

Beating the Worst Case

Practical Course – 2nd Meeting

Max, Jean-Pierre

Some Questions

Implementation

Algorithms

Results

Some Questions

Implementation

- using the framework
- using further tools / languages
- workflow?

Algorithms

Results

Some Questions

Implementation

- using the framework
- using further tools / languages
- workflow?

Algorithms

- asymptotical running time?
- multiple variants?

Results

Some Questions

Implementation

- using the framework
- using further tools / languages
- workflow?

Algorithms

- asymptotical running time?
- multiple variants?

Results

- did you identify the mysterious networks?
- what did you learn about the algorithms' performance?
- any other learnings? *properties of the graphs, interesting observations, . . .*

Some Questions

Implementation

- using the framework
- using further tools / languages
- workflow?

Algorithms

- asymptotical running time?
- multiple variants?

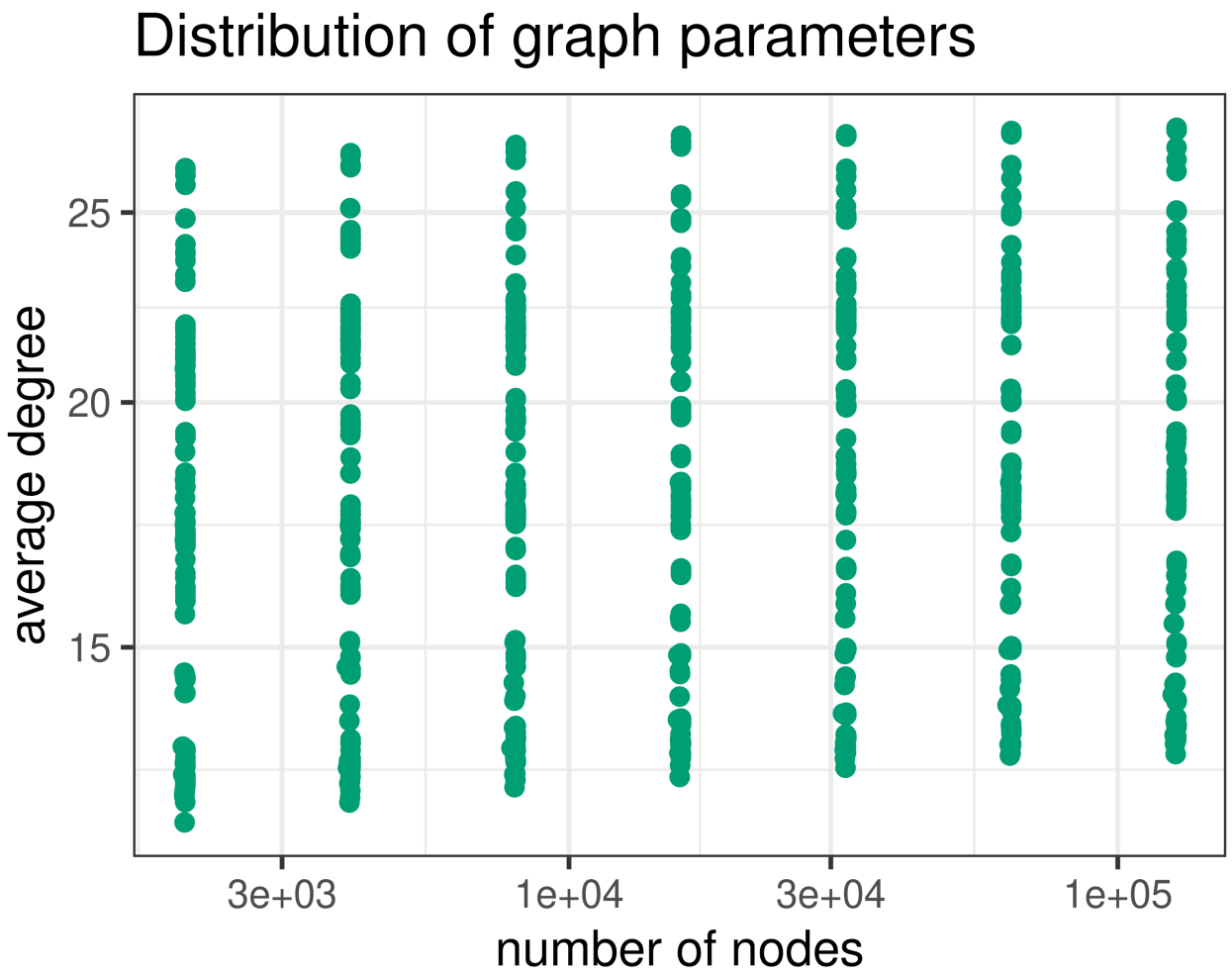
Anything else?

Results

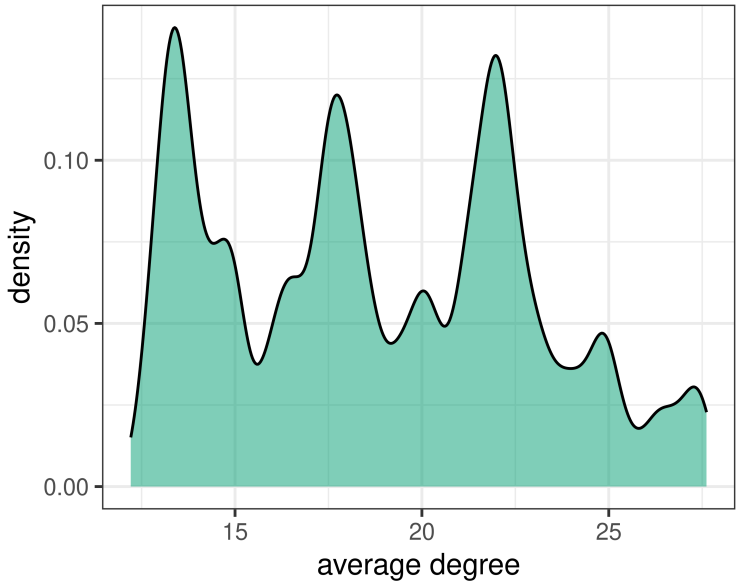
- did you identify the mysterious networks?
- what did you learn about the algorithms' performance?
- any other learnings? *properties of the graphs, interesting observations, . . .*

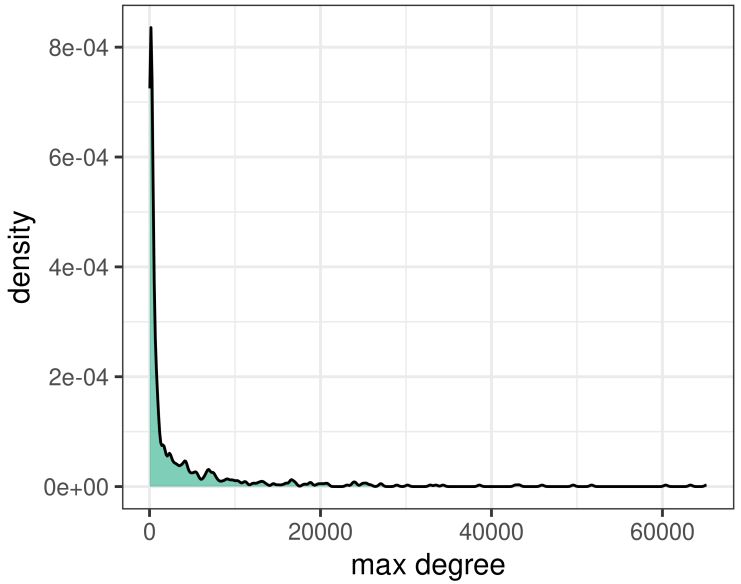
Properties of our graphs

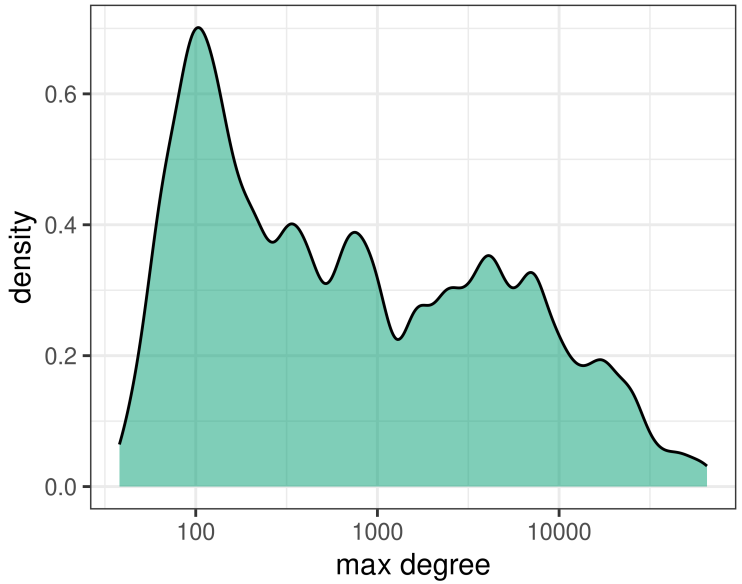
Properties of our graphs



Properties of our graphs

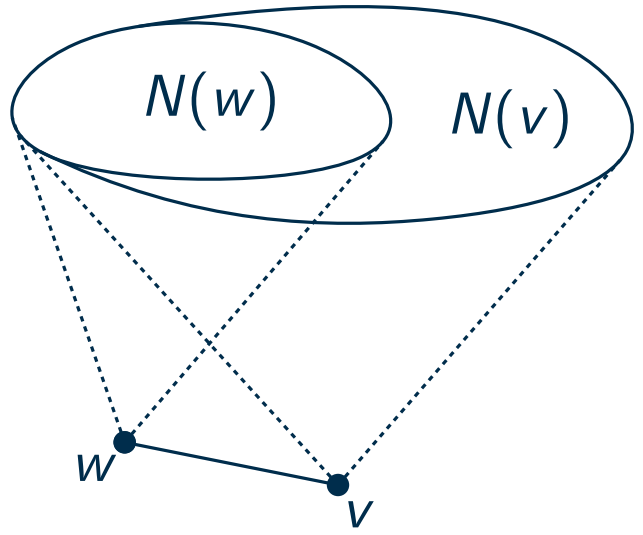






Computing the VC Dominance Rule

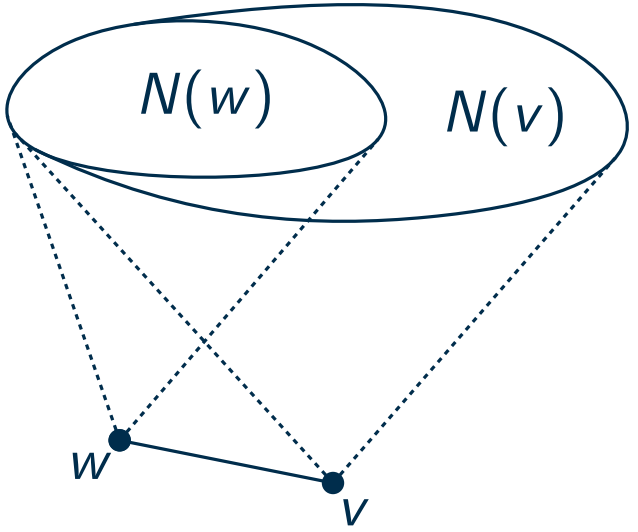
- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



Computing the VC Dominance Rule

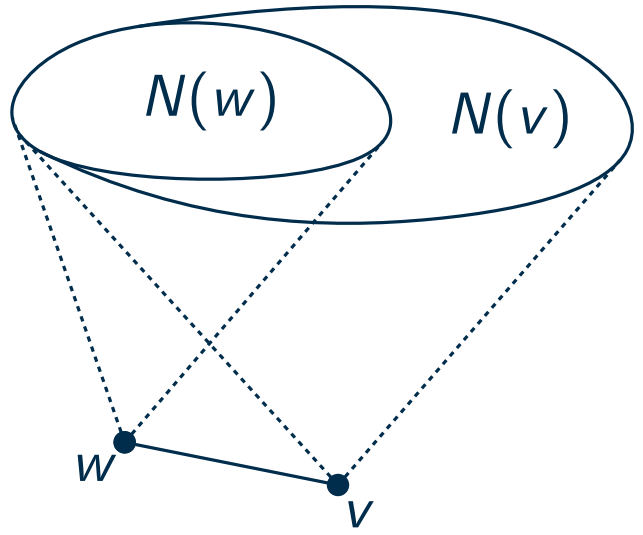
- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover

How to efficiently check the rule and compute the kernel?



Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover

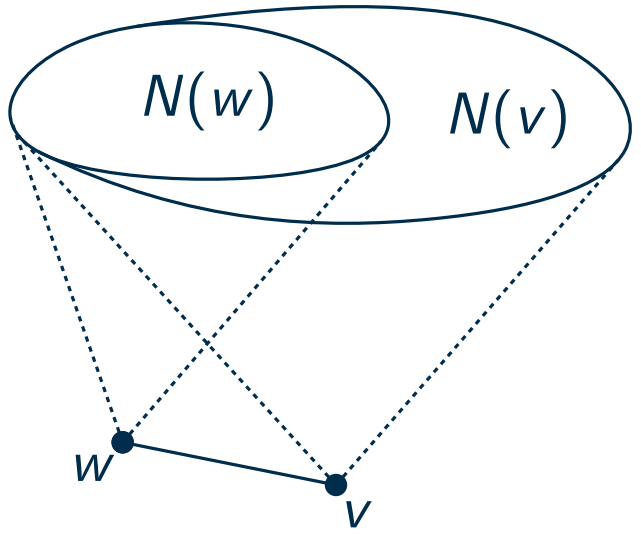


How to efficiently check the rule and compute the kernel?

Check all vertex pairs?

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover

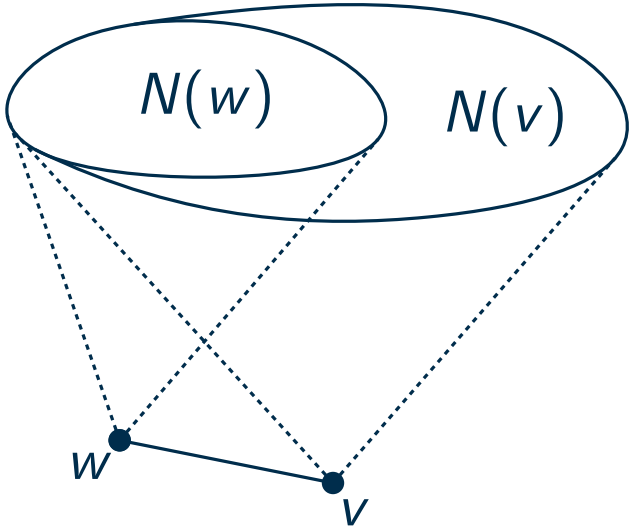


How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



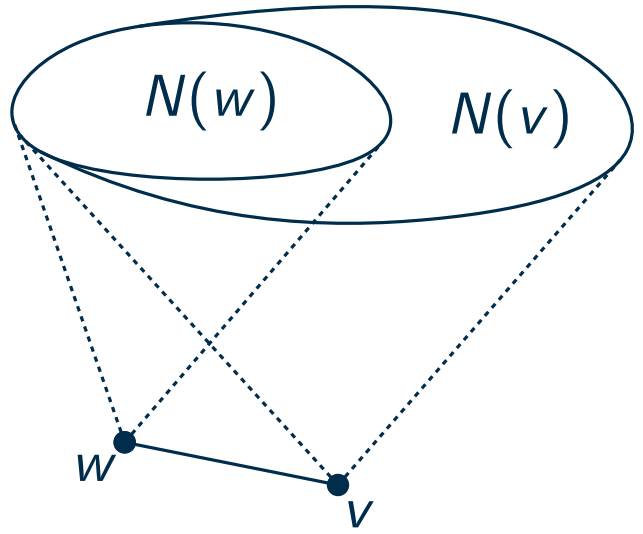
How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors!

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



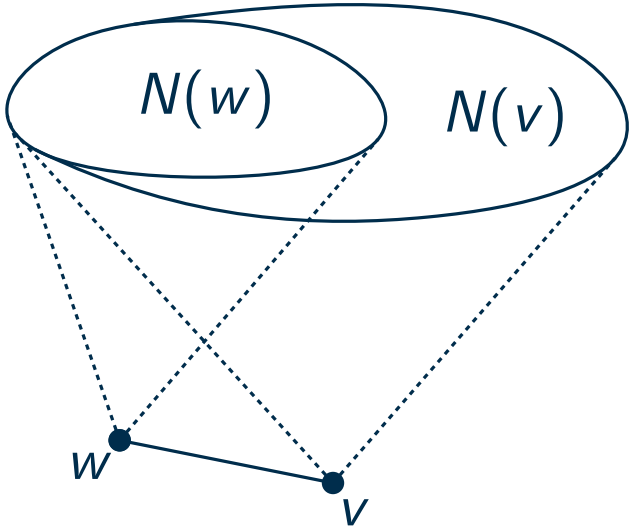
How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

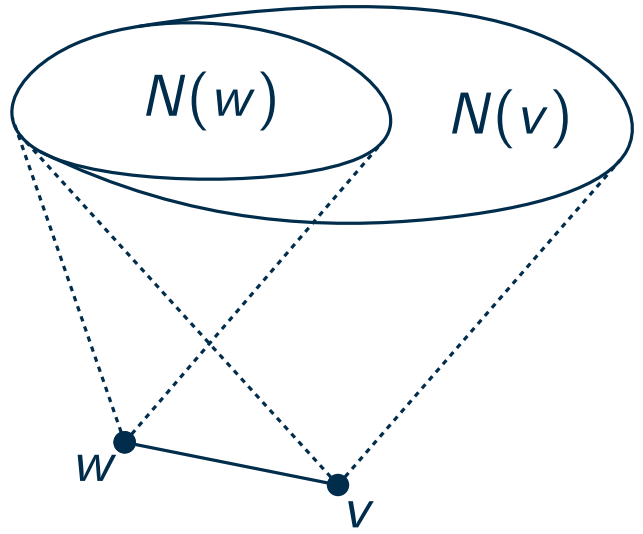
Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

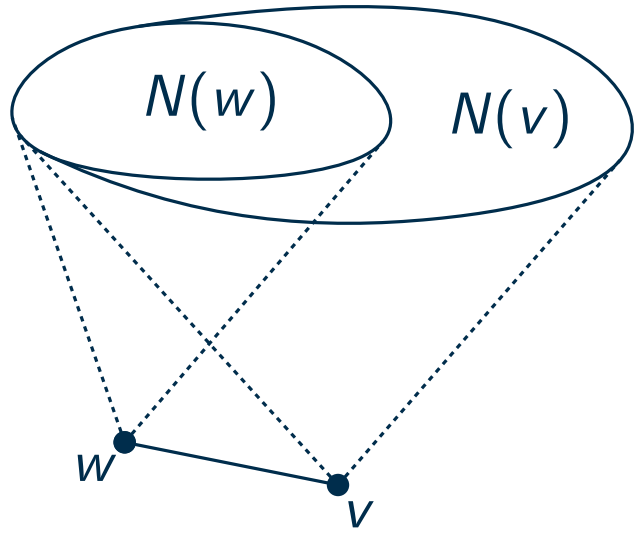
only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

- sorted lists
- hash table
- singelton size n array

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

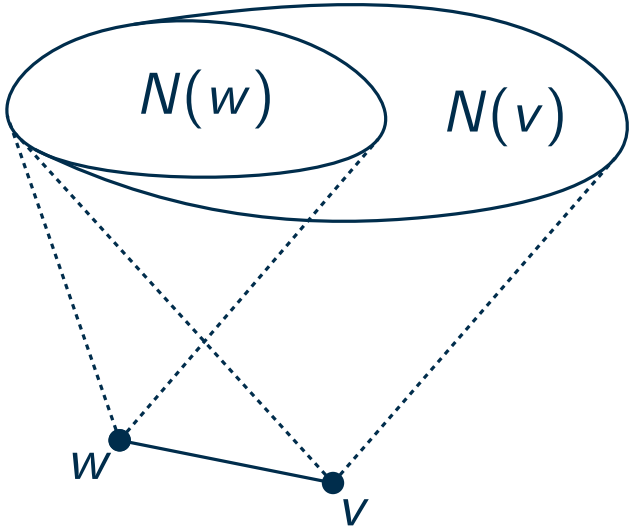
How to delete dominated vertex?

How to check $N[w] \subseteq N[v]$?

- sorted lists
- hash table
- singelton size n array

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

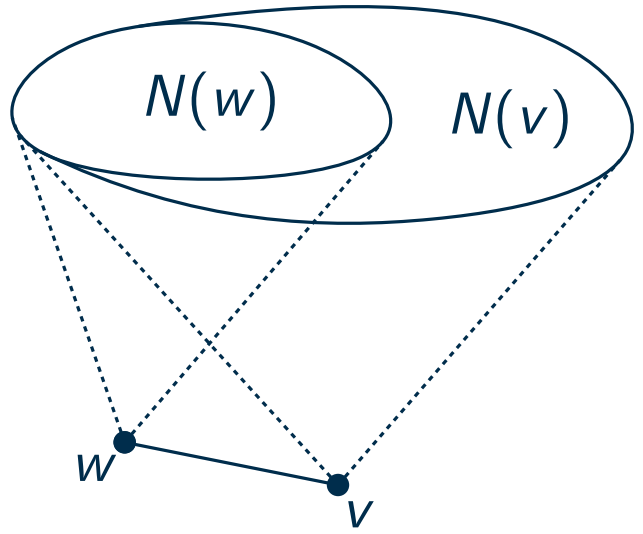
How to delete dominated vertex? only set flag!

How to check $N[w] \subseteq N[v]$?

- sorted lists
- hash table
- singelton size n array

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

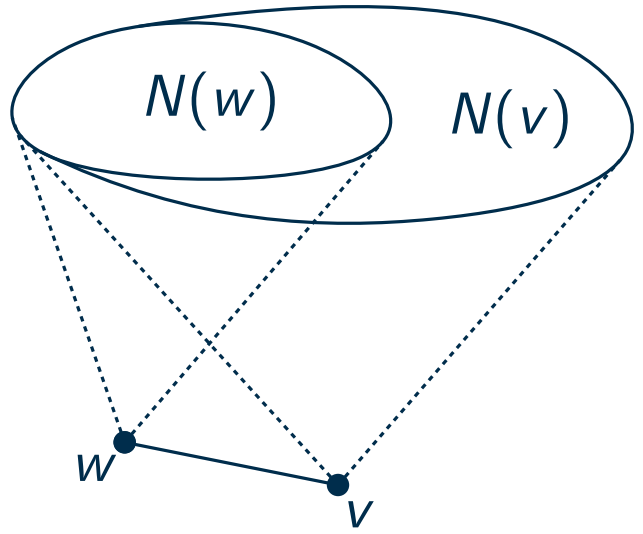
- sorted lists
- hash table
- singelton size n array

How to delete dominated vertex? only set flag!

Repeatedly delete single vertex?

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

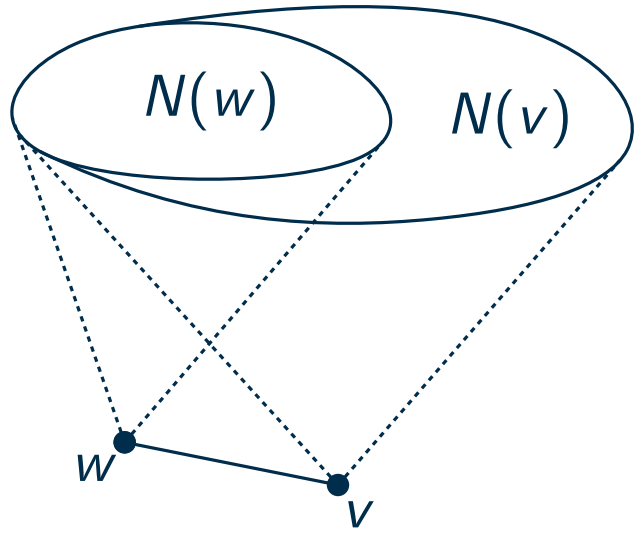
- sorted lists
- hash table
- singelton size n array

How to delete dominated vertex? only set flag!

Repeatedly delete single vertex? up to $O(n \cdot m)$ iterations

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

- sorted lists
- hash table
- singelton size n array

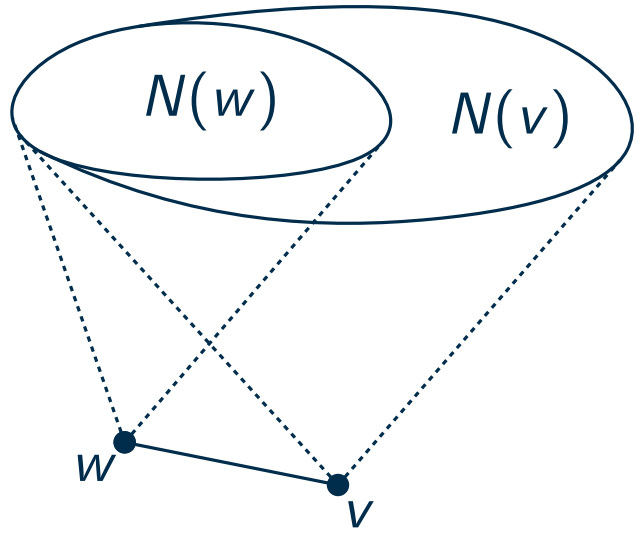
How to delete dominated vertex? only set flag!

Repeatedly delete single vertex? up to $O(n \cdot m)$ iterations

- better: delete all dominating vertices in one step!

Computing the VC Dominance Rule

- vertex v *dominates* its neighbor w if $N[w] \subseteq N[v]$
- *dominance rule:* a dominating vertex can be included in the vertex cover



How to efficiently check the rule and compute the kernel?

Check all vertex pairs? $O(n^2)$ pairs

only check neighbors! $O(m)$ pairs

How to check $N[w] \subseteq N[v]$?

- sorted lists
- hash table
- singelton size n array

How to delete dominated vertex? only set flag!

Repeatedly delete single vertex? up to $O(n \cdot m)$ iterations

- better: delete all dominating vertices in one step!
- show: if a and b are dominating in G, then b is dominating in $G - a$

Computing the VC Dominance Rule

Algorithm 1: Vertex Cover Reduction

```
while ∃u ∈ V dominating do
    for {a, b} ∈ E do
        if N[a] ⊆ N[b] then
            mark b as dominating
        end
    end
    mark all dominating nodes as deleted
end
mark all isolated vertices as deleted
```

Computing the VC Dominance Rule

Algorithm 1: Vertex Cover Reduction

```
while ∃u ∈ V dominating do
    for {a, b} ∈ E do
        if N[a] ⊆ N[b] then
            mark b as dominating
        end
    end
    mark all dominating nodes as deleted
end
mark all isolated vertices as deleted
```

ignore deleted a, b!

sorted lists, hash table or size n array

only flag, no actual deletion!

Bi-directional BFS

Question: what is the asymptotic running time of bi-BFS?

Bi-directional BFS

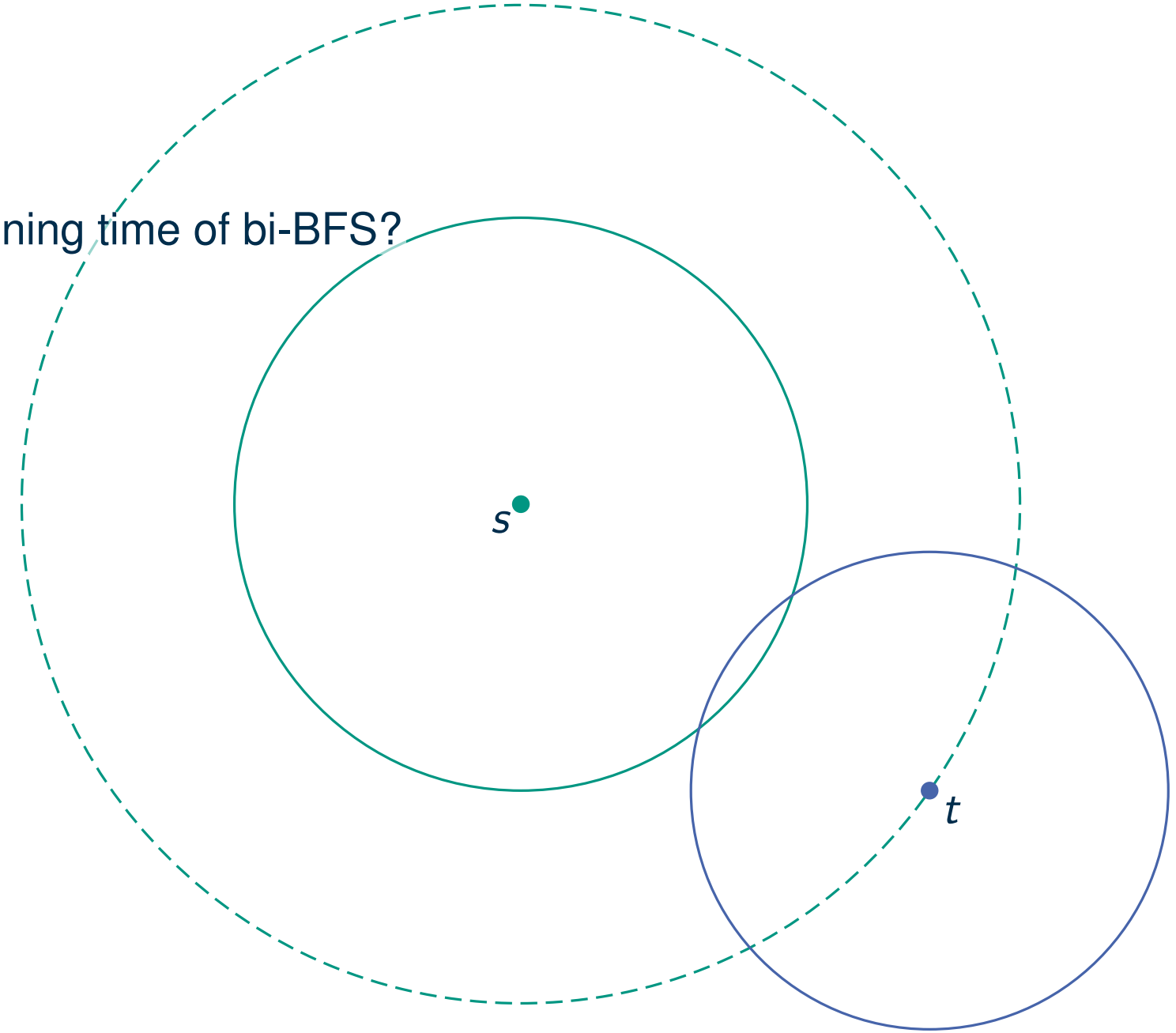
Question: what is the asymptotic running time of bi-BFS?

. . . on street network?

Bi-directional BFS

Question: what is the asymptotic running time of bi-BFS?

. . . on street network?

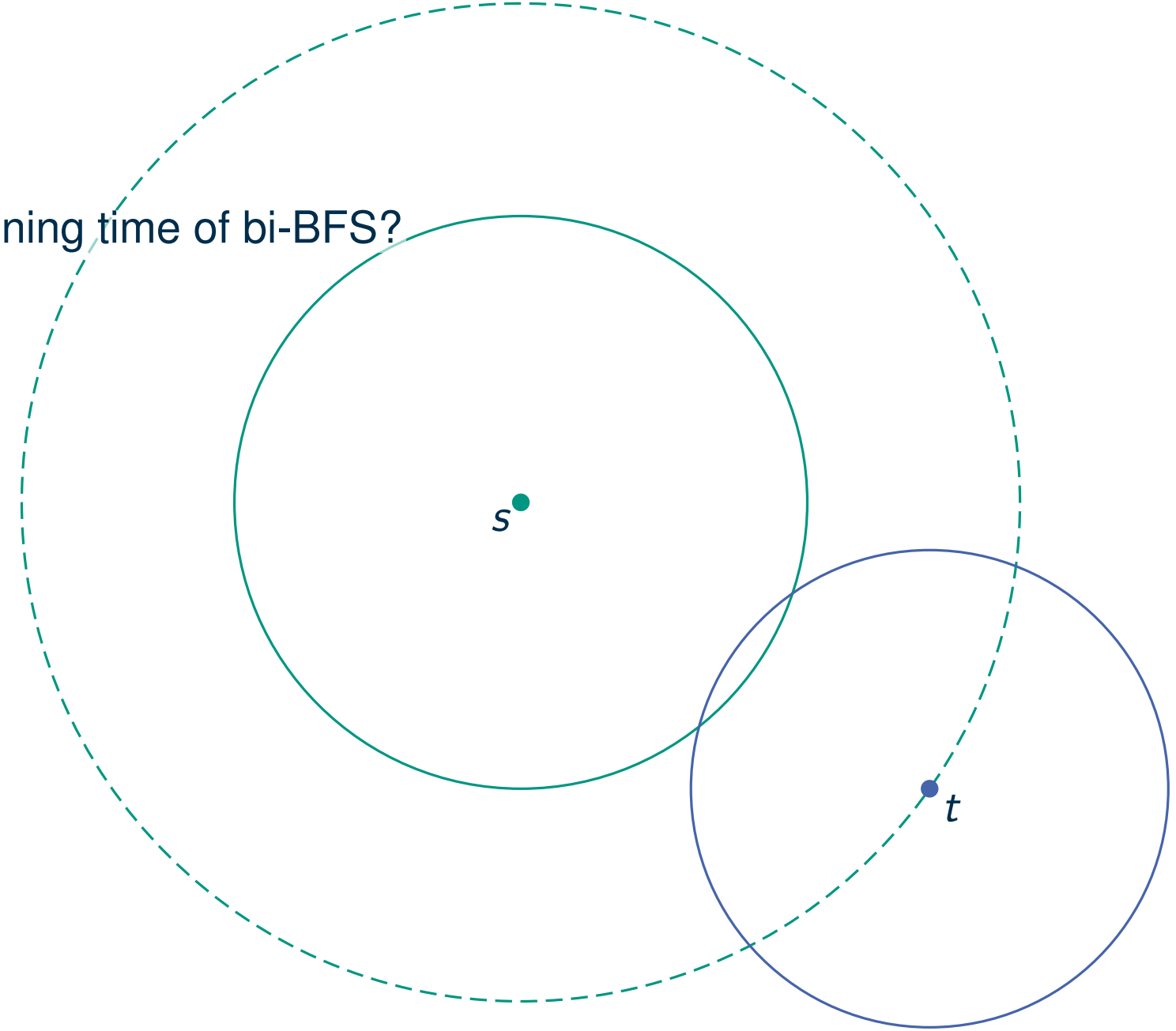


Bi-directional BFS

Question: what is the asymptotic running time of bi-BFS?

. . . on street network?

. . . on social network?



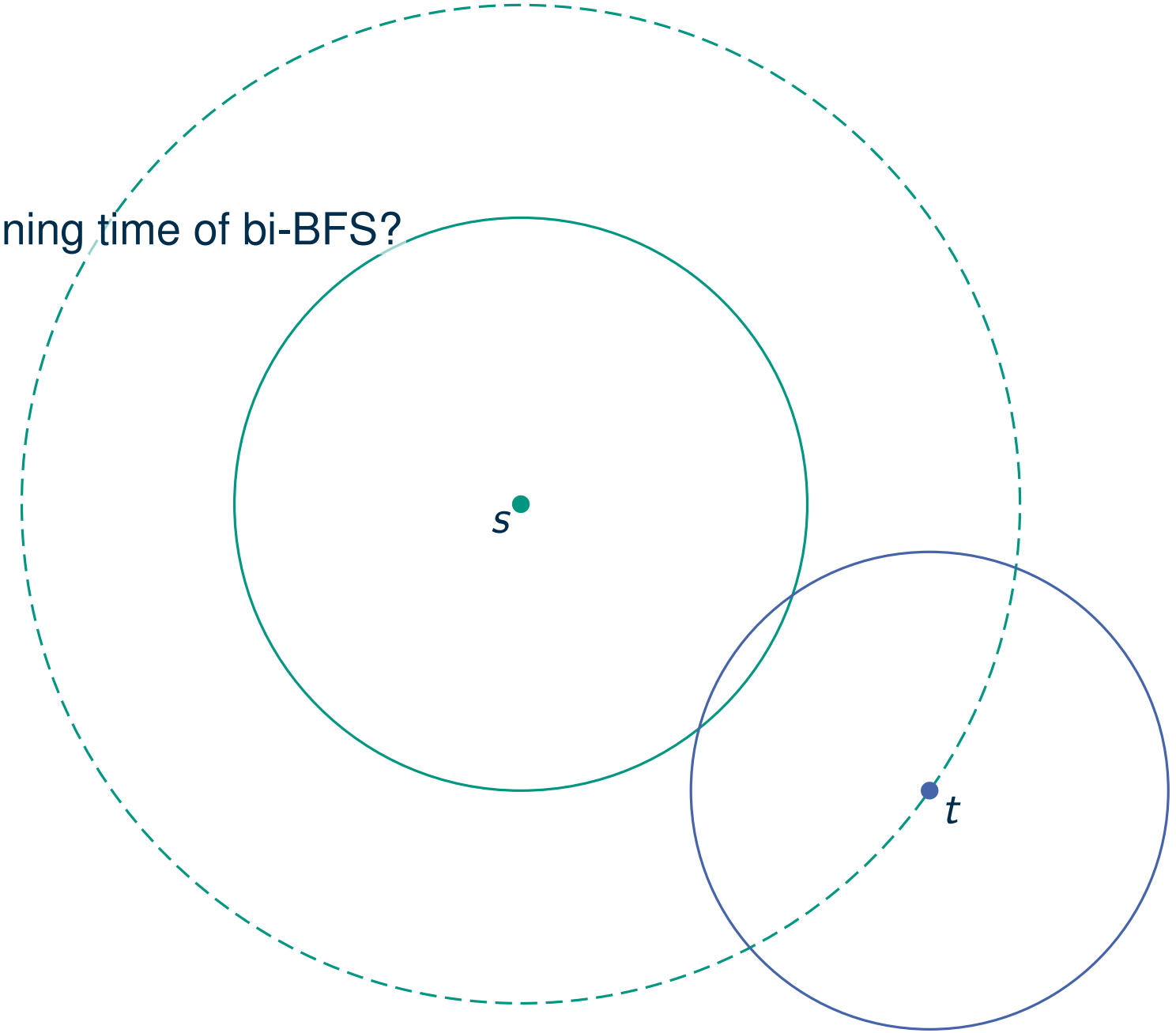
Bi-directional BFS

Question: what is the asymptotic running time of bi-BFS?

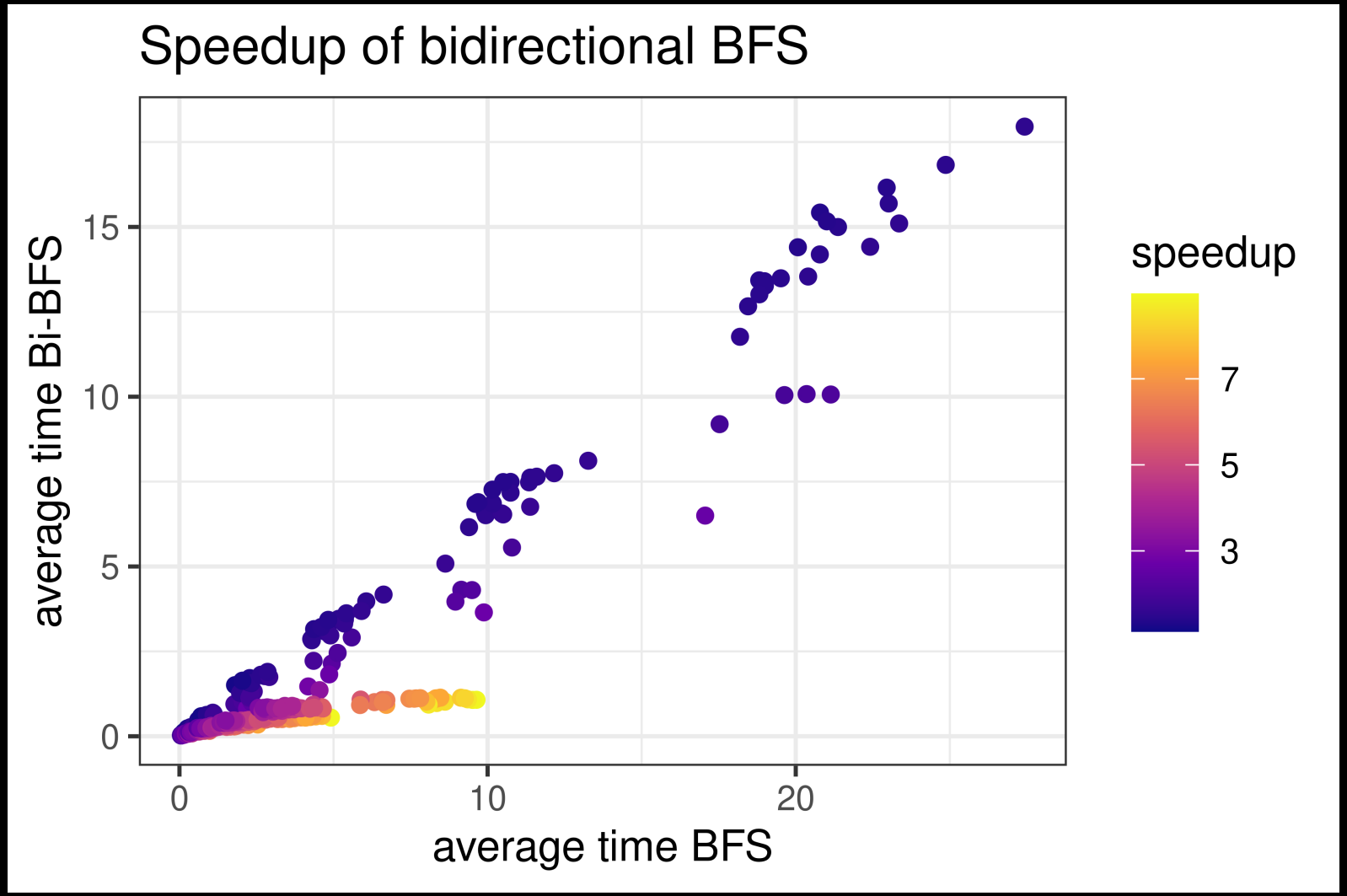
. . . on street network?

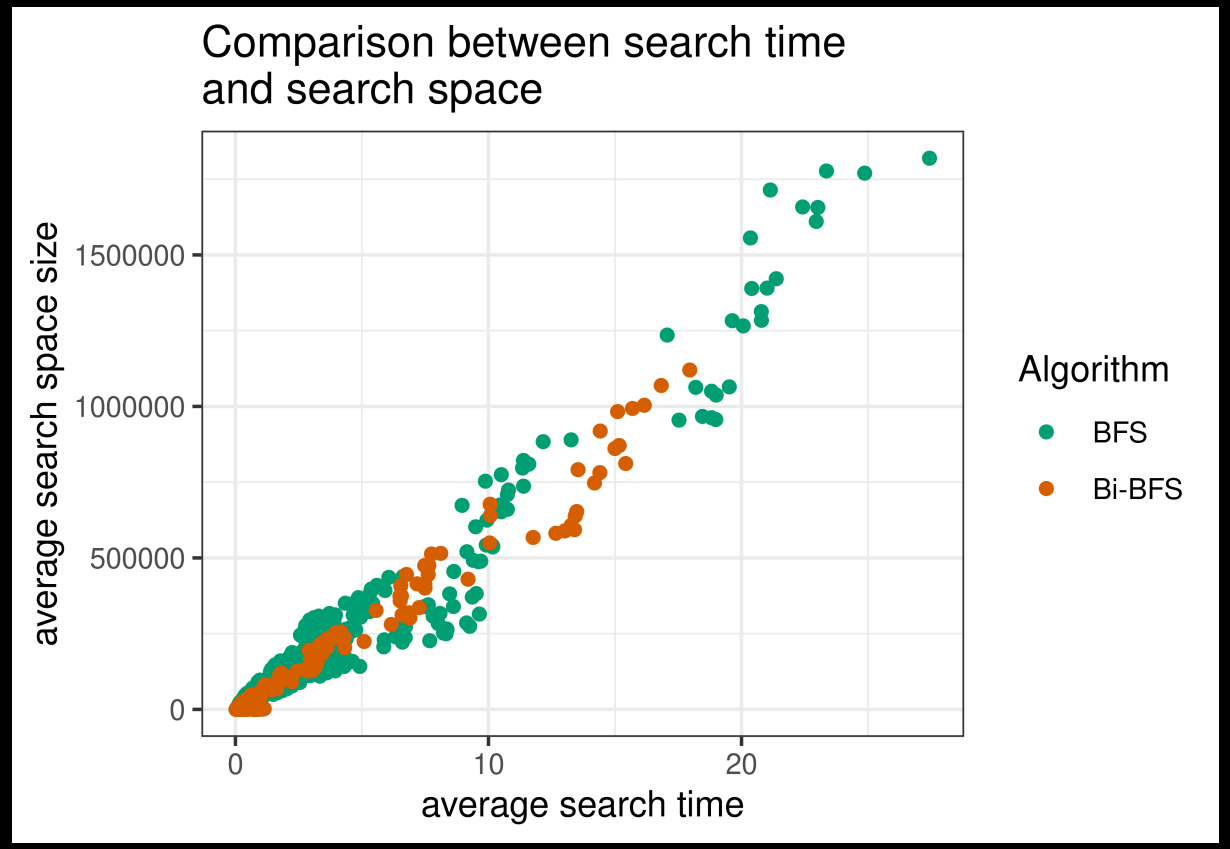
. . . on social network?

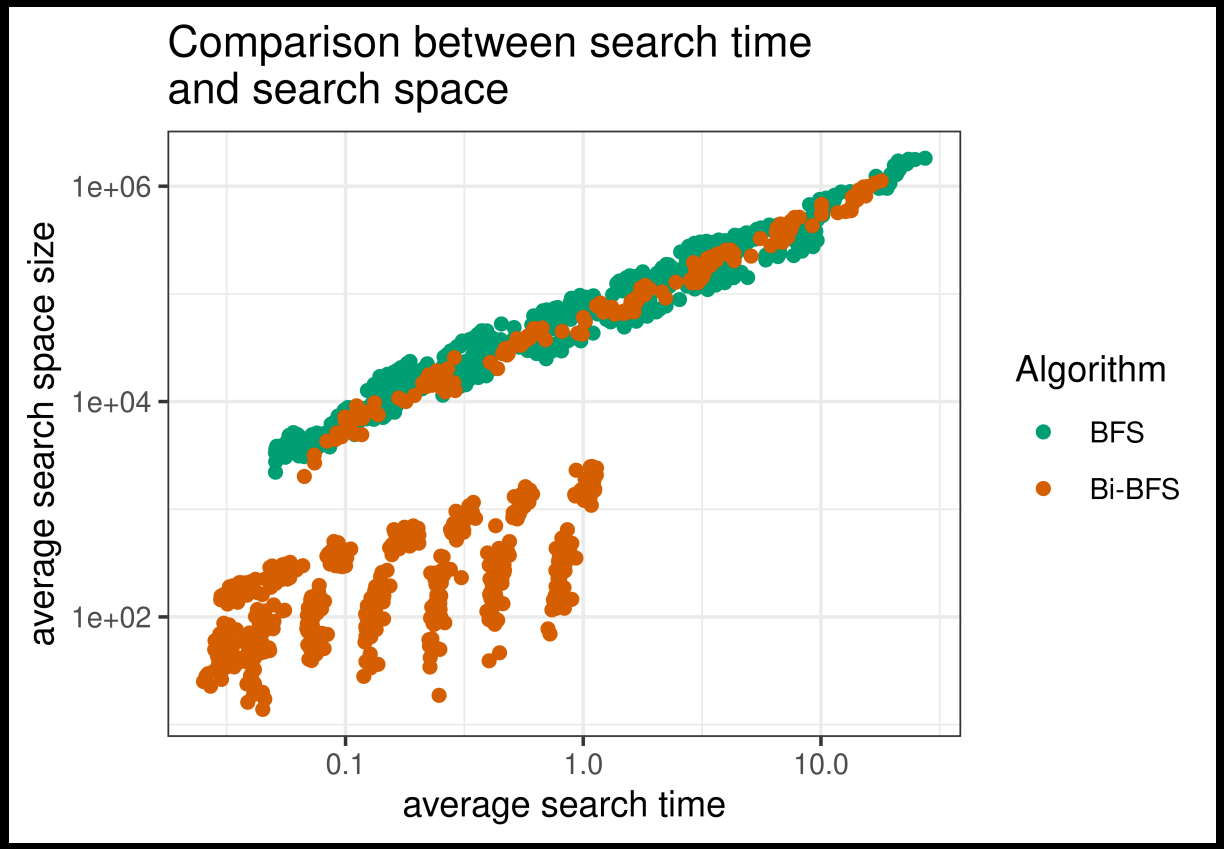
. . . on random network?

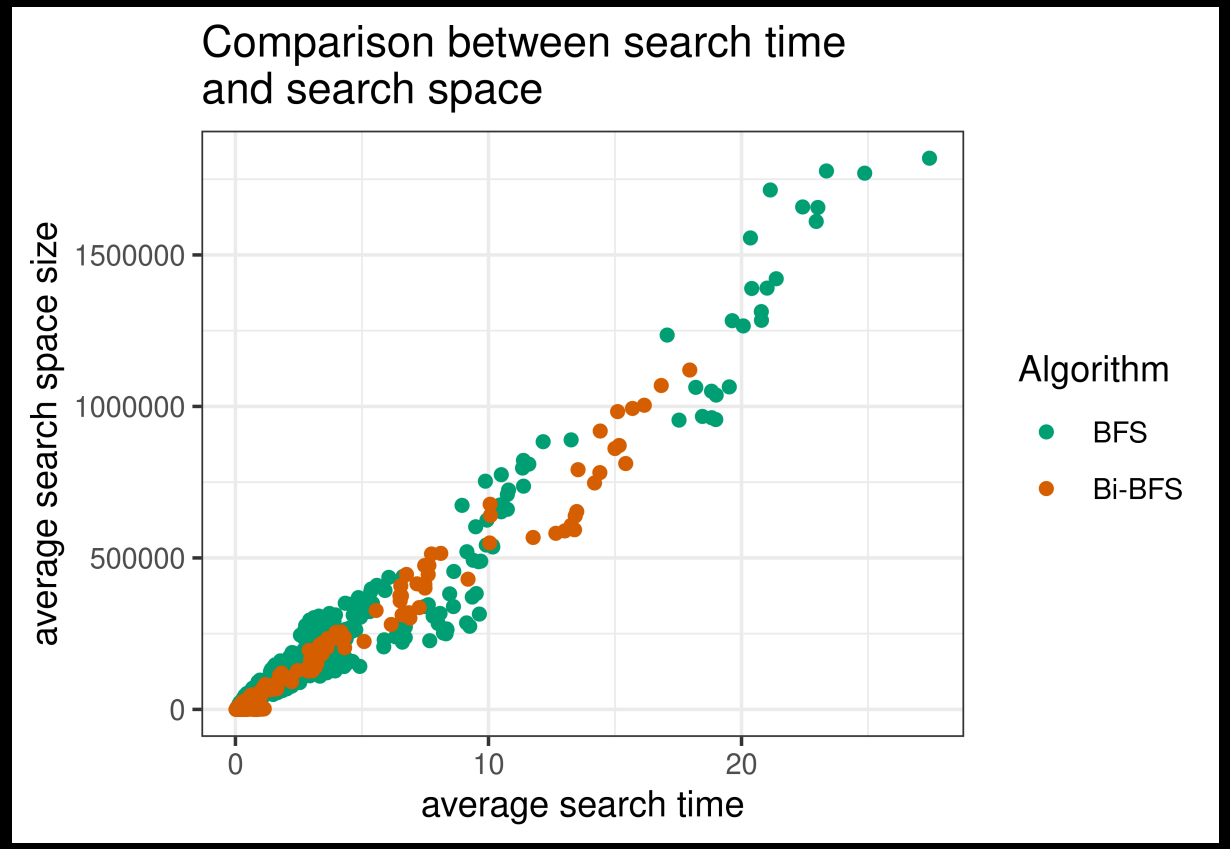


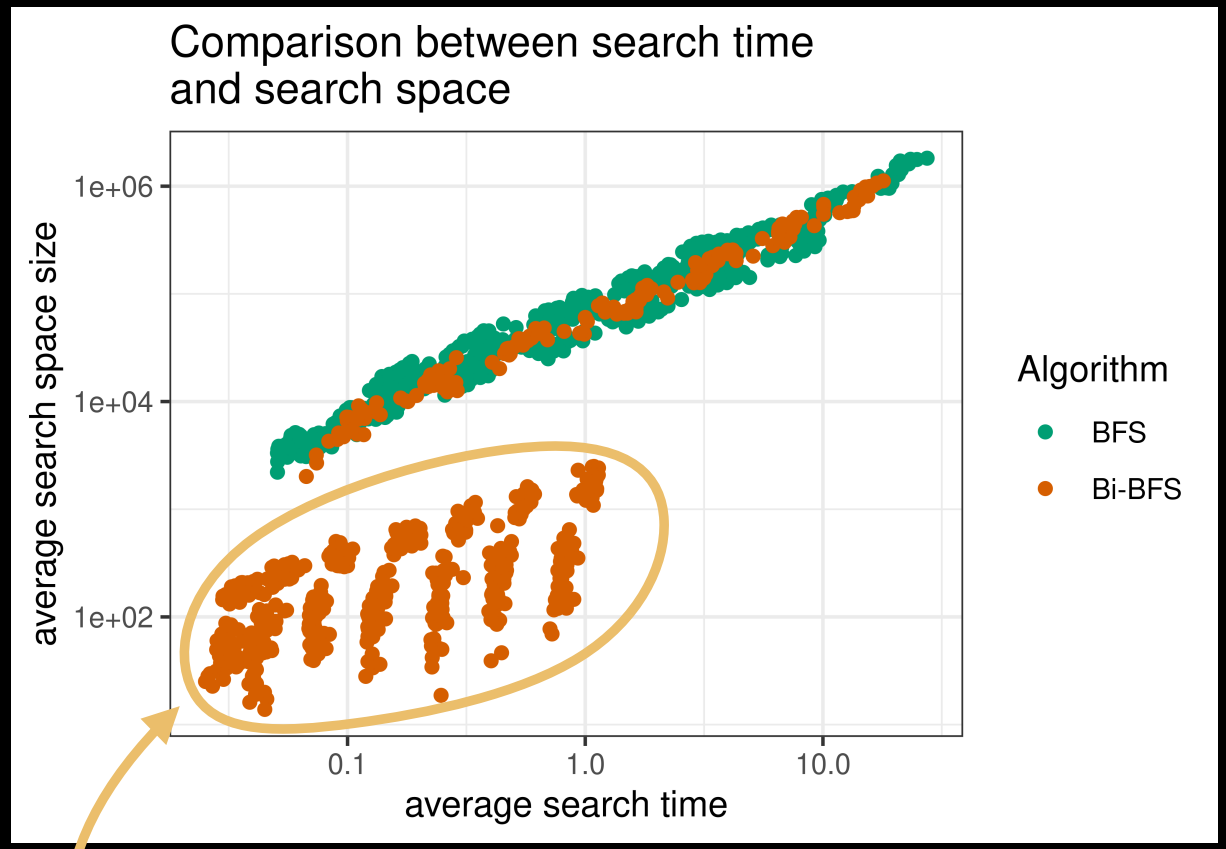
Demo


Speedup of bidirectional BFS
average time Bi-BFS
15
10
5
0
0
10
20
average time BFS
speedup
7
5
3

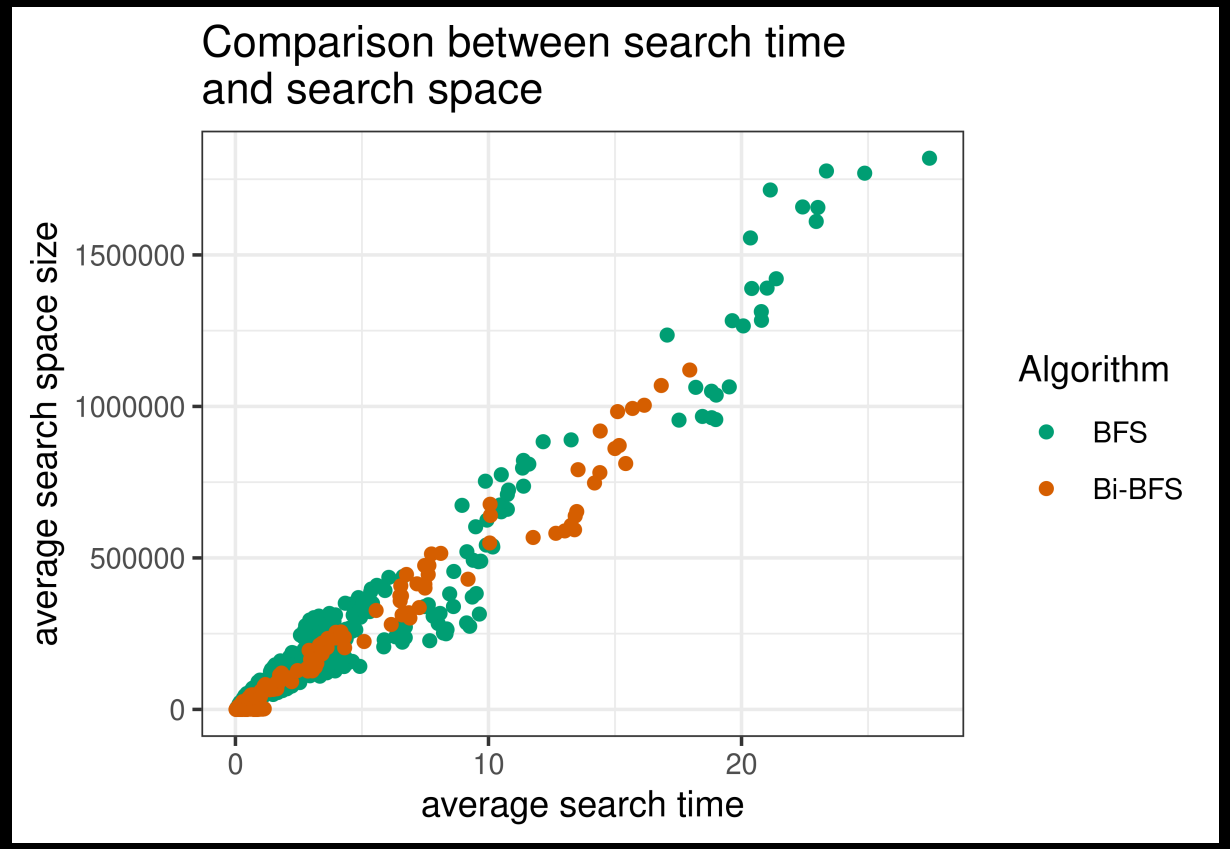

Comparison between search time
and search space
average search space size
1500000
1000000
500000
0
0
10
20
average search time
Algorithm
BFS
Bi-BFS

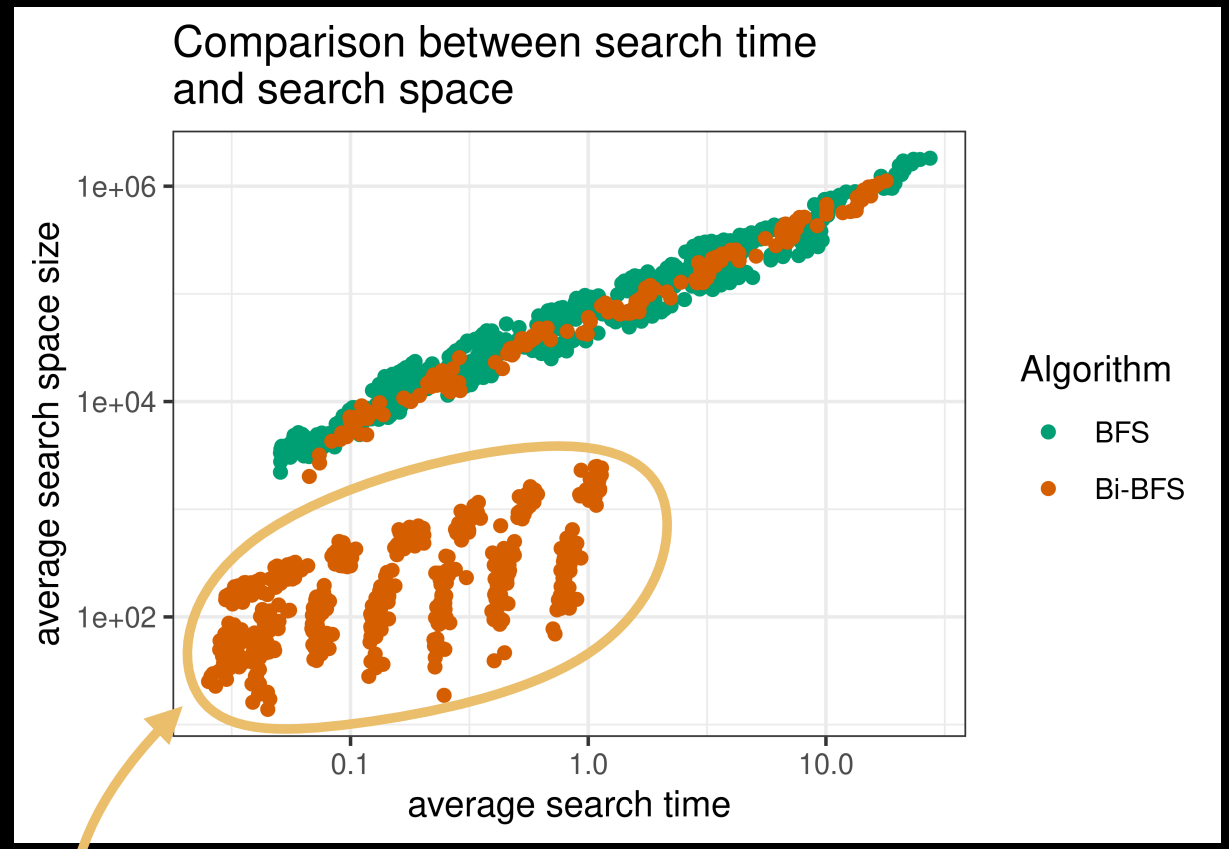

Comparison between search time
and search space
average search space size
1e+06
1e+04
1e+02
0.1
1.0
10.0
average search time
Algorithm
BFS
Bi-BFS





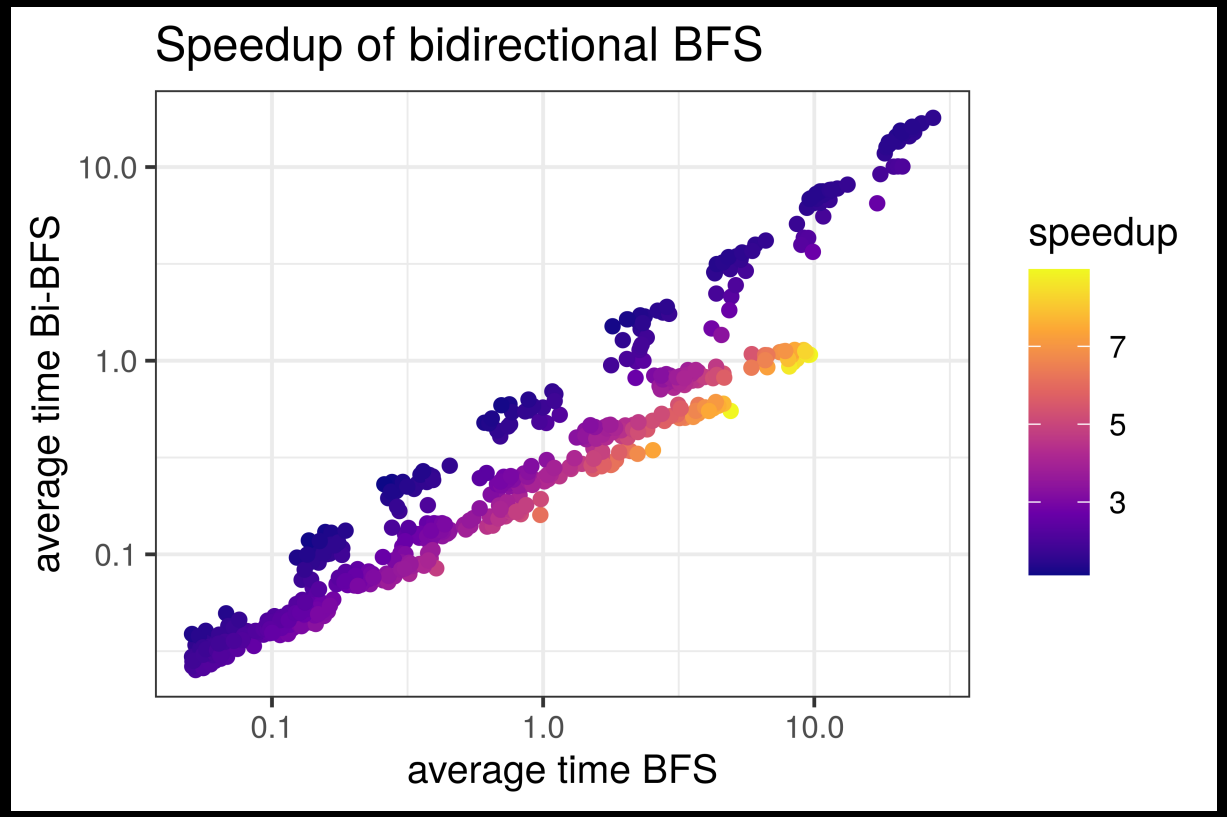
something is off

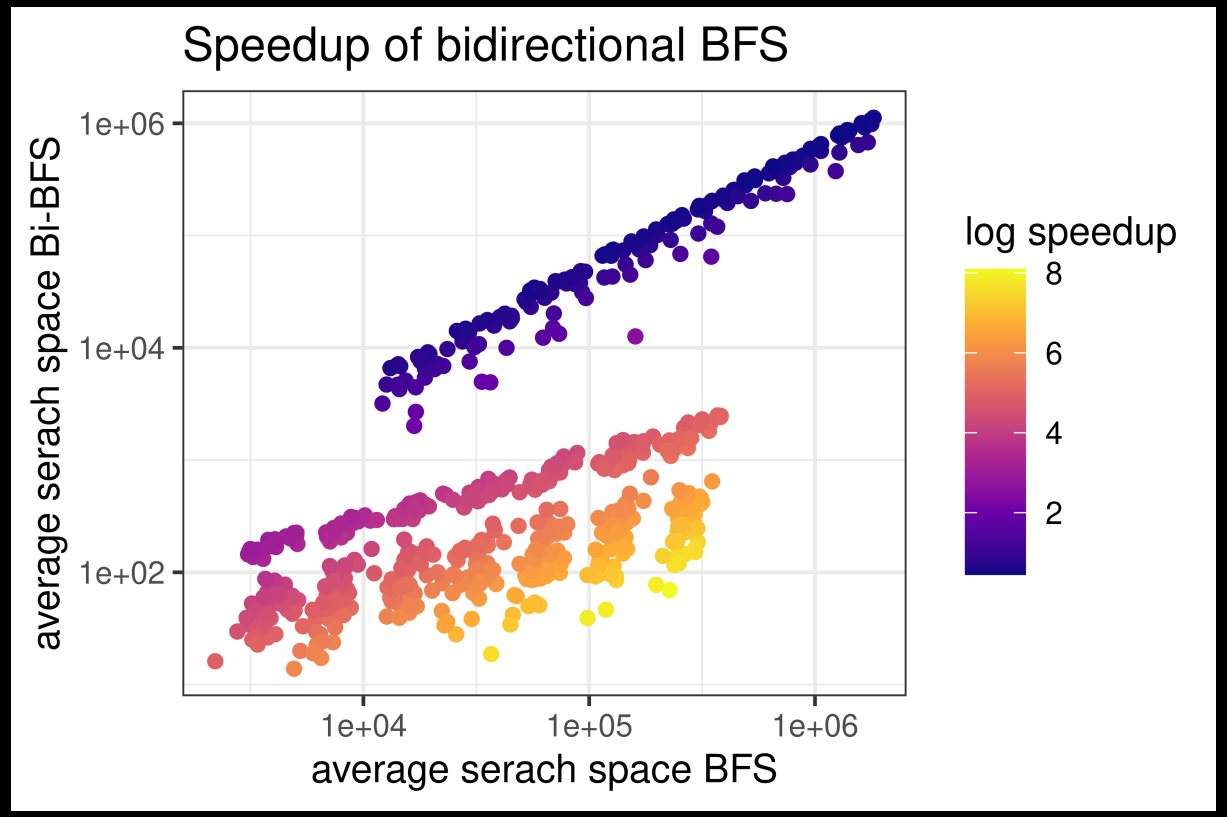


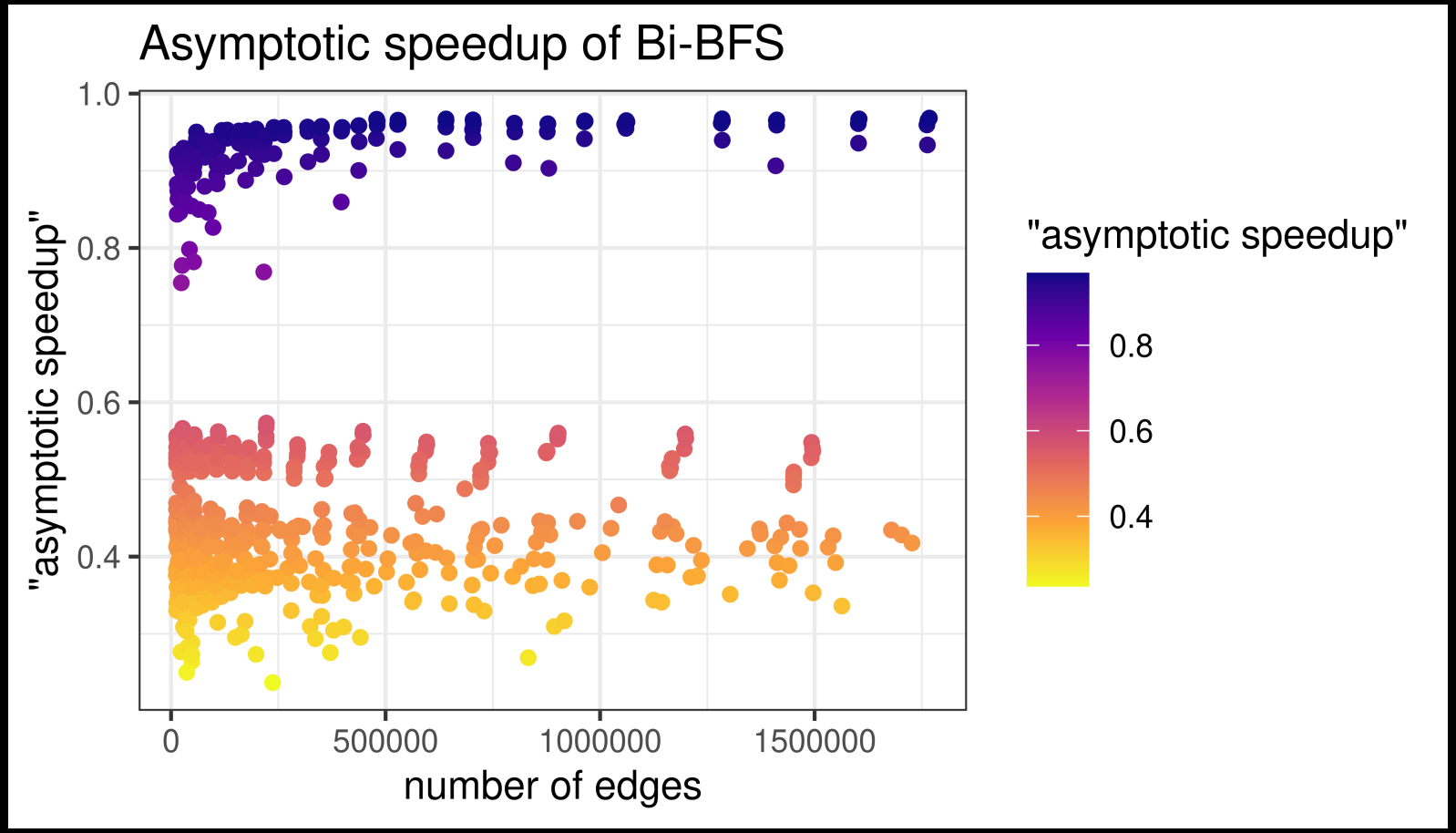


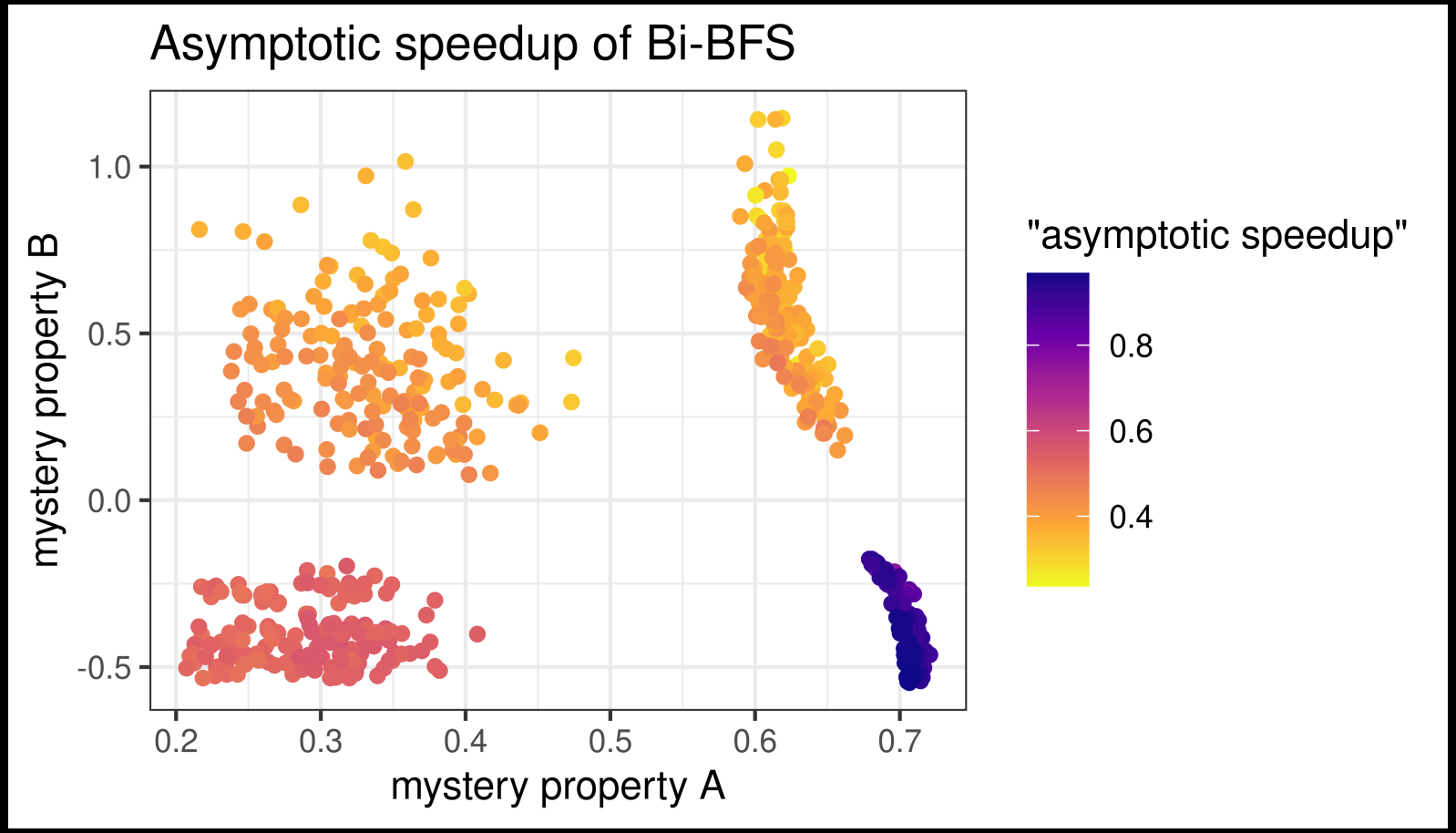
something is off

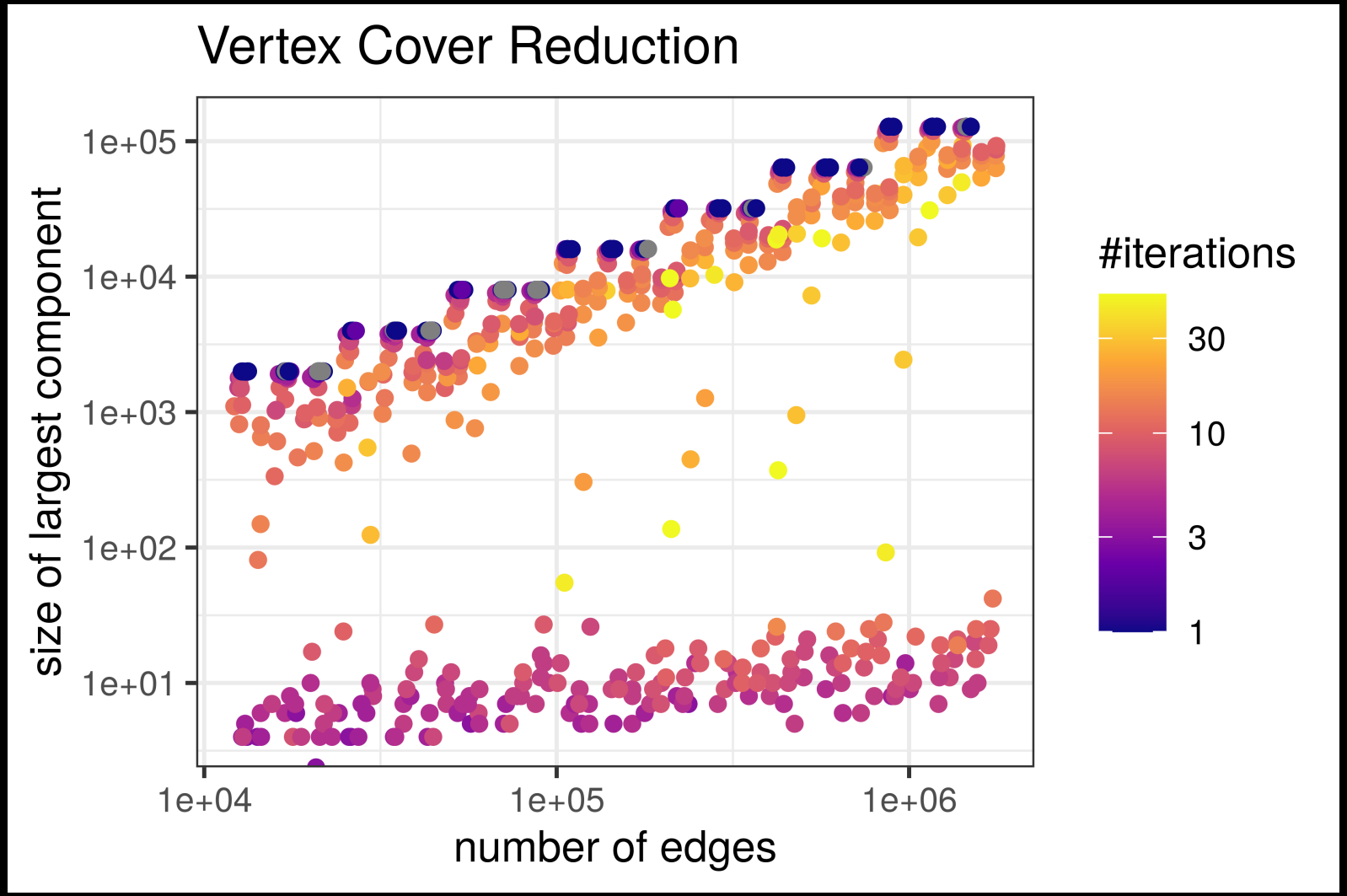
our Bi-BFS allocates an array of size $O(n)$


Speedup of bidirectional BFS
average time Bi-BFS
10.0
1.0
0.1
0.1
1.0
10.0
average time BFS
speedup
7
5
3


Speedup of bidirectional BFS
average serach space Bi-BFS
1e+06
1e+04
1e+02
1e+04
1e+05
1e+06
average serach space BFS
log speedup
8
6
4
2


Asymptotic speedup of Bi-BFS
"asymptotic speedup"
1.0
0.8
0.6
0.4
0
500000
1000000
1500000
number of edges
"asymptotic speedup"
0.8
0.6
0.4


Asymptotic speedup of Bi-BFS
mystery property B
1.0
0.5
0.0
-0.5
0.2
0.3
0.4
0.5
0.6
0.7
mystery property A
"asymptotic speedup"
0.8
0.6
0.4


Vertex Cover Reduction
size of largest component
1e+05
1e+04
1e+03
1e+02
1e+01
1e+04
1e+05
1e+06
number of edges
#iterations
30
10
3
1

Tasks until next week

- finish exercise sheet
- submit your guesses for the correct graphs via e-mail
- prepare short presentation (5 min.)
- bonus question: can you recognise networks on which the VC-kernel is small / bi-BFS fast?
  without executing the algorithm on them