(n)^{O(1)} \cdot \Pr[ab] \cdot \Pr[bc] \cdot \Pr[cd] \cdot \Pr[da] \\
= \frac{\log(n)^{O(1)}}{abcd}
\]
expectation of $\# \blacksquare$ in $G_m^n$
Bounding Number of Leaf-Free Subgraphs

expectation of $\#\Box$ in $G^n_m$

$$= \sum_{a=1}^{n}$$
Bounding Number of Leaf-Free Subgraphs

expectation of \( \# \square \) in \( G_m^n \)

\[
= \sum_{a=1}^{n} \sum_{b=1}^{n} \log(n) O(\text{one.oldstyle} n \text{two.oldstyle})
\]
Bounding Number of Leaf-Free Subgraphs

expectation of \#\begin{tikzpicture}[baseline=0pt, scale=0.5]
\draw (0,0) -- (1,0) -- (1,1) -- (0,1) -- cycle;
\end{tikzpicture} in $G_m^n$

\[
= \sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n}
\]
Bounding Number of Leaf-Free Subgraphs

expectation of $\# \square$ in $G^n_m$

$$= \sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n} \sum_{d=1}^{n}$$
Bounding Number of Leaf-Free Subgraphs

expectation of \( \# \square \) in \( G^n_m \)

\[
= \sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n} \sum_{d=1}^{n} \frac{\log(n)^{O(1)}}{abcd}
\]
Bounding Number of Leaf-Free Subgraphs

expectation of \( \# \square \) in \( G_m^n \)

\[
= \sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n} \sum_{d=1}^{n} \frac{\log(n)^{O(1)}}{abcd}
\]

\[
= \log(n)^{O(1)} \sum_{a=1}^{n} \frac{1}{a} \sum_{b=1}^{n} \frac{1}{b} \sum_{c=1}^{n} \frac{1}{c} \sum_{d=1}^{n} \frac{1}{d}
\]
Bounding Number of Leaf-Free Subgraphs

Expectation of \( \# \quad \) in \( G^m_n \)

\[
= \sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n} \sum_{d=1}^{n} \frac{\log(n)^{O(1)}}{abcd}
\]

\[
= \log(n)^{O(1)} \sum_{a=1}^{n} \frac{1}{a} \sum_{b=1}^{n} \frac{1}{b} \sum_{c=1}^{n} \frac{1}{c} \sum_{d=1}^{n} \frac{1}{d}
\]

\[
= \log(n)^{O(1)}
\]

(similarly for every leaf-free graph)
Analysis of run time:
Analysis of run time:

- Number of vertices: $O(n)$
Analysis of run time:

- Number of vertices: $O(n)$
- Subroutine calls per vertex: $\log(n)^{O(1)}$
Total Runtime

Analysis of run time:

- Number of vertices: $O(n)$
- Subroutine calls per vertex: $\log(n)^{O(1)}$
- Time for each call: $O(n)$
Analysis of run time:

- Number of vertices: $O(n)$
- Subroutine calls per vertex: $\log(n)^{O(1)}$
- Time for each call: $O(n)$

Total run time: $\log(n)^{O(1)}n^2$
Total Runtime

Analysis of run time:

- Number of vertices: $O(n)$
- Subroutine calls per vertex: $\log(n)^{O(1)}$
- Time for each call: $O(n)$

Total run time: $\log(n)^{O(1)}n^2$

With a more careful analysis: $\log(n)^{O(1)}n$
First-Order Model Checking

- \( \exists \) and \( \forall \) quantification over vertices, equality, adjacency
- cannot be solved in time \( f(|\varphi|)n^o(|\varphi|) \) under ETH
- Grohe, Kreutzer, Siebertz: for fixed formula decidable in time \( O(n^{1+\varepsilon}) \) on nowhere-dense graph classes

Theorem

Let \( \varepsilon > 0, m \in \mathbb{N} \) be fixed.
One can decide every property expressible in FO logic on \( G_m^n \) in expected time \( O(n^{1+\varepsilon}) \).
Thank you!