k-local Graphs

Christian Beth, Pamela Fleischmann, Annika Huch, Daniyal Kazempour,
Peer Kröger, Andrea Kulow, Matthias Renz

24.07.2025

Kiel University

Motivation

- k-locality on words measures complexity of patterns[1]
- Can k-locality be translated as a graph parameter?
- What can such a parameter reveal about a given graph?

[1] **Local patterns.** Day et al. FSTTCS 2017 (pp. 24-1)

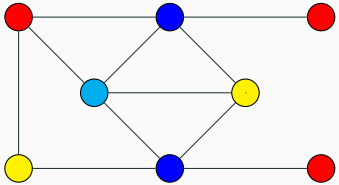
Motivation

- k-locality on words measures complexity of patterns[1]
- Can k-locality be translated as a graph parameter?
- What can such a parameter reveal about a given graph?
  ($\rightarrow$ Data Science perspective)

[1] **Local patterns.** Day et al. FSTTCS 2017 (pp. 24-1)

What does k-locality even mean?

Given are coloured graph $G = (V, E)$ and a *marking sequence* of colours.

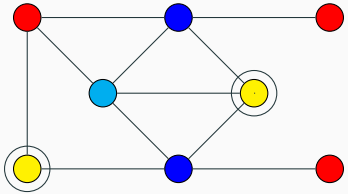


Marking sequence: $(\texttt{yellow}, \texttt{red}, \texttt{cyan}, \texttt{blue})$

If we mark G stepwise in these colours, what is the largest number of *marked* clusters?

k-locality example (i)

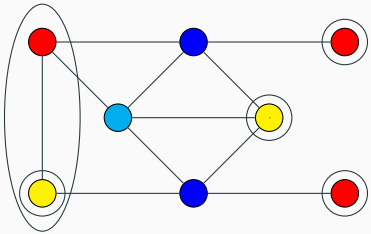
Marking sequence (yellow, red, cyan, blue)



Sequence of no. clusters: (2), (max. 2).

k-locality example (i)

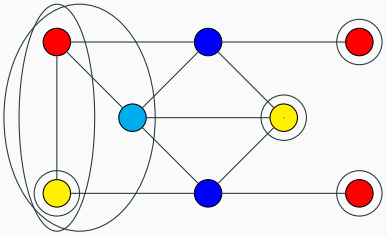
Marking sequence ($\underline{\texttt{yellow}}, \underline{\texttt{red}}, \texttt{cyan}, \texttt{blue}$)



Sequence of no. clusters: $(2, 4)$, (max. 4).

k-locality example (i)

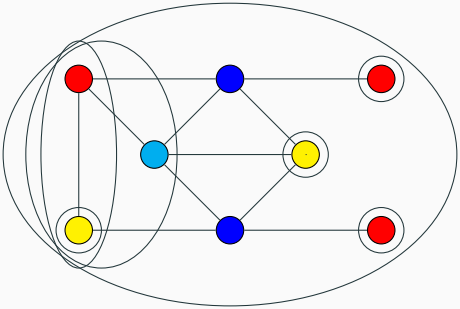
Marking sequence ($\underline{\texttt{yellow}}, \underline{\texttt{red}}, \underline{\texttt{cyan}}, \texttt{blue}$)



Sequence of no. clusters: $(2, 4, 4)$, (max. 4).

k-locality example (i)

Marking sequence (yellow, red, cyan, blue)

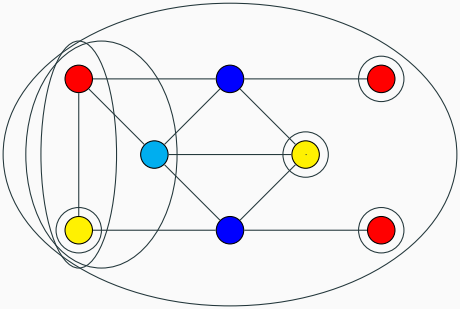


Sequence of no. clusters: $(2, 4, 4, 1)$, (max. 4)

$\Rightarrow$ G is 4-local w.r.t. this sequence

k-locality example (i)

Marking sequence (yellow, red, cyan, blue)



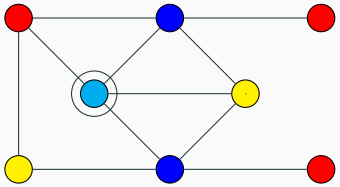
Sequence of no. clusters: $(2, 4, 4, 1)$, (max. 4)

$\Rightarrow$ G is 4-local w.r.t. this sequence

Is there a sequence, which yields smaller k-locality?

k-locality example (ii)

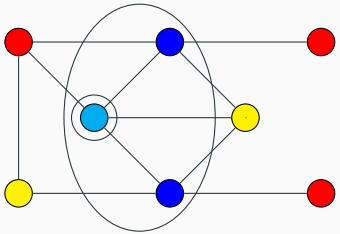
Marking sequence (cyan, blue, yellow, red)



Sequence of no. clusters: (1), (max. 1).

k-locality example (ii)

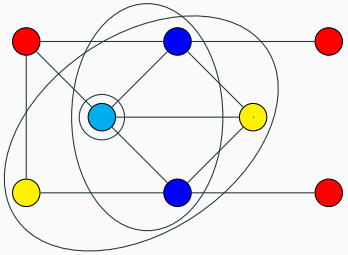
Marking sequence (cyan, blue, yellow, red)



Sequence of no. clusters: $(1, 1)$, (max. 1).

k-locality example (ii)

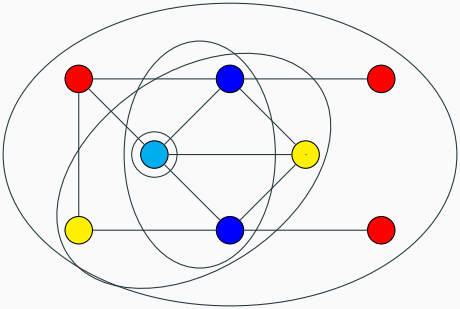
Marking sequence (cyan, blue, yellow, red)



Sequence of no. clusters: $(1, 1, 1)$, (max. 1).

k-locality example (ii)

Marking sequence (cyan, blue, yellow, red)

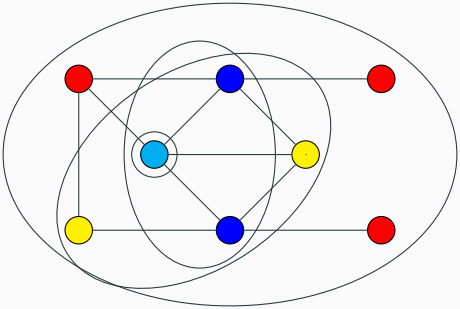


Sequence of no. clusters: $(1, 1, 1, 1)$, (max. 1)

$\Rightarrow$ G is 1-local w.r.t. this sequence.

k-locality example (ii)

Marking sequence (cyan, blue, yellow, red)



Sequence of no. clusters: $(1, 1, 1, 1)$, (max. 1)

$\Rightarrow$ G is 1-local w.r.t. this sequence.

What is the minimal k-locality of a graph? Can we find it efficiently?

Complexity

Problem.
Let Loc_G be the problem of deciding whether for a connected graph G with a valid ℓ-colouring c with $\ell \in \mathbb{N}$, we have $\mathrm{loc}(G, c) \leq k$.
The subproblem to decide whether we have $\mathrm{loc}(G, c) = 1$ is abbreviated by $\mathrm{Loc}_{G,1}$.

Complexity

Problem.
Let Loc_G be the problem of deciding whether for a connected graph G with a valid ℓ-colouring c with $\ell \in \mathbb{N}$, we have $\mathrm{loc}(G, c) \leq k$.
The subproblem to decide whether we have $\mathrm{loc}(G, c) = 1$ is abbreviated by $\mathrm{Loc}_{G,1}$.

Theorem.
For every undirected graph G, we have $\mathrm{Loc}_{G,1} \in \mathrm{P}$.

Complexity

Problem.
Let Loc_G be the problem of deciding whether for a connected graph G with a valid ℓ-colouring c with $\ell \in \mathbb{N}$, we have $\mathrm{loc}(G, c) \leq k$.
The subproblem to decide whether we have $\mathrm{loc}(G, c) = 1$ is abbreviated by $\mathrm{Loc}_{G,1}$.

Theorem.
For every undirected graph G, we have $\mathrm{Loc}_{G,1} \in \mathrm{P}$.

Proposition.
For an undirected graph G, the problem Loc_G is NP-complete.

k-locality on bipartite graphs

Proposition.
Let $G = (V_1 \dot\cup V_2, E)$ be a bipartite graph that is 2-coloured.
Then G is strictly $\min\{|V_1|, |V_2|\}$-local.

k-locality on bipartite graphs

Proposition.
Let $G = (V_1 \dot\cup V_2, E)$ be a bipartite graph that is 2-coloured.
Then G is strictly $\min\{|V_1|, |V_2|\}$-local.

Theorem.
Let G be a bipartite graph that is coloured by c with more than $\ell > 2$ colours.
Then $(G, c, k) \in \mathrm{Loc}_G$ is NP-complete.

Upper and Lower Bounds for the Locality of Different Graph Classes

Class	Lower Bound	Upper Bound
Complete Graphs, Star Graphs Wheel Graphs, Friendship Graphs	1	1
Paths Cycles, Web graphs 1/2-regular Graphs	1	$\lfloor n/2 \rfloor$
Complete Bipartite Graphs K_{n_1,n_2} Crown Graphs, Hypercubes Bipartite Graphs with 2-colouring Knight's graph, Gear graph	1	$\min\{n_1, n_2\}$
Sunflower graphs S_d, Helm graphs H_d	1	$\lfloor \frac{d-1}{2} \rfloor$
Peterson Graph	1	3
Edgeless Graphs, 0-regular Graphs	n	n

Practical Applications?

- Data exploration: "Getting to know your data"
- Measure and optimize utility for location types (groceries, pharmacies, restaurants etc.) in a road network

Practical Applications?

- Data exploration: "Getting to know your data"
- Measure and optimize utility for location types (groceries, pharmacies, restaurants etc.) in a road network

$\rightarrow$ requires reasonably efficient calculation to be practical

How do we calculate k-locality? - A naive approach

- Enumerate *all* marking sequences and keep the one with the lowest k-locality
- Obvious downside: runs in $\mathcal{O}(|C|! \cdot (|V| + |E|))$

How do we calculate k-locality? - A naive approach

- Enumerate *all* marking sequences and keep the one with the lowest k-locality
- Obvious downside: runs in $\mathcal{O}(|C|! \cdot (|V| + |E|))$
- But:
  - When checking a complete sequence, we always count the clusters for the prefices first

How do we calculate k-locality? - A naive approach

- Enumerate *all* marking sequences and keep the one with the lowest k-locality
- Obvious downside: runs in $\mathcal{O}(|C|! \cdot (|V| + |E|))$
- But:
  - When checking a complete sequence, we always count the clusters for the prefices first
  - Many of the prefices overlap

How do we calculate k-locality? - A naive approach

- Enumerate *all* marking sequences and keep the one with the lowest k-locality
- Obvious downside: runs in $\mathcal{O}(|C|! \cdot (|V| + |E|))$
- But:
  - When checking a complete sequence, we always count the clusters for the prefices first
  - Many of the prefices overlap
  - Once we have a complete candidate sequence with k-lokality x, we may exclude all *prefices* with locality greater than x

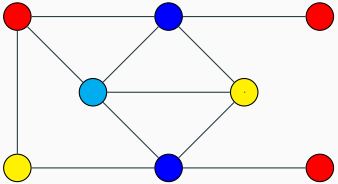
How do we calculate k-locality? - A naive approach

- Enumerate *all* marking sequences and keep the one with the lowest k-locality
- Obvious downside: runs in $\mathcal{O}(|C|! \cdot (|V| + |E|))$
- But:
  - When checking a complete sequence, we always count the clusters for the prefices first
  - Many of the prefices overlap
  - Once we have a complete candidate sequence with k-lokality x, we may exclude all *prefices* with locality greater than x

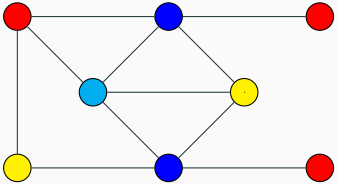
  $\rightarrow$ priority search (a.k.a. best first search) in a prefix tree over marking sequences

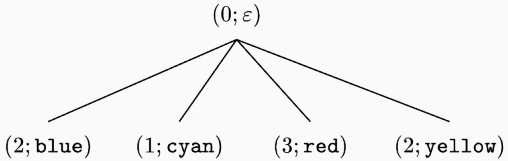
How do we calculate k-locality? - Priority Search

$(0; \varepsilon)$

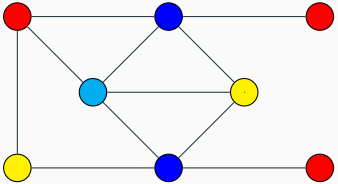


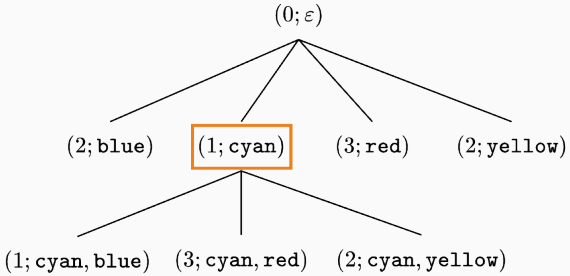
How do we calculate k-locality? - Priority Search



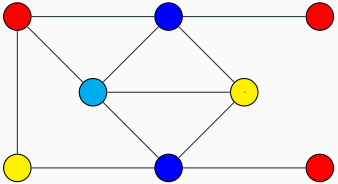


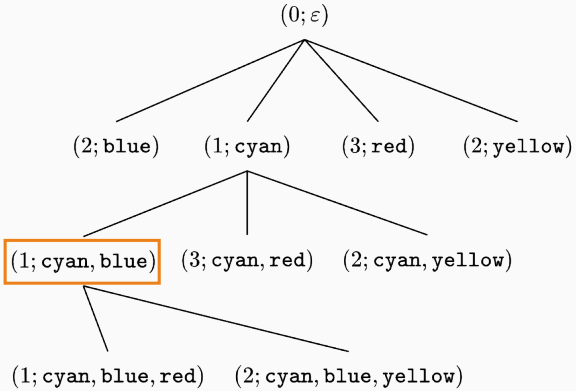
How do we calculate k-locality? - Priority Search



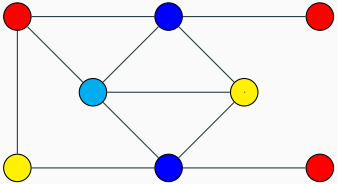


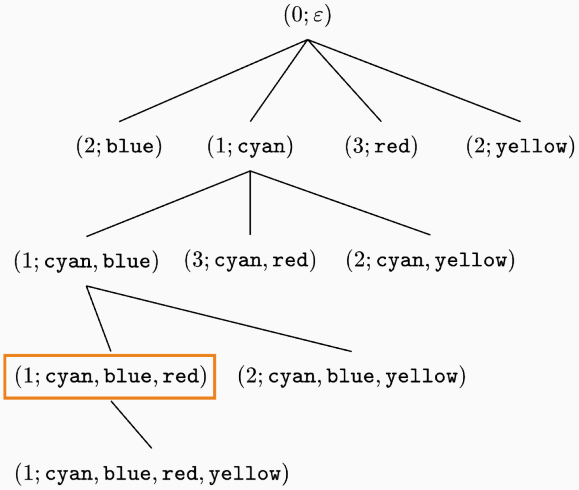
How do we calculate k-locality? - Priority Search





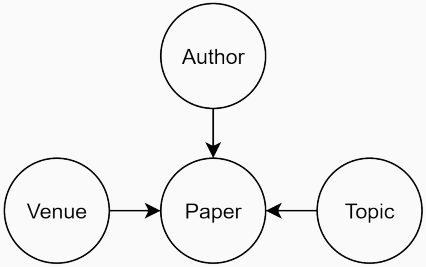
How do we calculate k-locality? - Priority Search





Case Study: DBLP (4 Areas Subset)

- Small subset of DBLP graph covering 4 research areas in 2009
- 5.9k author vertices, 5.2k papers, 18 venues, 4.4k topics
- In total about 53k edges



DBLP network Schema

Case Study: DBLP (4 Areas Subset)

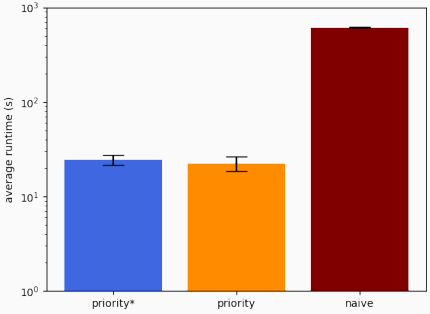
- DBLP subset is 18-local
- The two minimal sequences are (V, P, A, T) and (V, P, T, A)
- The next smallest sequences have k-locality larger than 4.4k

$\rightarrow$ Venues are very central to the graph

Case Study: DBLP (4 Areas Subset)

- DBLP subset is 18-local
- The two minimal sequences are (V, P, A, T) and (V, P, T, A)
- The next smallest sequences have k-locality larger than 4.4k

$\rightarrow$ Venues are very central to the graph



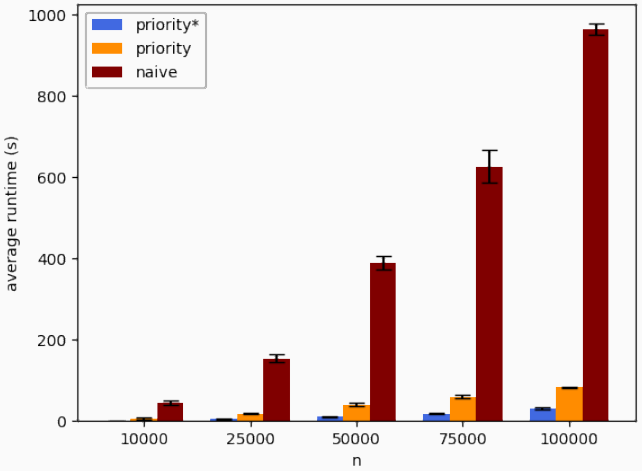
DBLP runtime comparison (5 runs)

Testing Scalability: Barabási-Albert Random Graphs

- We use the Barabási-Albert random graph model to create 'natural' graphs of varying sizes[2]
- The graph is grown from an initial node set based on *preferential attachment*
- Node types/colours are distributed uniformly at random.

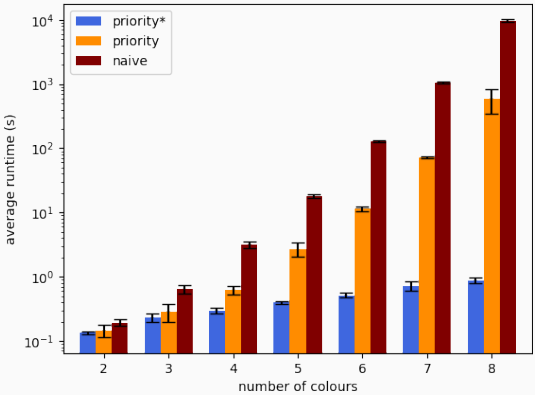
[2] **Statistical mechanics of complex networks.** R. Albert, A.-L. Barabási. In: Rev. Mod. Phys. 74 (1 2002), pp. 47–97

Scalability w.r.t. Graph Size



$|C| = 5$ (measured over 5 runs)

Scalability w.r.t. the Number of Colours



$|V| = 5000$ (measured over 5 runs)

Conclusion

- We proposed k-locality on graphs as a new parameter

Conclusion

- We proposed k-locality on graphs as a new parameter
- k-locality on graphs is NP-complete

Conclusion

- We proposed k-locality on graphs as a new parameter
- k-locality on graphs is NP-complete
- We developed an optimal priority search algorithm for its calculation

Conclusion

- We proposed k-locality on graphs as a new parameter
- k-locality on graphs is NP-complete
- We developed an optimal priority search algorithm for its calculation
- k-locality may be useful for data exploration (cf. DBLP case-study)

Conclusion

- We proposed k-locality on graphs as a new parameter
- k-locality on graphs is NP-complete
- We developed an optimal priority search algorithm for its calculation
- k-locality may be useful for data exploration (cf. DBLP case-study)
- Ongoing and future work:
  - Investigate utility optimization in urban planning
  - Relation to other graph parameters, e.g., neighbourhood diversity.

Appendix

Priority Search: Finding all minimal marking sequences

Algorithm 1: Priority Search

Input : Graph $G = (V, E)$, valid colouring $c : V \to [\ell]$
Output : minimum k-locality of G, *all* marking sequences of that k-locality

```
1  min_k ← ∞
2  minSequences ← ∅
3  queue ← [(0, ε)]                                   // [(max_k, prefix), ...] sorted asc. by max_k
4  while queue ≠ ∅ ∧ queue[0][0] ≤ min_k do
5      (current_k, prefix) ← queue.pop()
6      if prefix incomplete then
7          forall remaining colours c_j not in prefix do
8              count ← no. components in G marked with prefix + [c_j]
9              queue.insort((max(current_k, count), prefix + [c_j]))
10     else                                           // current sequence is complete
11         if current_k = min_k then
12             minSequences ← minSequences ∪ {prefix}
13         if current_k ¡ min_k then
14             min_k ← current_k
15             minSequences ← {prefix}
16 return min_k, minSequences
```

Priority Search*: Finding only the first minimal marking sequence

Algorithm 2: Priority Search*

Input : Graph $G = (V, E)$, valid colouring $c : V \to [\ell]$
Output : minimum k-locality of G, *one* marking sequence of that k-locality

```
1  min_k ← ∞
2  minSequences ← ∅
3  queue ← [(0, ε)]                                   // [(max_k, prefix), ...] sorted asc. by max_k
4  while queue ≠ ∅ ∧ queue[0][0] ≤ min_k do
5      (current_k, prefix) ← queue.pop()
6      if prefix incomplete then
7          forall remaining colours c_j not in prefix do
8              count ← no. components in G marked with prefix + [c_j]
9              queue.insort((max(current_k, count), prefix + [c_j]))
10     else                                           // current sequence is complete
11         if current_k = min_k then
12             minSequences ← minSequences ∪ {prefix}
13         if current_k < min_k then
14             min_k ← current_k
15             minSequences ← {prefix}
16 return min_k, minSequences
```

Priority Search*: Finding only the first minimal marking sequence

Algorithm 2: Priority Search*

Input : Graph $G = (V, E)$, valid colouring $c : V \to [\ell]$
Output : minimum k-locality of G, *one* marking sequence of that k-locality

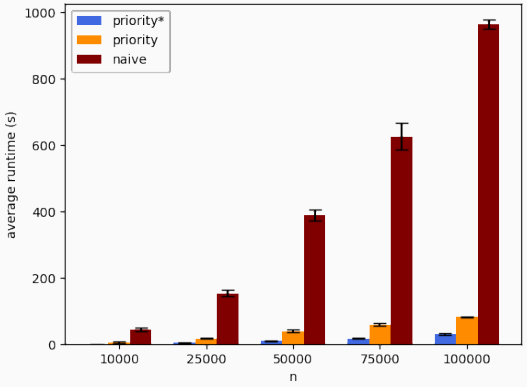
1 $\min_k \leftarrow \infty$
2 minSequences $\leftarrow \emptyset$
3 queue $\leftarrow [(0, \varepsilon)]$ // $[(\max_k, \text{prefix}), \ldots]$ sorted asc. by $\max_k$
4 **while** queue $\neq \emptyset \wedge$ queue[0] [0] $<$ $\min_k$ **do**
5 $(\text{current}_k, \text{prefix}) \leftarrow$ queue.pop()
6 **if** prefix incomplete **then**
7 **forall** remaining colours c_j not in prefix **do**
8 count $\leftarrow$ no. components in G marked with prefix $+ [c_j]$
9 queue.insort($(\max(\text{current}_k, \text{count}), \text{prefix} + [c_j])$)
10 **else** // current sequence is complete
11 **if** $\text{current}_k < \min_k$ **then**
12 $\min_k \leftarrow \text{current}_k$
13 minSequences $\leftarrow \{\text{prefix}\}$
14 **return** $\min_k$, minSequences

Testing Scalability: Barabási-Albert Random Graphs

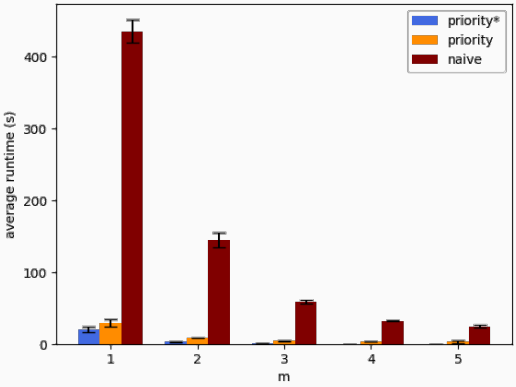
- It has two parameters n, m, where n is the number of nodes, and m is the number of edges each new node is added with.[3]
- We vary n between 10k and 100k, and fix $m = 10$ and $|C| = 5$ for the impact of the graph size.
- We vary $|C|$ between 2 and 8, and fix $n = 5k$ and $m = 10$.
- Node types/colours are distributed uniformly at random.

[3] **Statistical mechanics of complex networks.** R. Albert, A.-L. Barab´asi. In: Rev. Mod. Phys. 74 (1 2002), pp. 47–97

Scalability w.r.t. Graph Size



(a) Sensitivity to $|V|$



(b) Sensitivity to m

Average runtime on Barabási-Albert random graphs over 5 runs.

In case of emergency: Ask Pamela!