Distributed Derandomization via Network Decomposition

Václav Rozhoň (ETH)

joint work with
Mohsen Ghaffari (ETH), Christoph Grunau (ETH)

We will see...

...distributed deterministic algorithm for network decomposition.

This is a key technical tool for a lot of theory of the **LOCAL** model built in past few years.

Teaser:

“For locally checkable problems, **P-RLOCAL = P-LOCAL**, i.e., **all** randomized distributed algorithms can be efficiently derandomized. ”

“$\Delta+1$ coloring problem (and other problems) can now be solved in poly(log log n) rounds by a randomized algorithm. “

Plan

1. See why is network decomposition a very handy technique.
2. Formulate general framework for turning sequential algorithms into distributed ones. See how it relates to derandomization.
3. See a simple deterministic algorithm for network decomposition.
4. See what this tells us about deterministic **CONGEST** algorithms and randomized **LOCAL**/**MPC** algorithms.

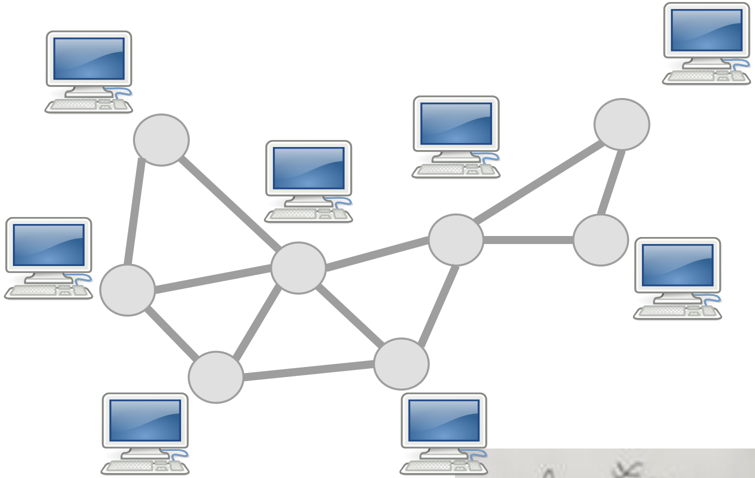
The **LOCAL** model of distributed graph algorithms

- Undirected graph $G=(V,E)$ with n nodes
- One computer in each node

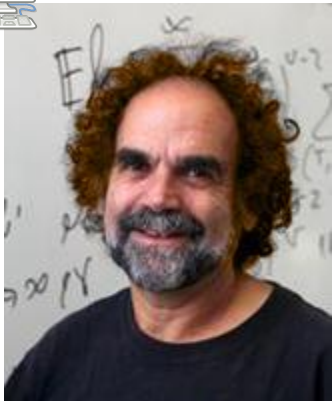
- Synchronous message passing rounds
- Unbounded message size and $\boxed{\text{computation!}}$ (more honest version: **CONGEST** model)

- Initially, nodes know only (upper bound on) n and their $\boxed{\text{unique } O(\log n) \text{ bit label}}$
- In the end, each node should know its part of output

- Time complexity: number of rounds



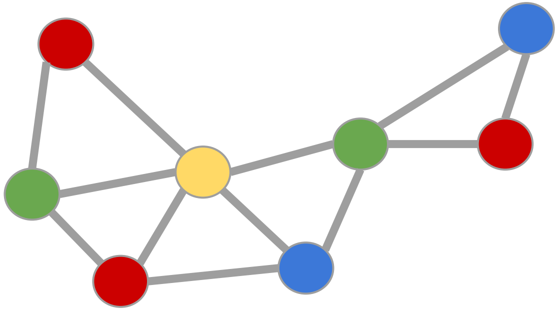
LOCAL model
[Linial FOCS'87]



Network decomposition: why we like it

Our running example: Δ+1 coloring

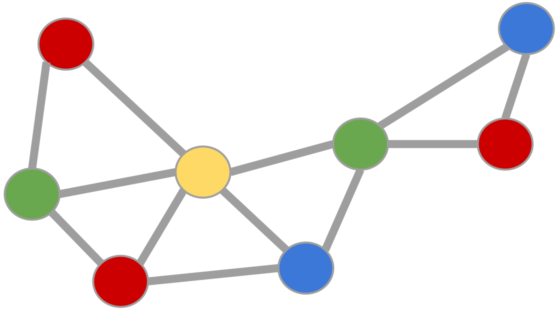
- greedy sequential algorithm
- efficient randomized solution



Our running example: Δ+1 coloring

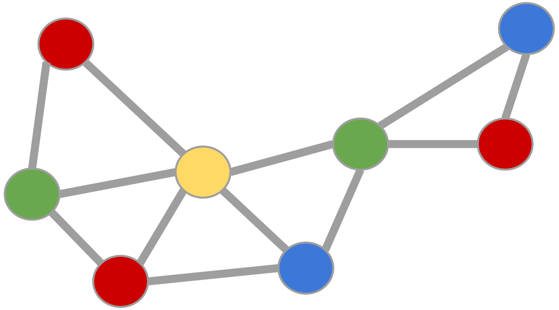
- greedy sequential algorithm
- efficient randomized solution

Q: Is there a principled approach to the problem that would also work for maximal independent set, matching, …?



Our running example: Δ+1 coloring

- greedy sequential algorithm
- efficient randomized solution



Q: Is there a principled approach to the problem that would also work for maximal independent set, matching, …?

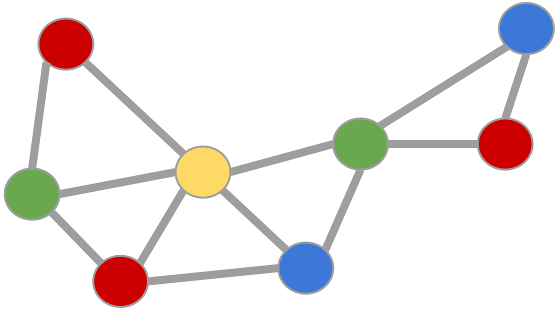
Q: Is there an efficient deterministic algorithm?

Our running example: Δ+1 coloring

- greedy sequential algorithm
- efficient randomized solution

Q: Is there a principled approach to the problem that would also work for maximal independent set, matching, …?

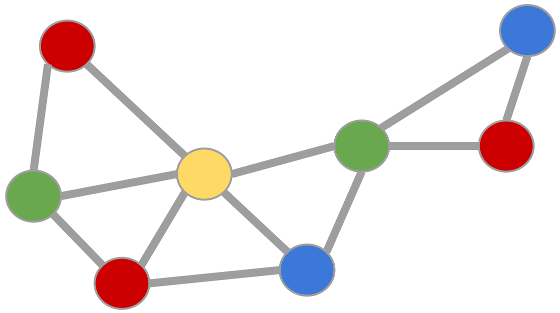
Q: Is there an efficient deterministic algorithm?



Teaser: we will see that the answer is YES for both questions.

Our running example: Δ+1 coloring

- greedy sequential algorithm
- efficient randomized solution



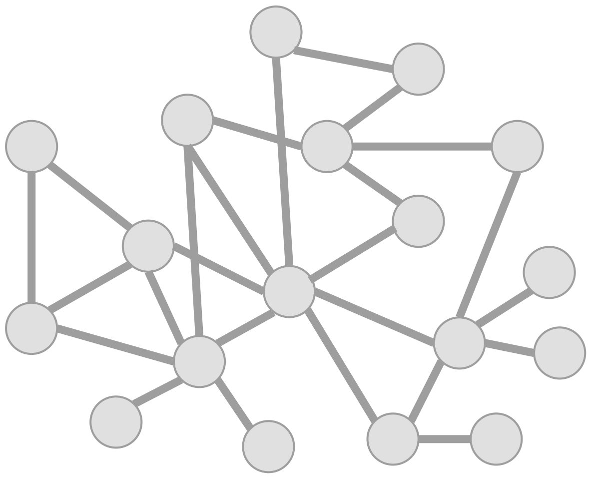
Q: Is there a principled approach to the problem that would also work for maximal independent set, matching, …?

Q: Is there an efficient deterministic algorithm?

Teaser: we will see that the answer is YES for both questions.

Principled approach

Easy case for our quest: the underlying graph has poly(log n) diameter.

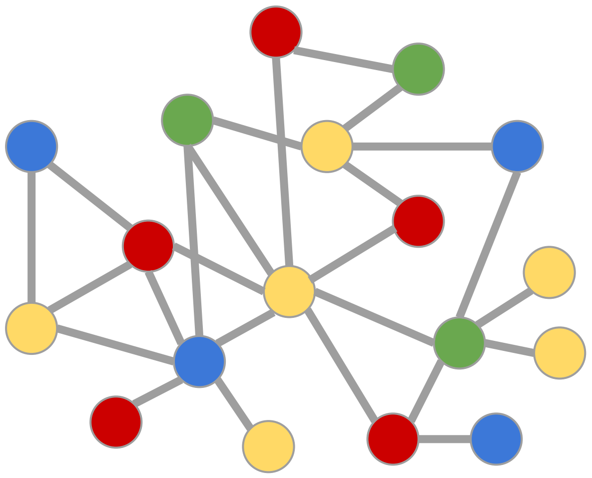


Solution: bruteforce

Principled approach

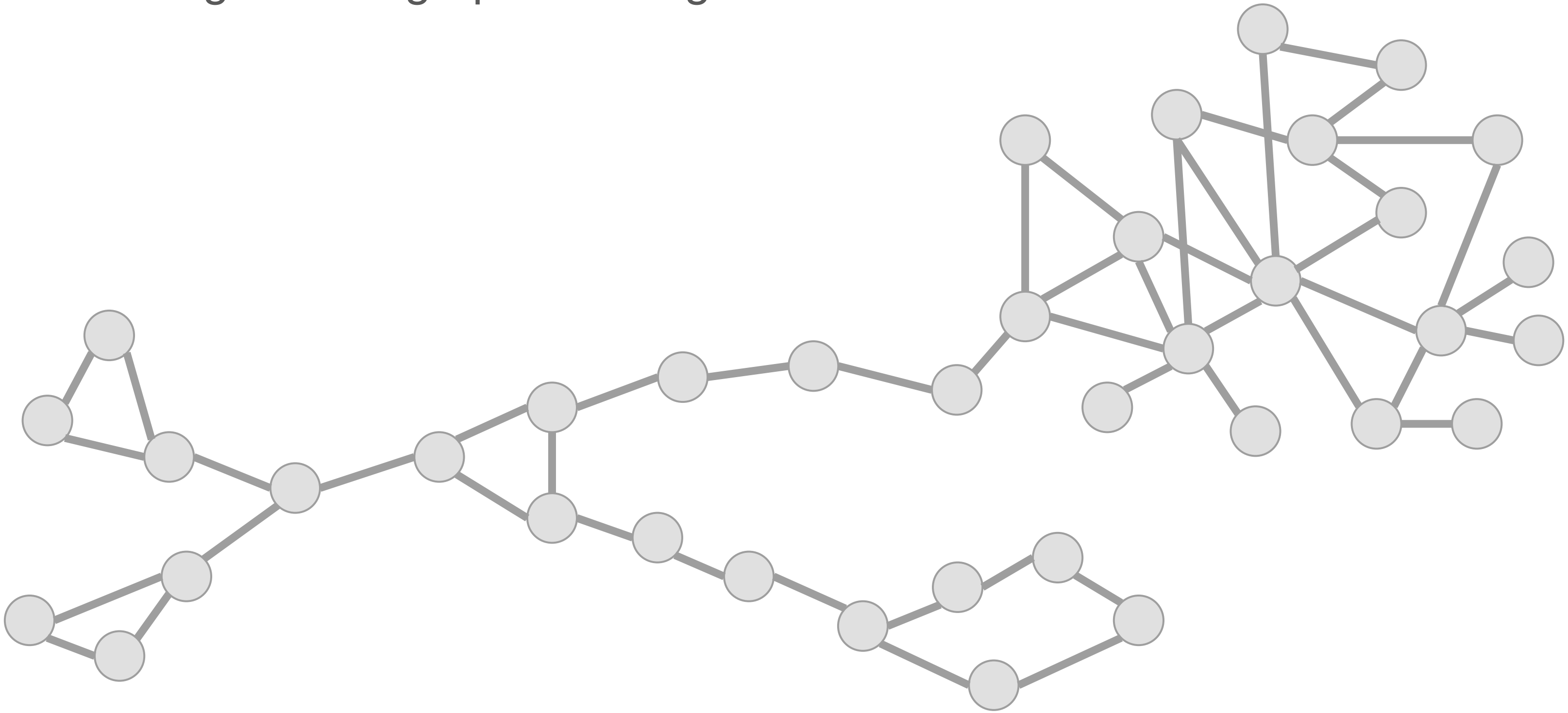
Easy case for our quest: the underlying graph has poly(log n) diameter.

Solution: bruteforce



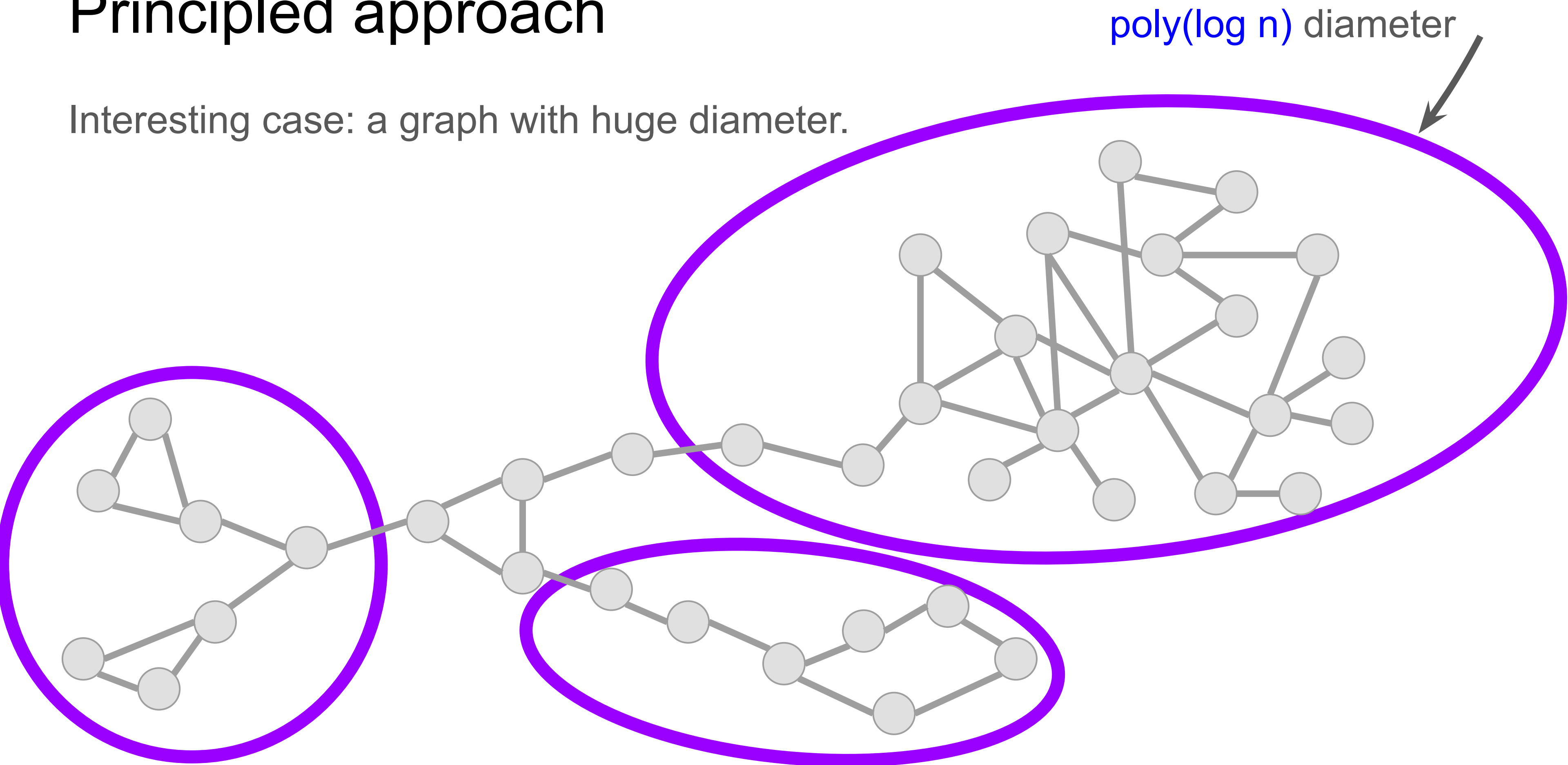
Principled approach

Interesting case: a graph with huge diameter.



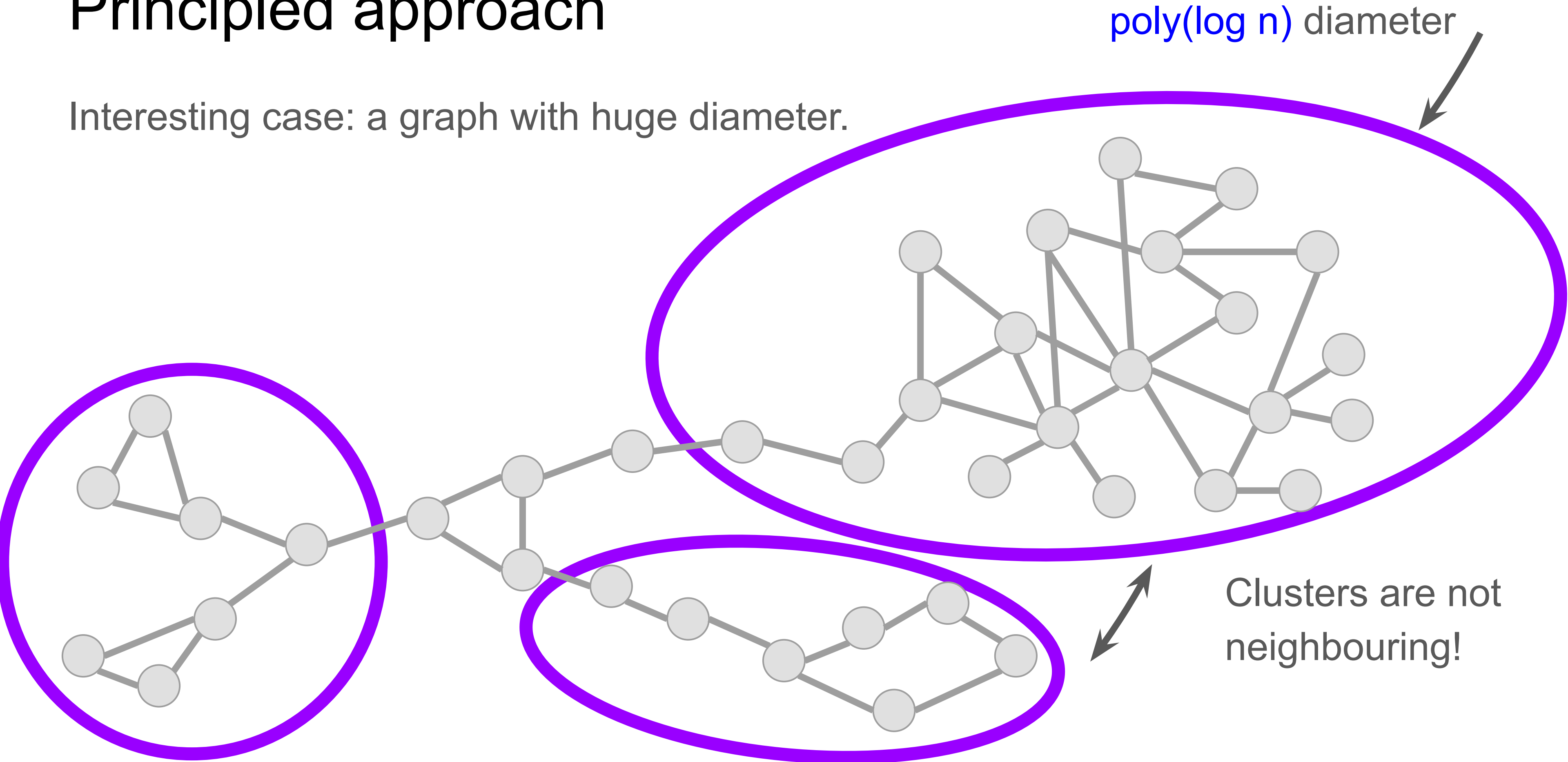
Principled approach

Interesting case: a graph with huge diameter.



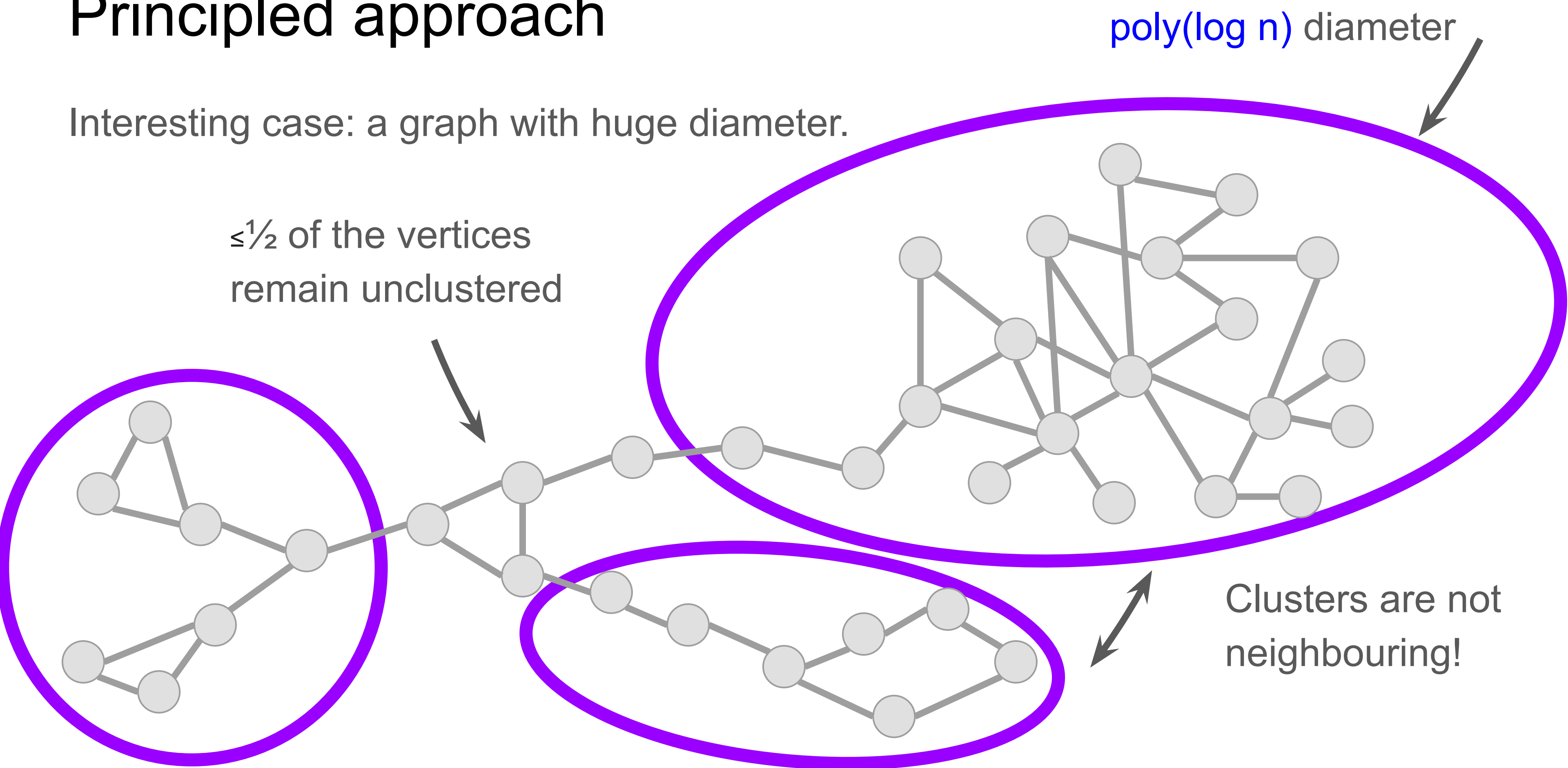
Principled approach

Interesting case: a graph with huge diameter.



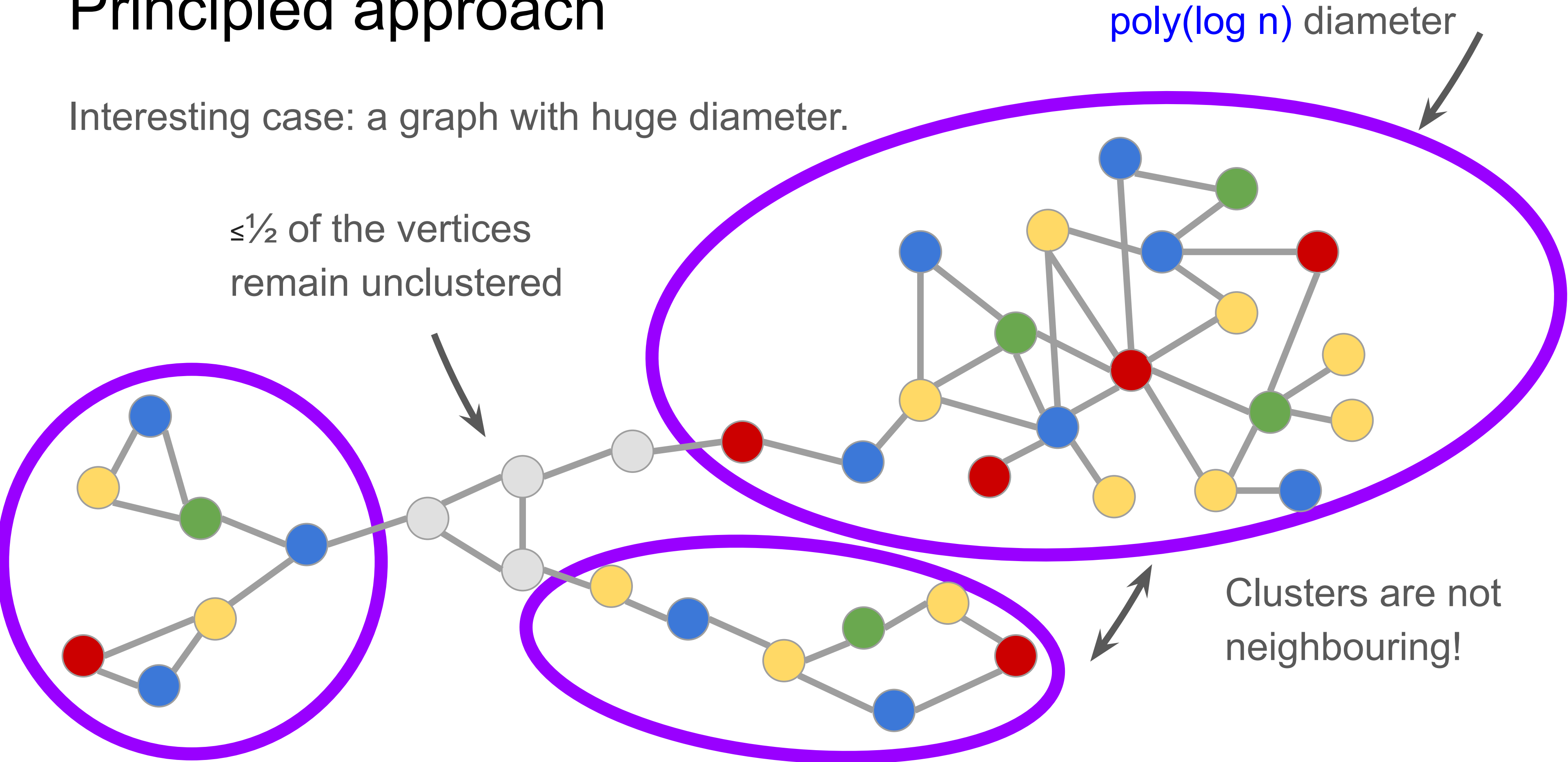
Principled approach

Interesting case: a graph with huge diameter.

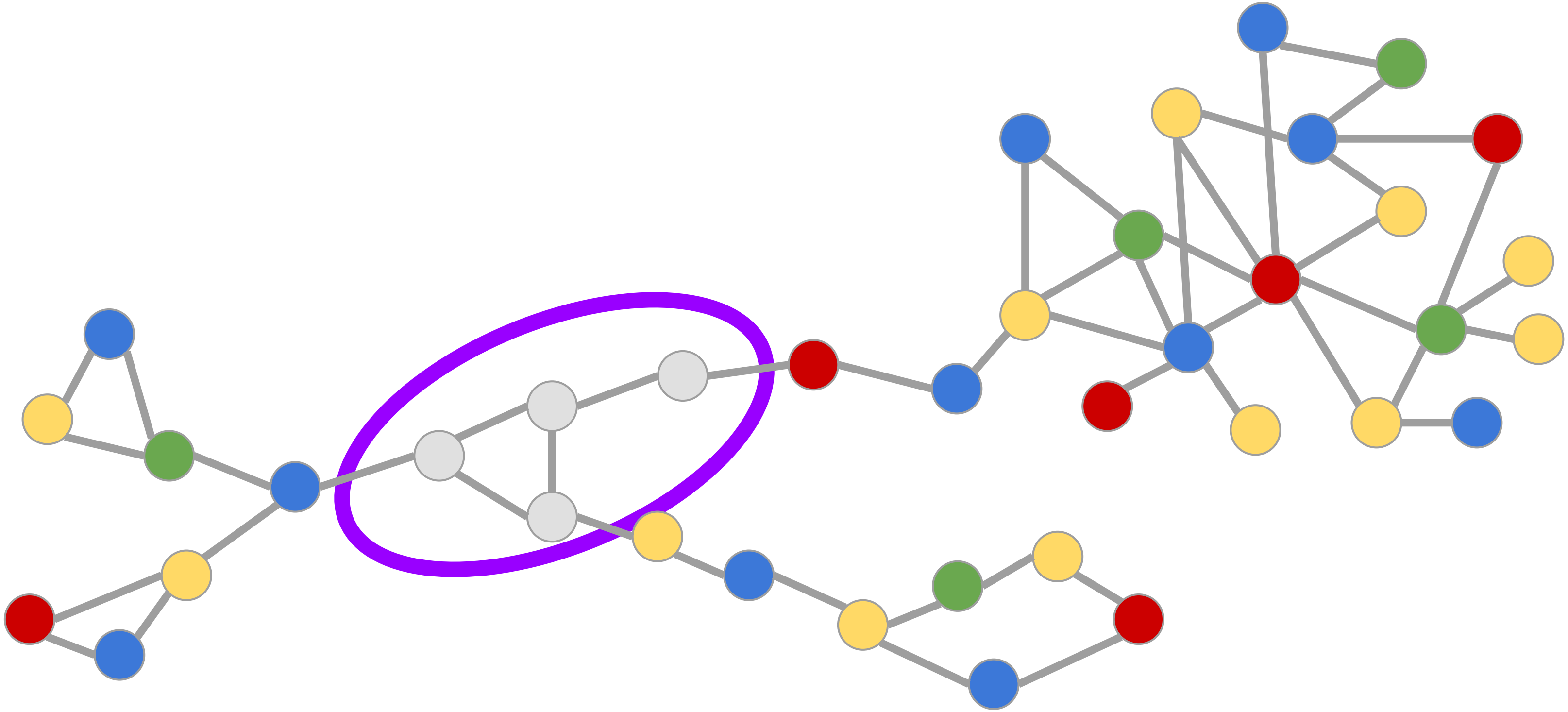


Principled approach

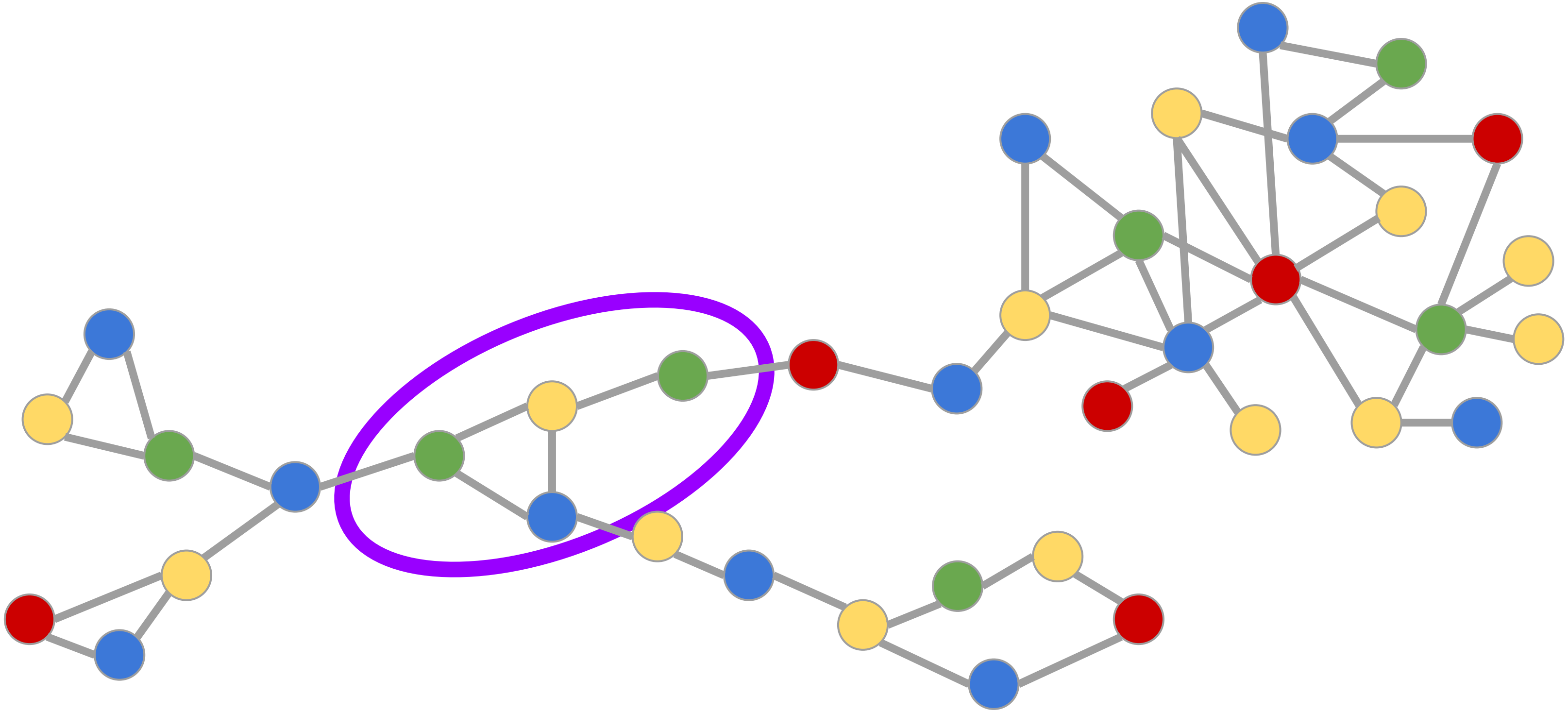
Interesting case: a graph with huge diameter.



Principled approach

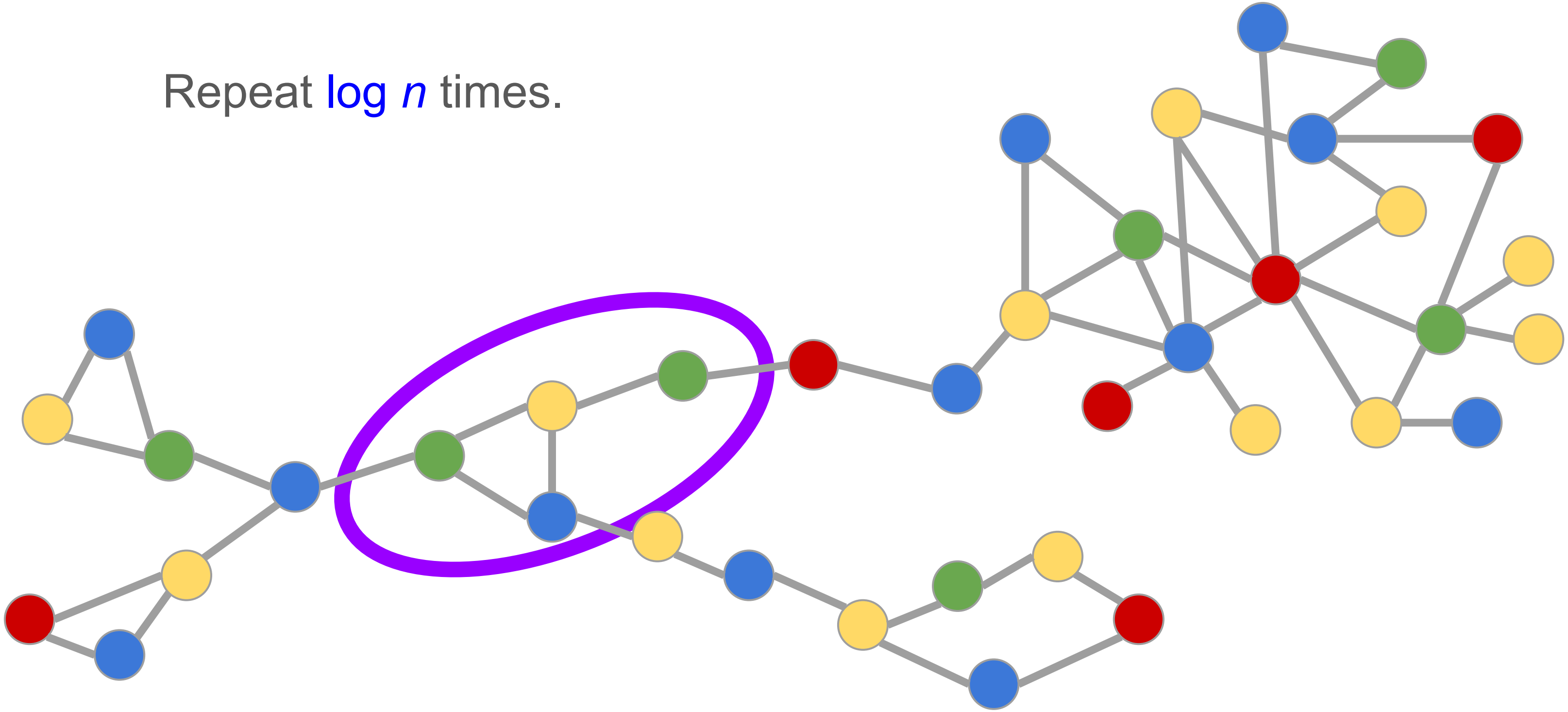


Principled approach



Principled approach

Repeat log *n* times.



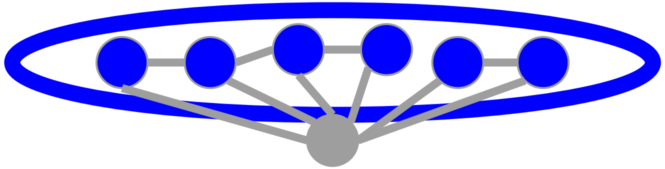
Making things formal

Dictionary

Network decomposition with **C** colors and diameter **D**:

Coloring of the vertices with **C** colors, such that each component induced by a particular color has diameter at most **D**.

Dictionary



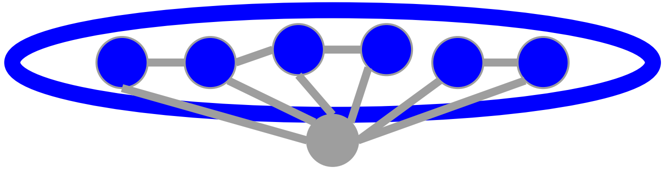
Network decomposition with **C** colors and diameter **D**:

Coloring of the vertices with **C** colors, such that each component induced by a particular color has diameter at most **D**.

Weak-diameter network decomposition

...any two vertices of a cluster are at most **D** hops apart in the original graph.

Dictionary



Network decomposition with **C** colors and diameter **D**:

Coloring of the vertices with **C** colors, such that each component induced by a particular color has diameter at most **D**.

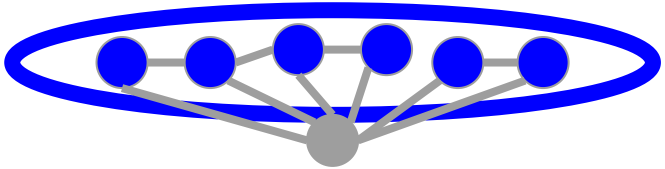
Weak-diameter network decomposition

...any two vertices of a cluster are at most **D** hops apart in the original graph.

Ball carving

Algorithm (that I show next) that finds independent clusters of diameter $O(\log n)$ while leaving at most ½ vertices unclustered.

Dictionary



Network decomposition with **C** colors and diameter **D**:

Coloring of the vertices with **C** colors, such that each component induced by a particular color has diameter at most **D**.

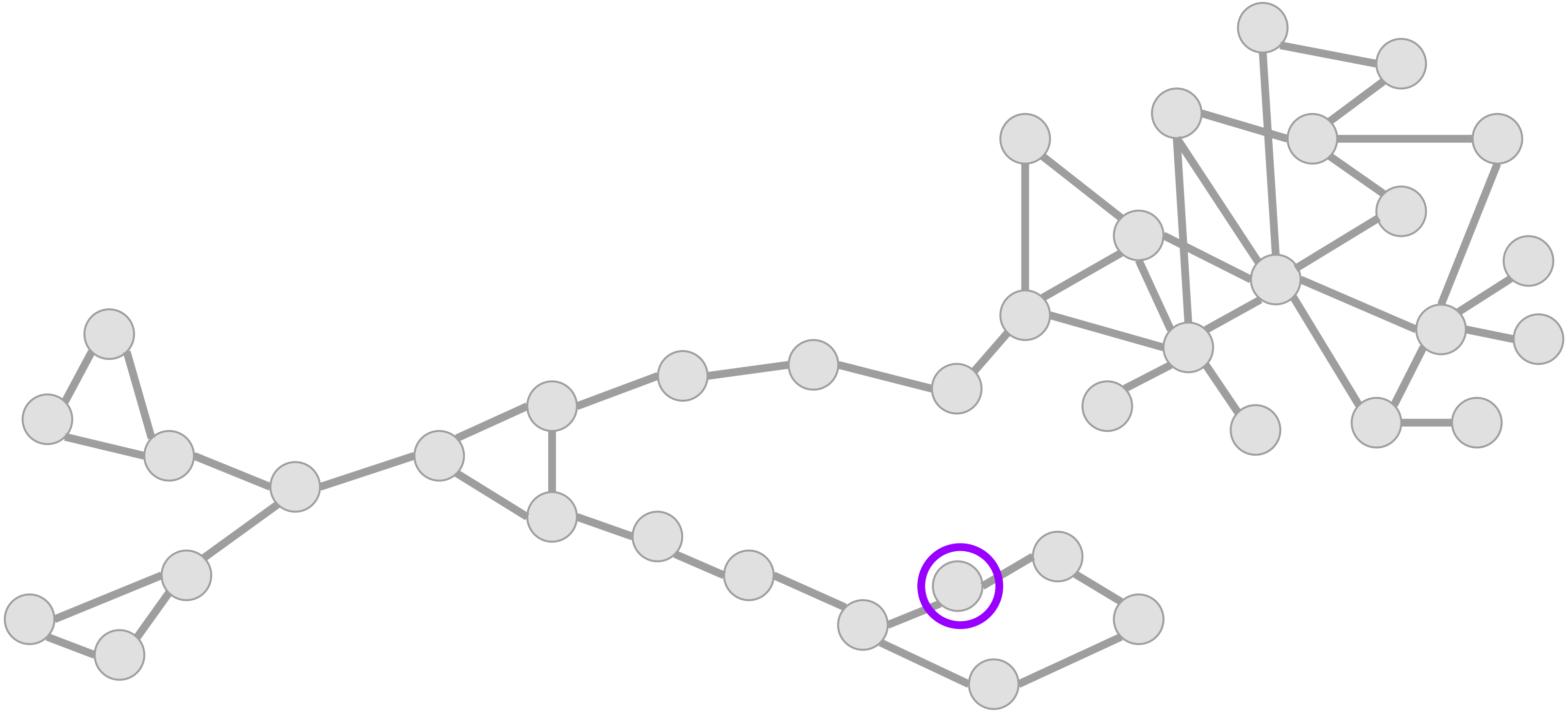
Weak-diameter network decomposition

...any two vertices of a cluster are at most **D** hops apart in the original graph.

Ball carving

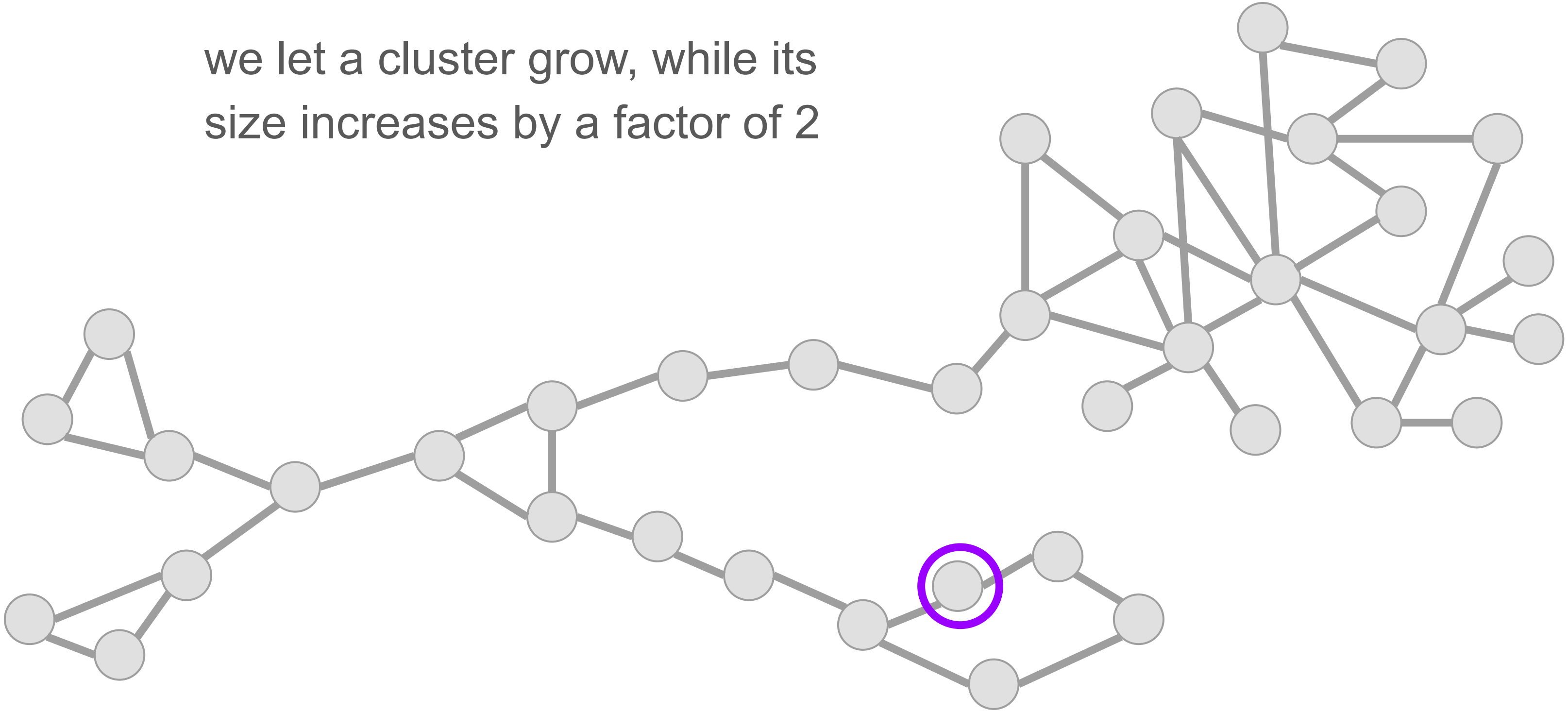
Algorithm (that I show next) that finds independent clusters of diameter $O(\log n)$ while leaving at most ½ vertices unclustered.
This implies existence of decomposition with $\mathbf{C}=O(\log n)$ and $\mathbf{D}=O(\log n)$.

Sequential ball carving

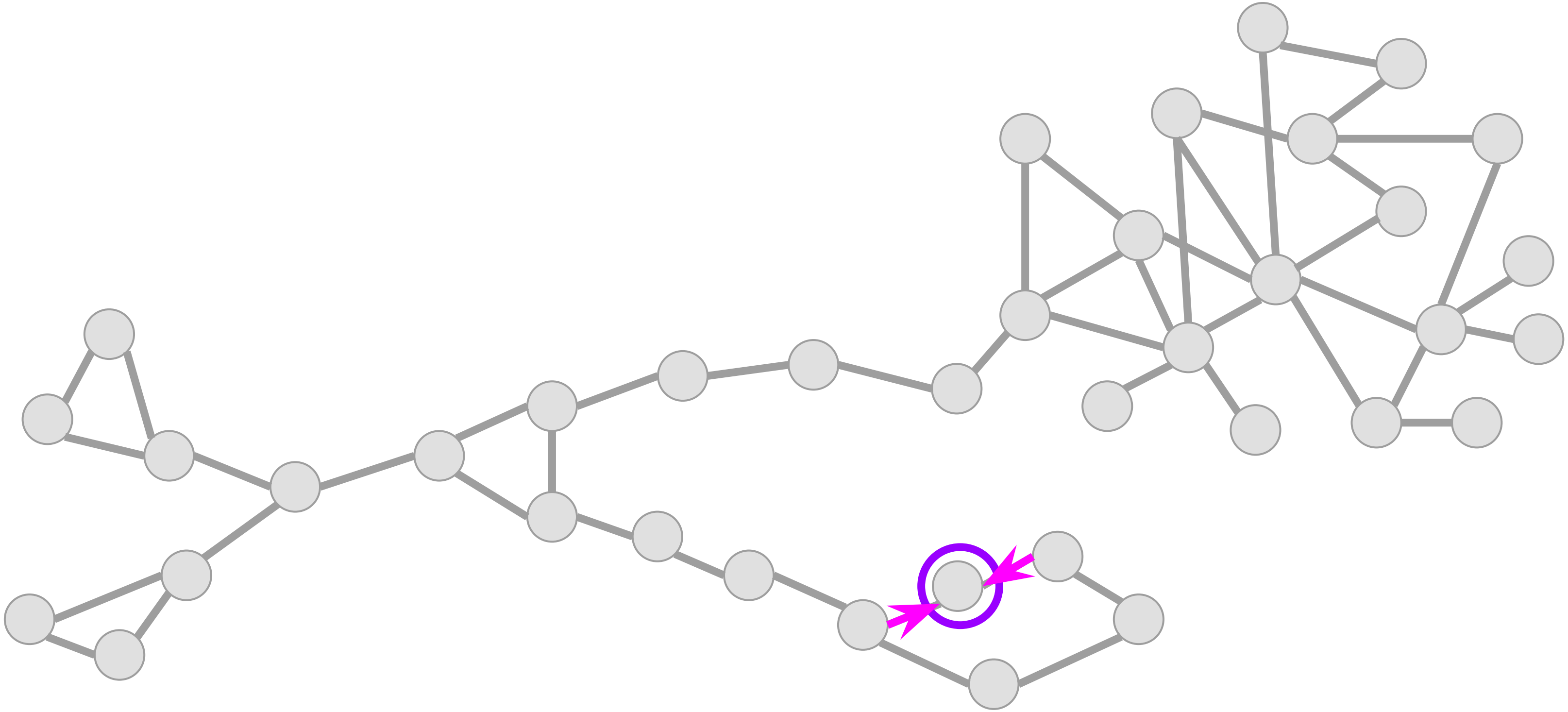


Sequential ball carving

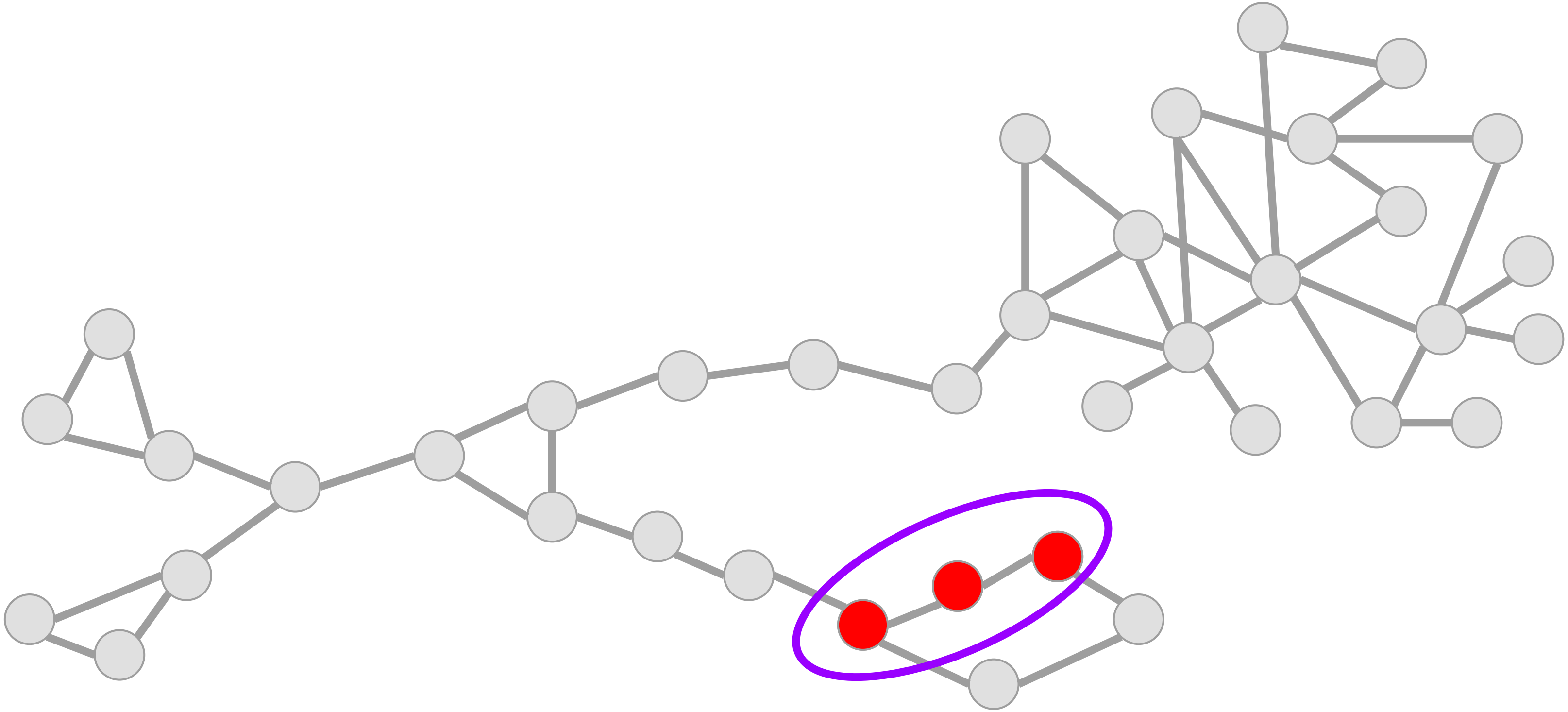
we let a cluster grow, while its size increases by a factor of 2



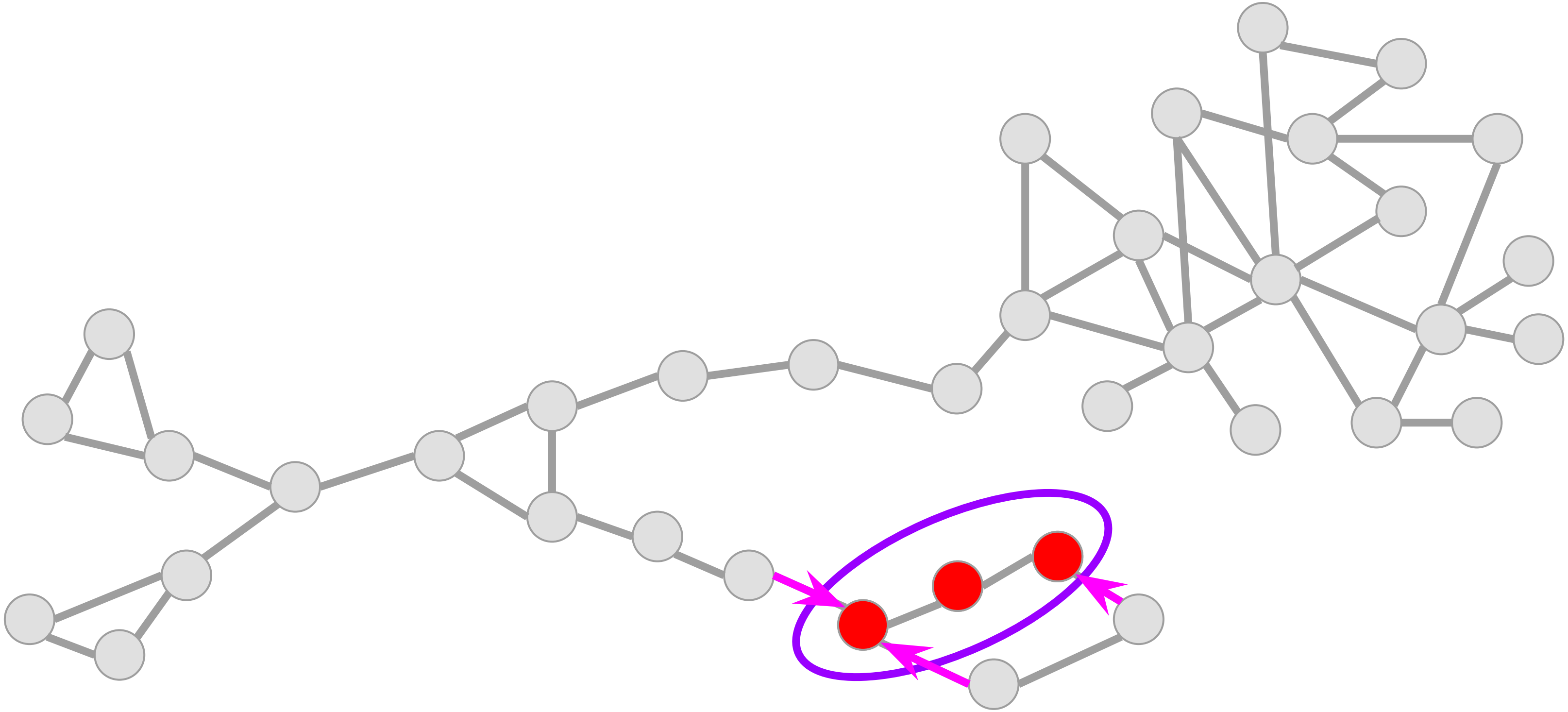
Sequential ball carving



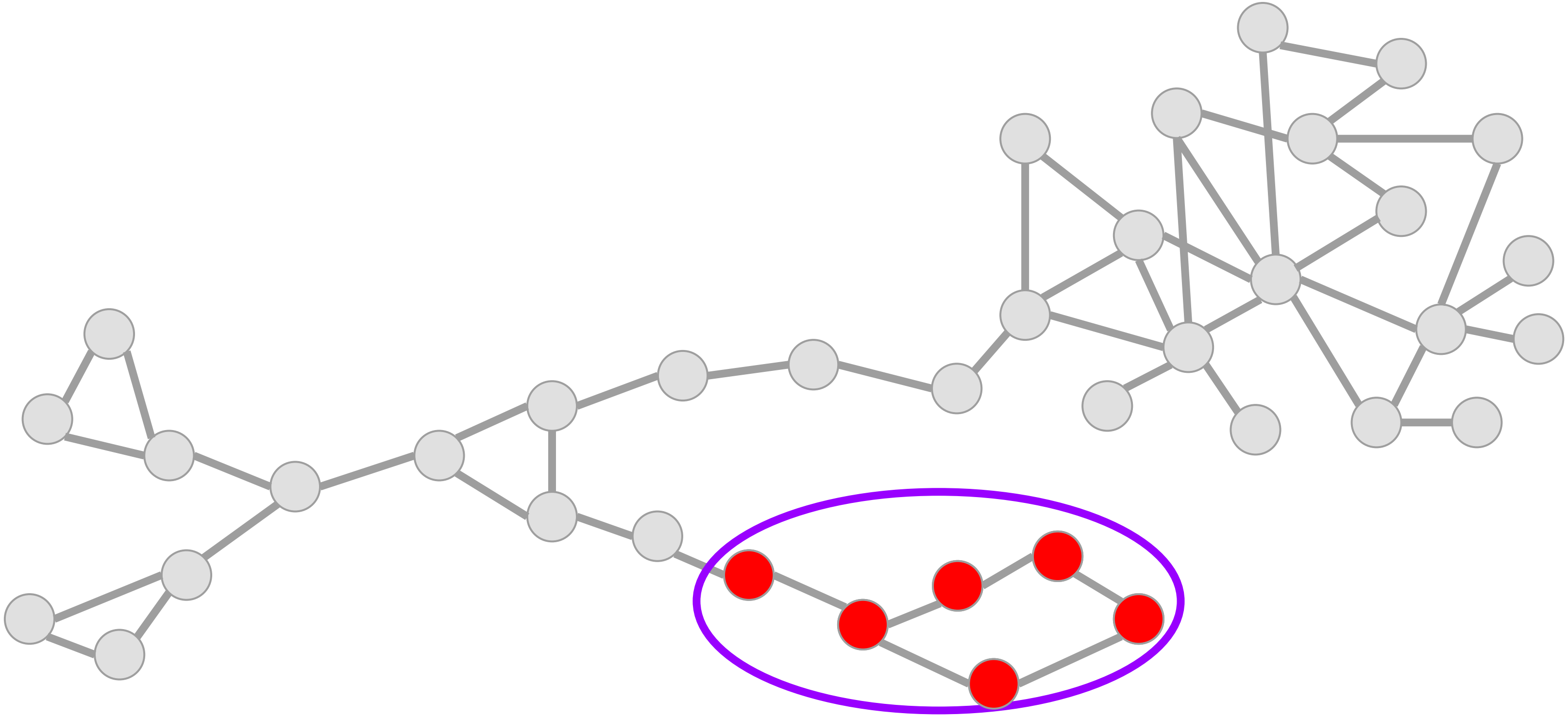
Sequential ball carving



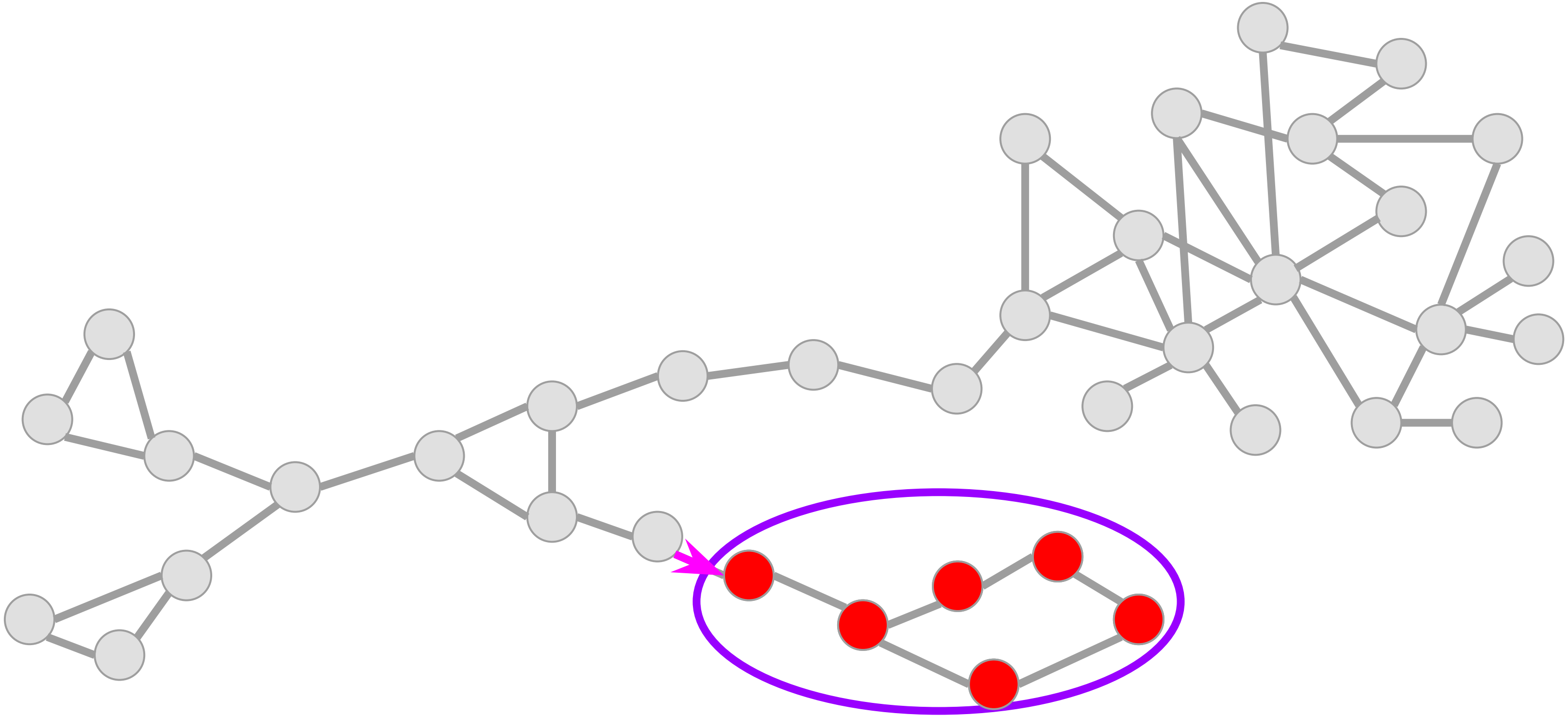
Sequential ball carving



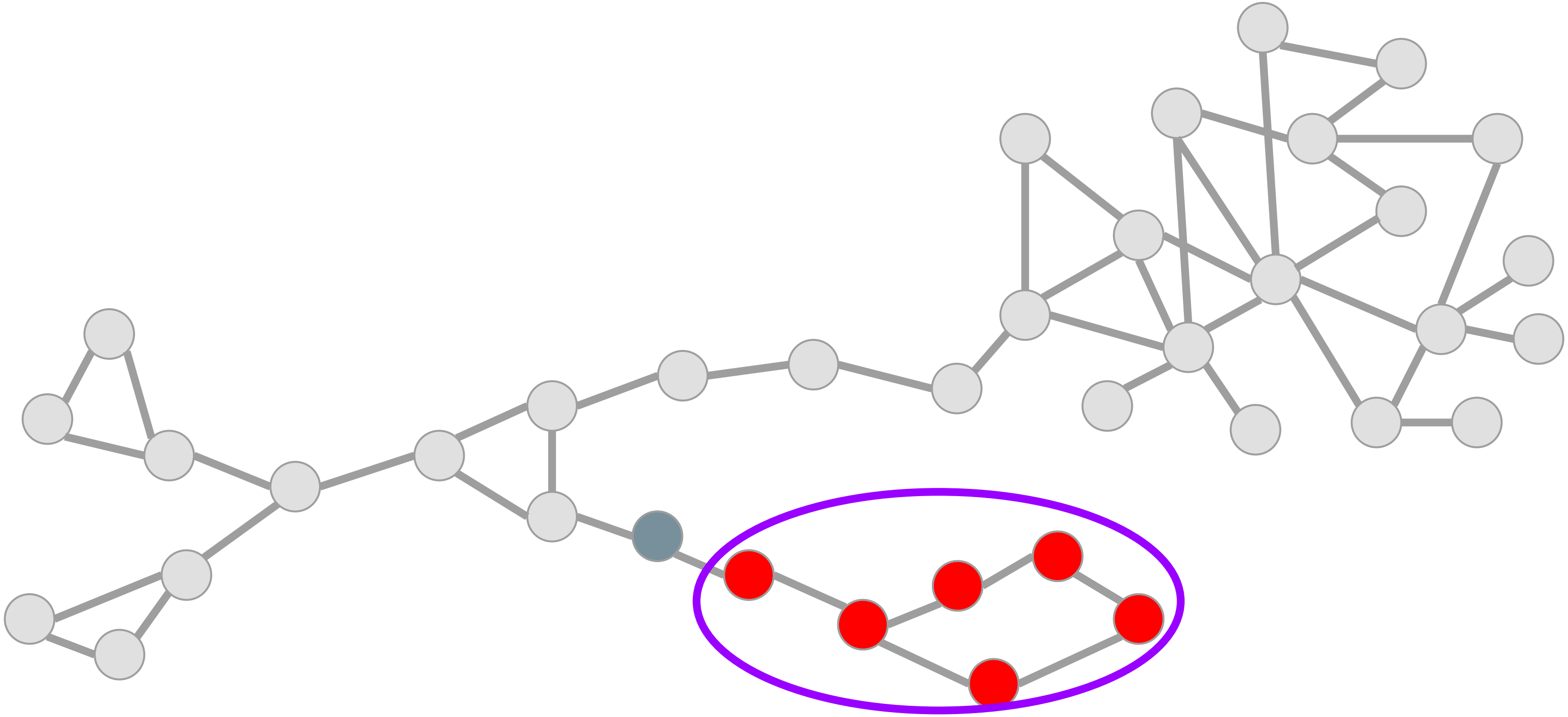
Sequential ball carving



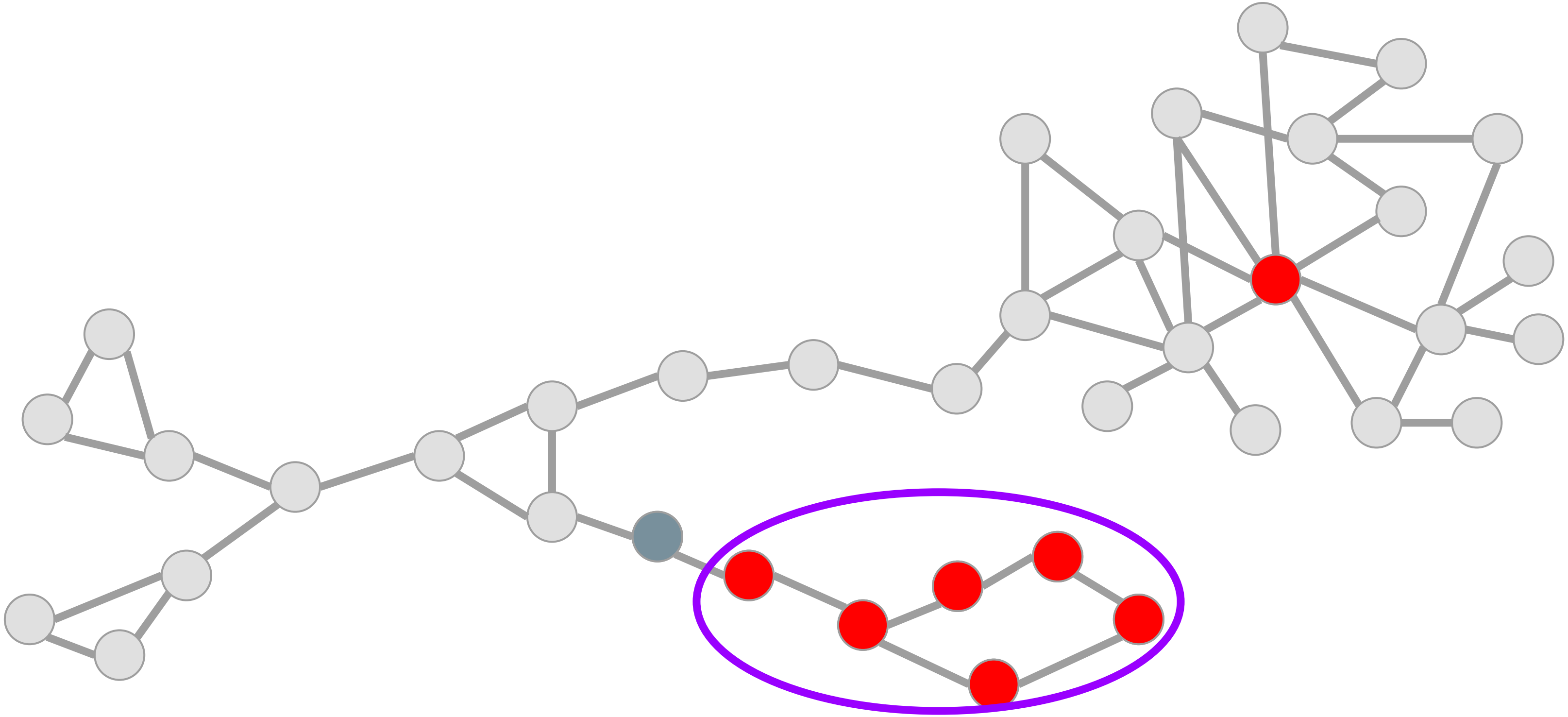
Sequential ball carving



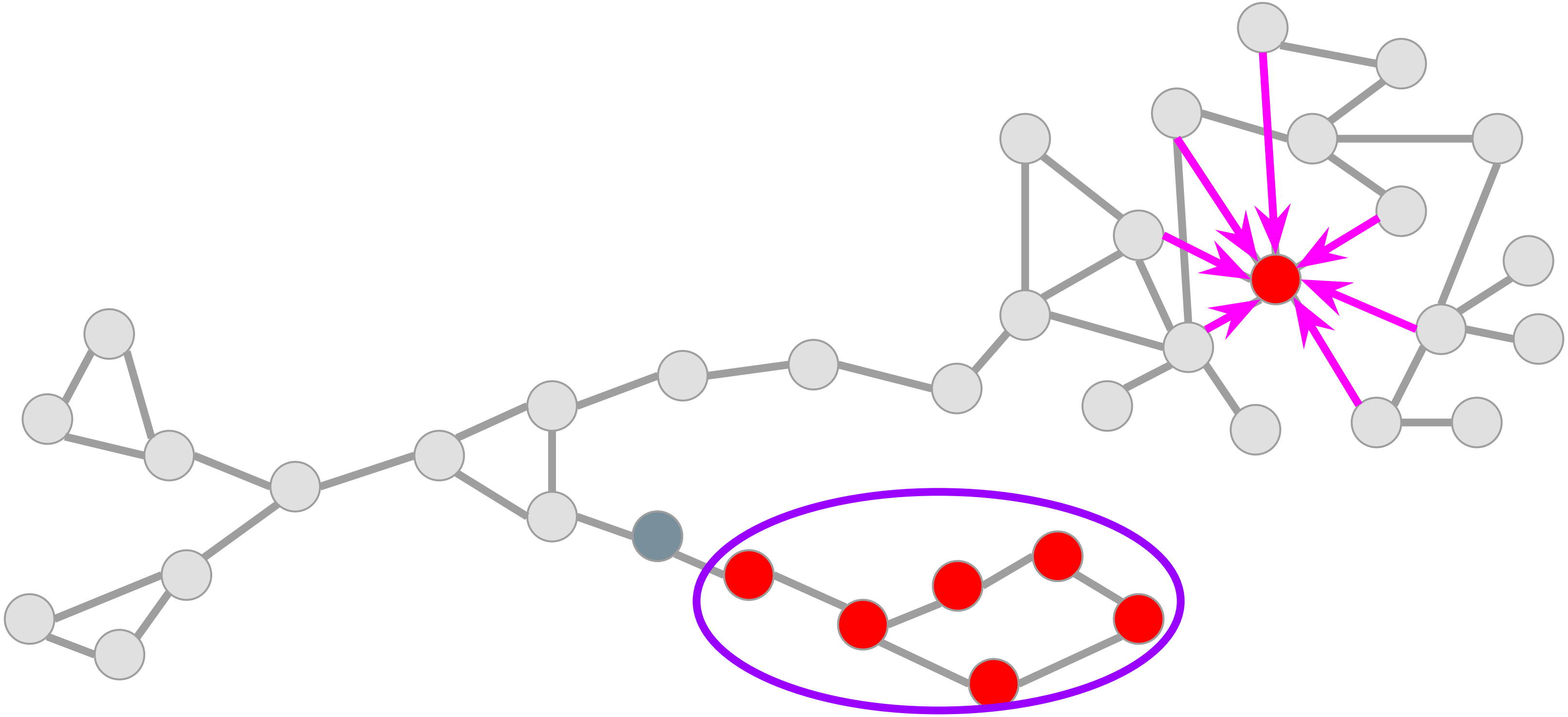
Sequential ball carving



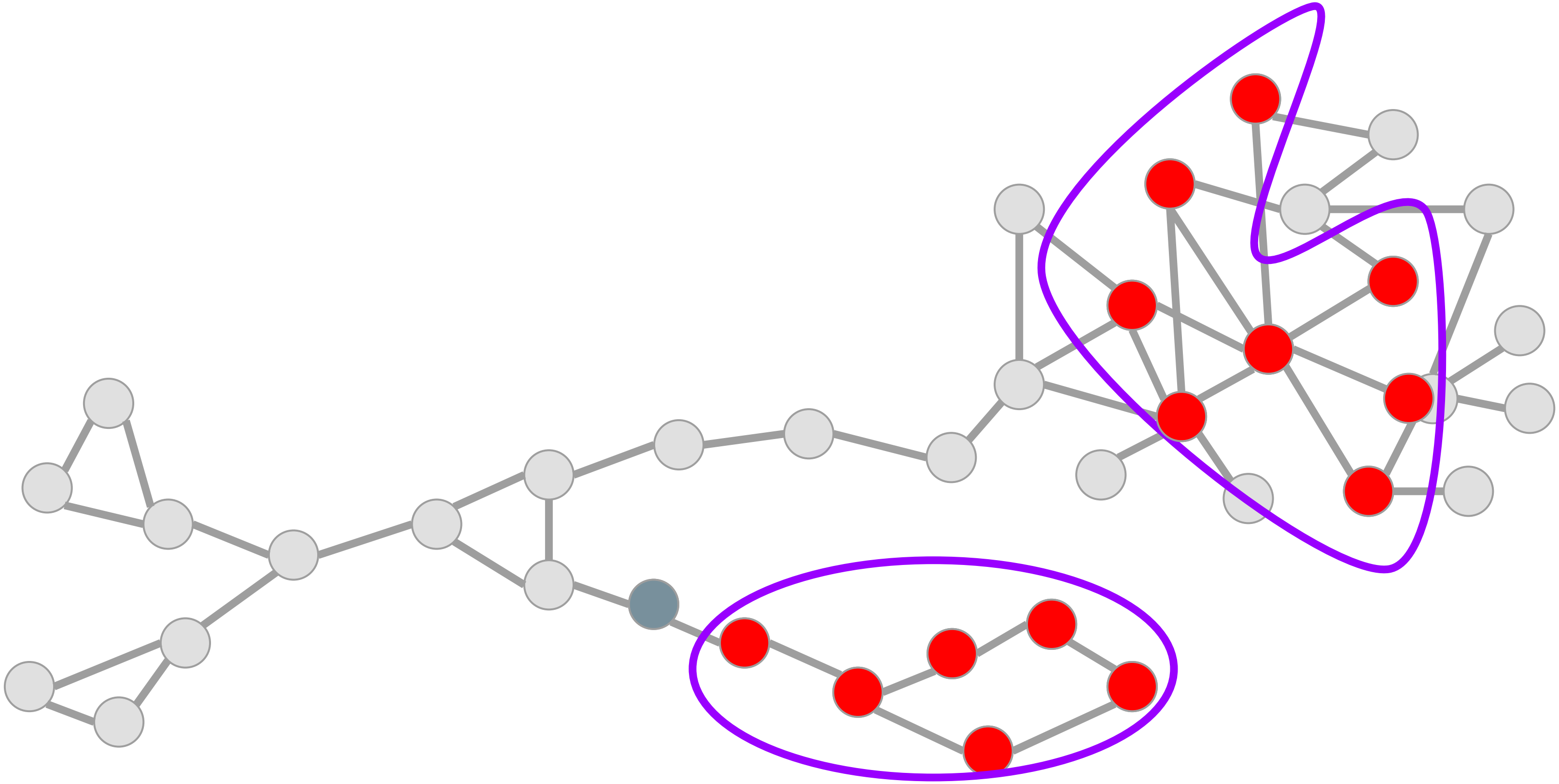
Sequential ball carving



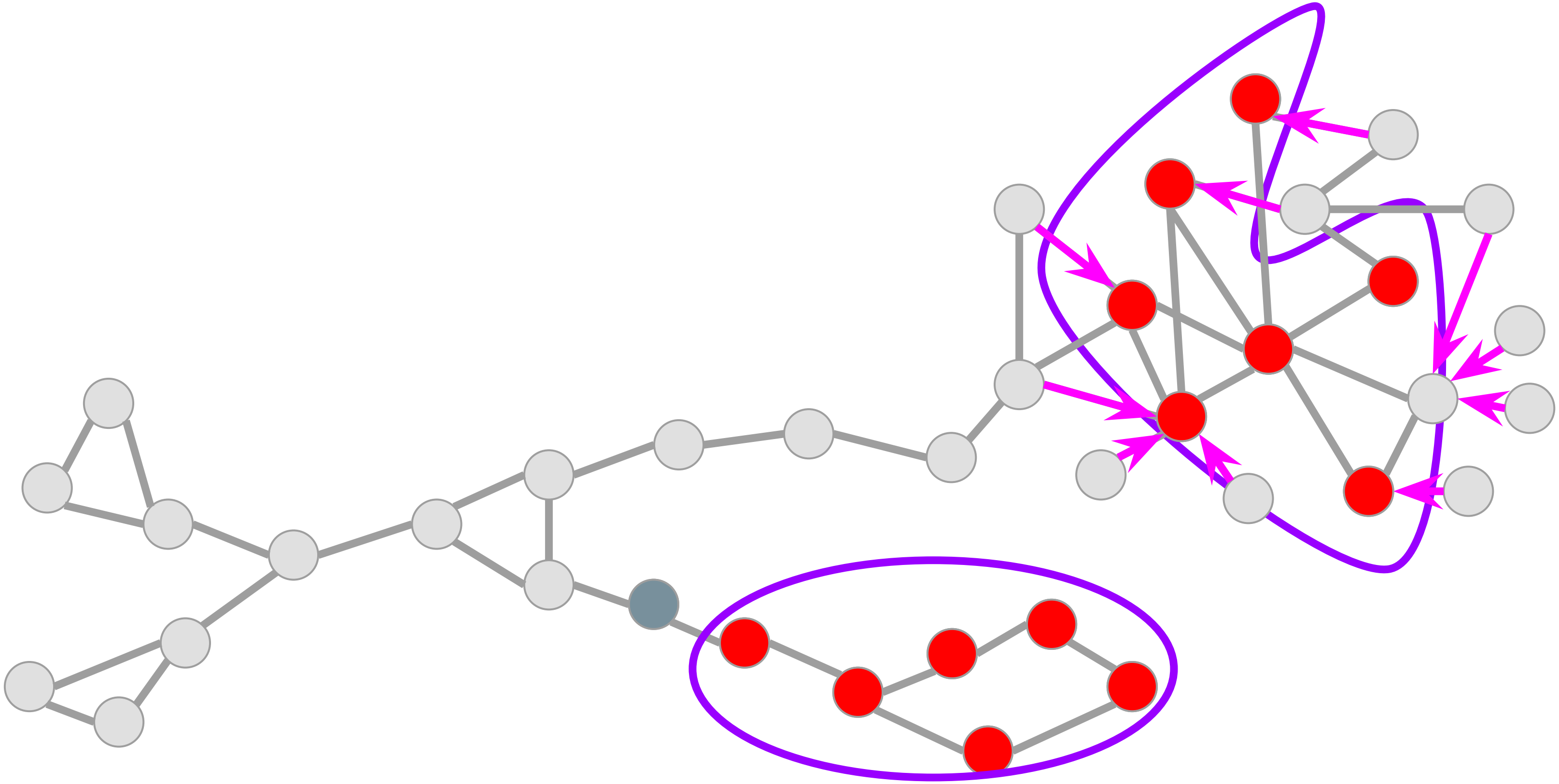
Sequential ball carving



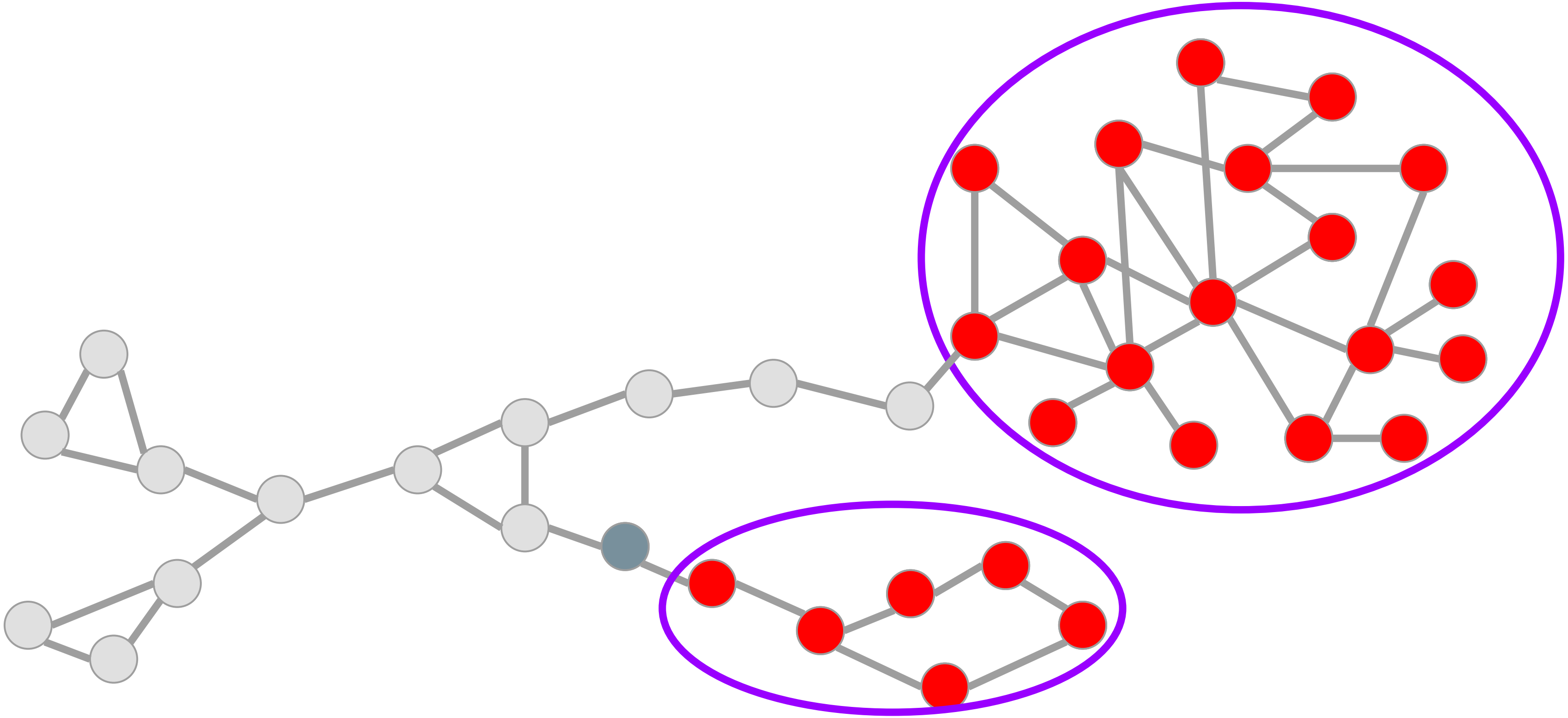
Sequential ball carving



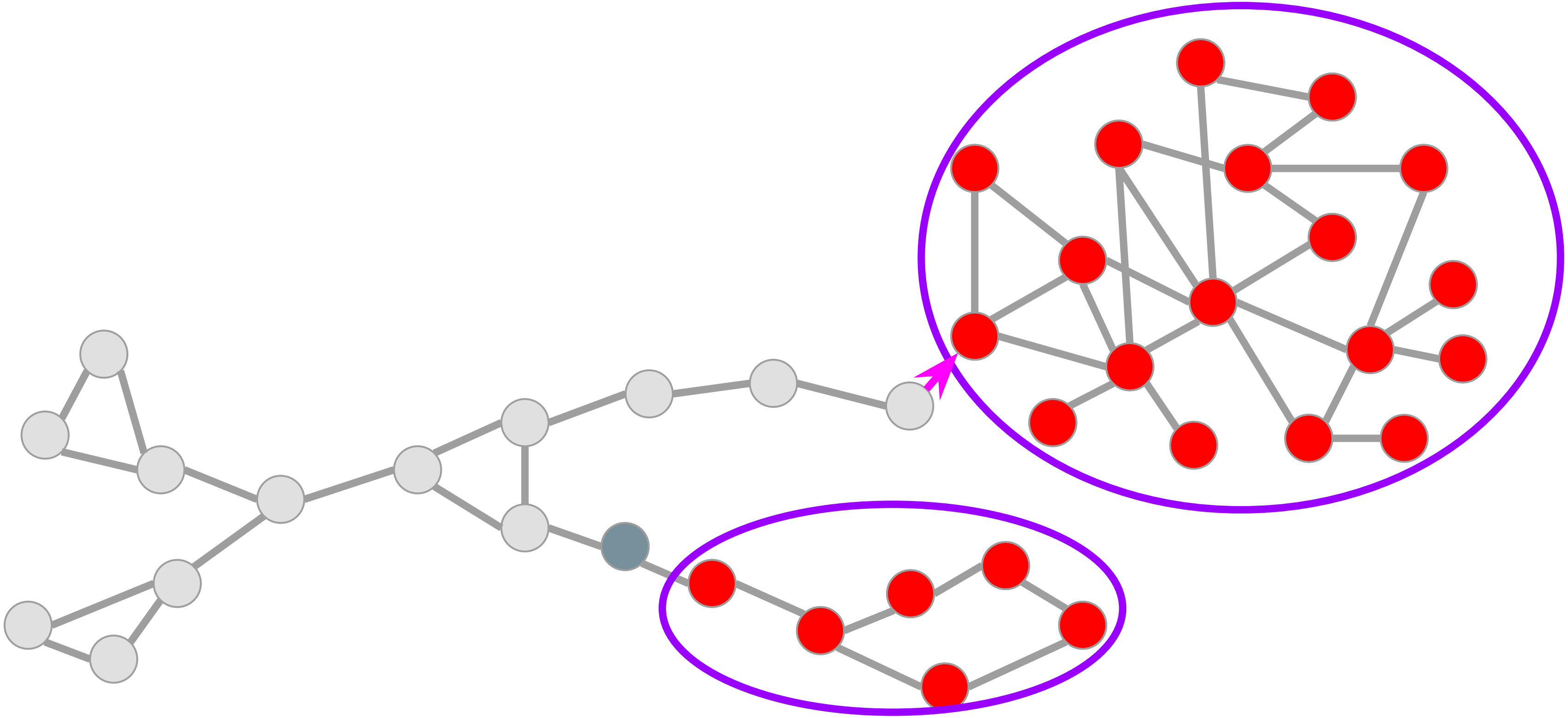
Sequential ball carving



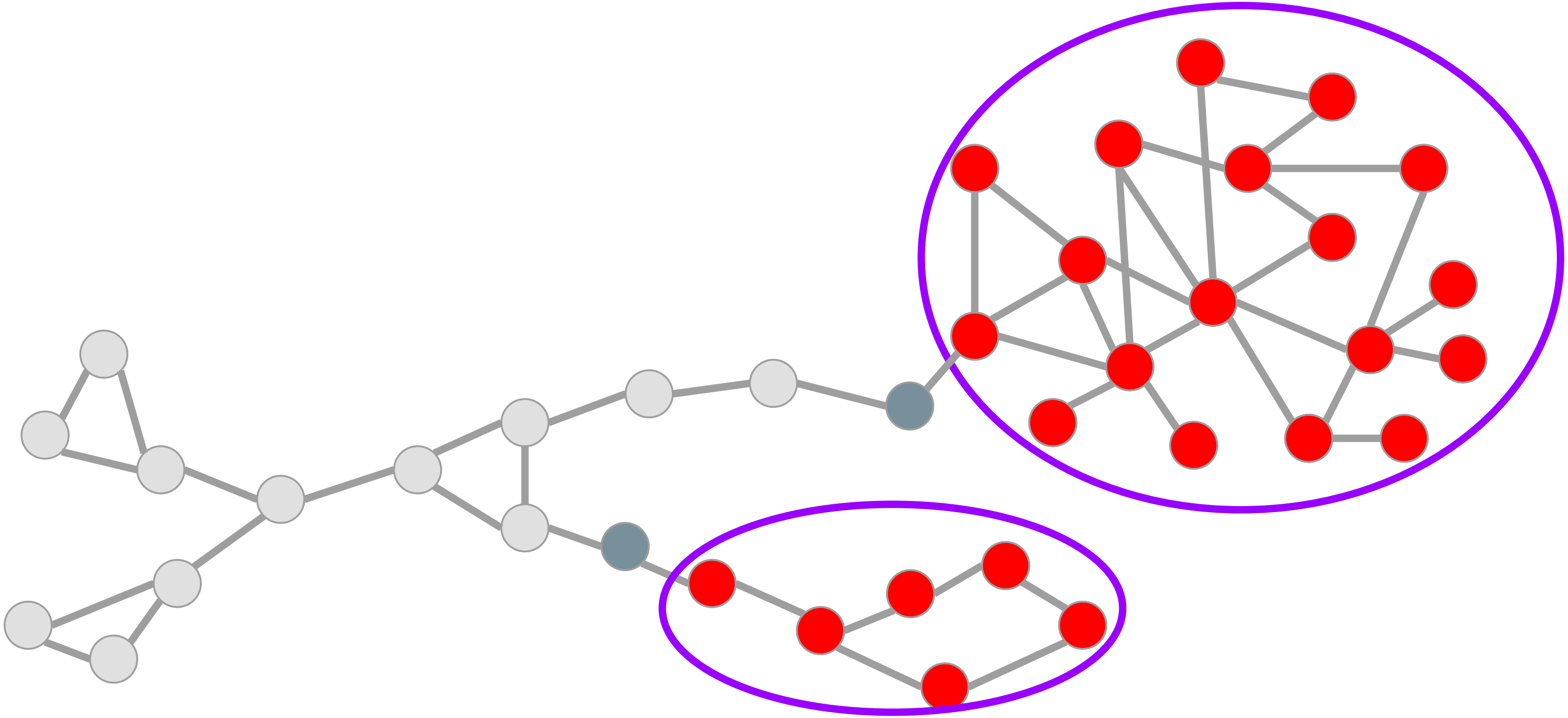
Sequential ball carving



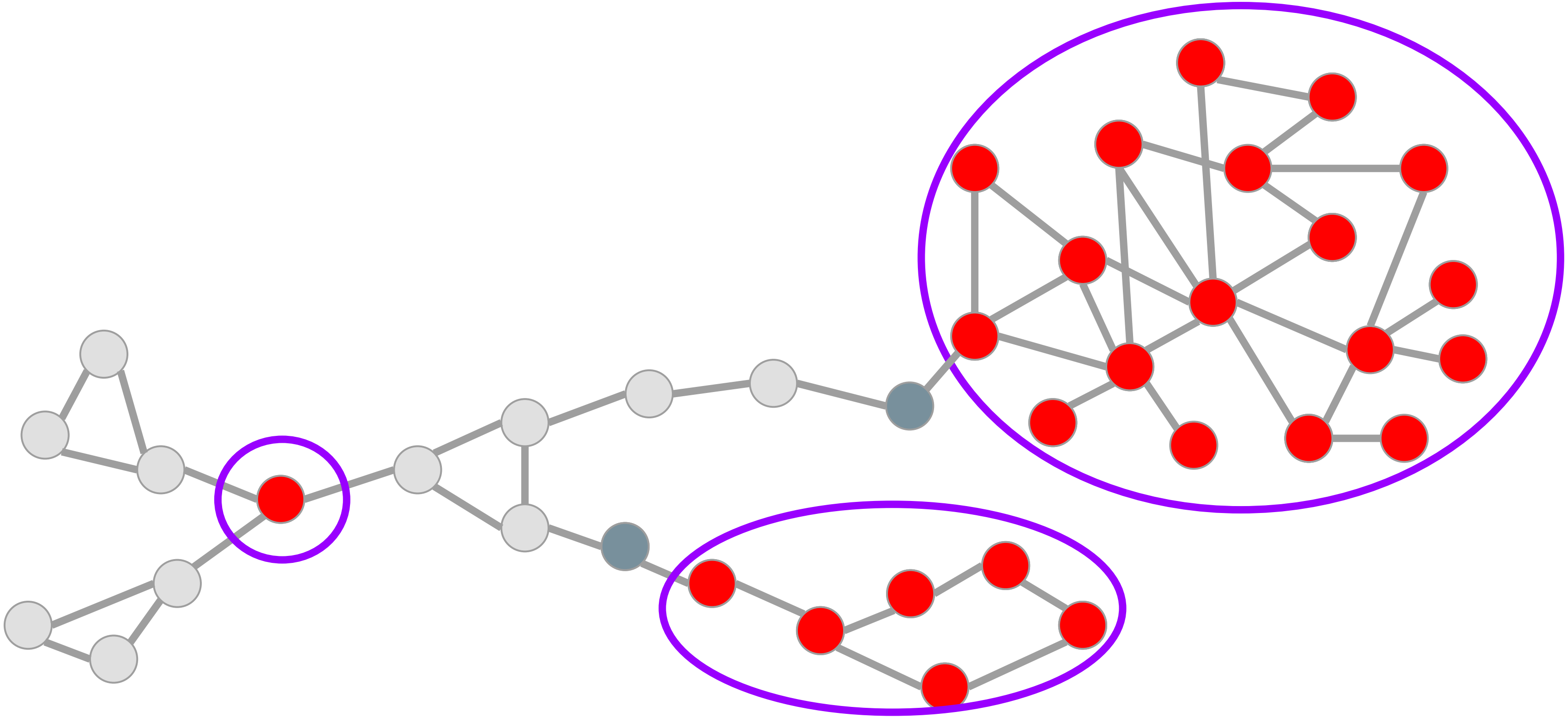
Sequential ball carving



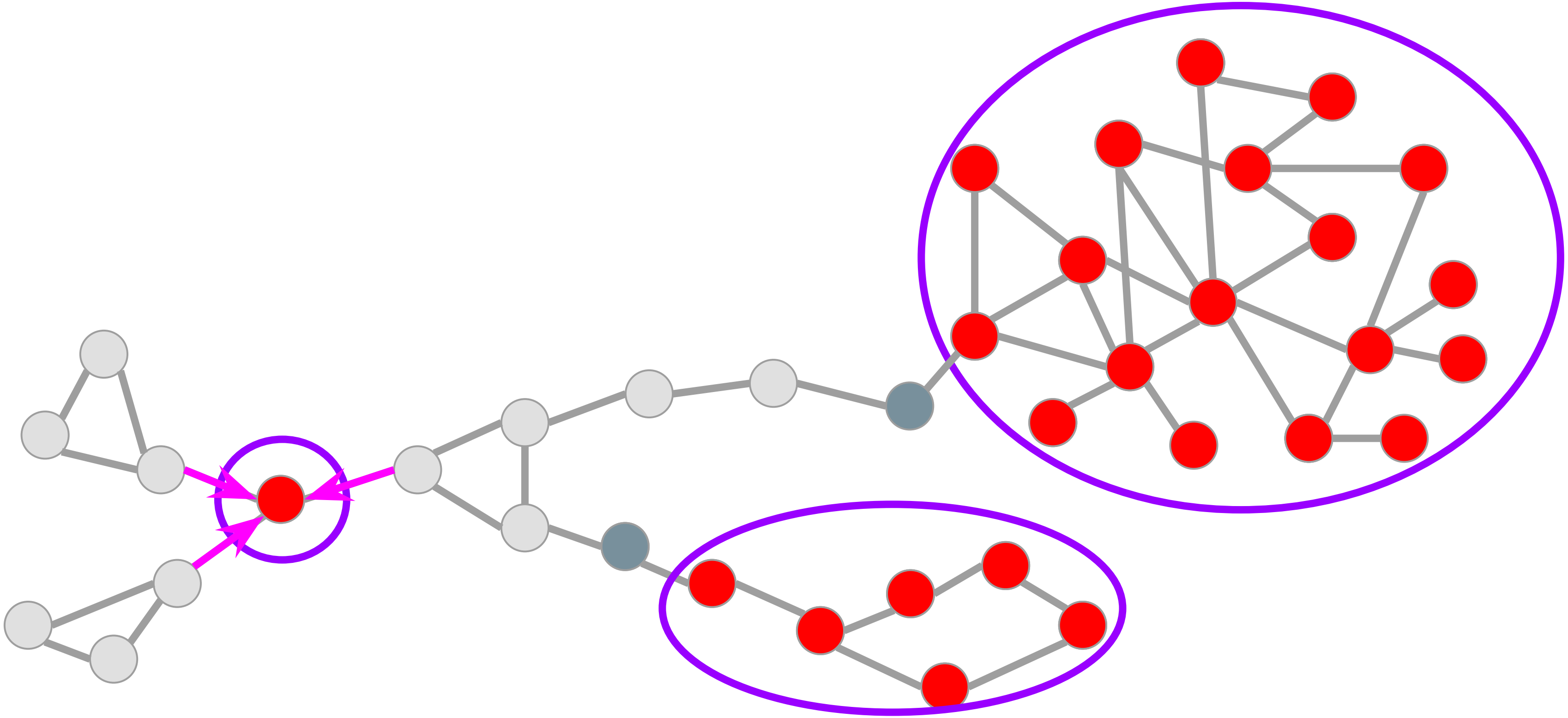
Sequential ball carving



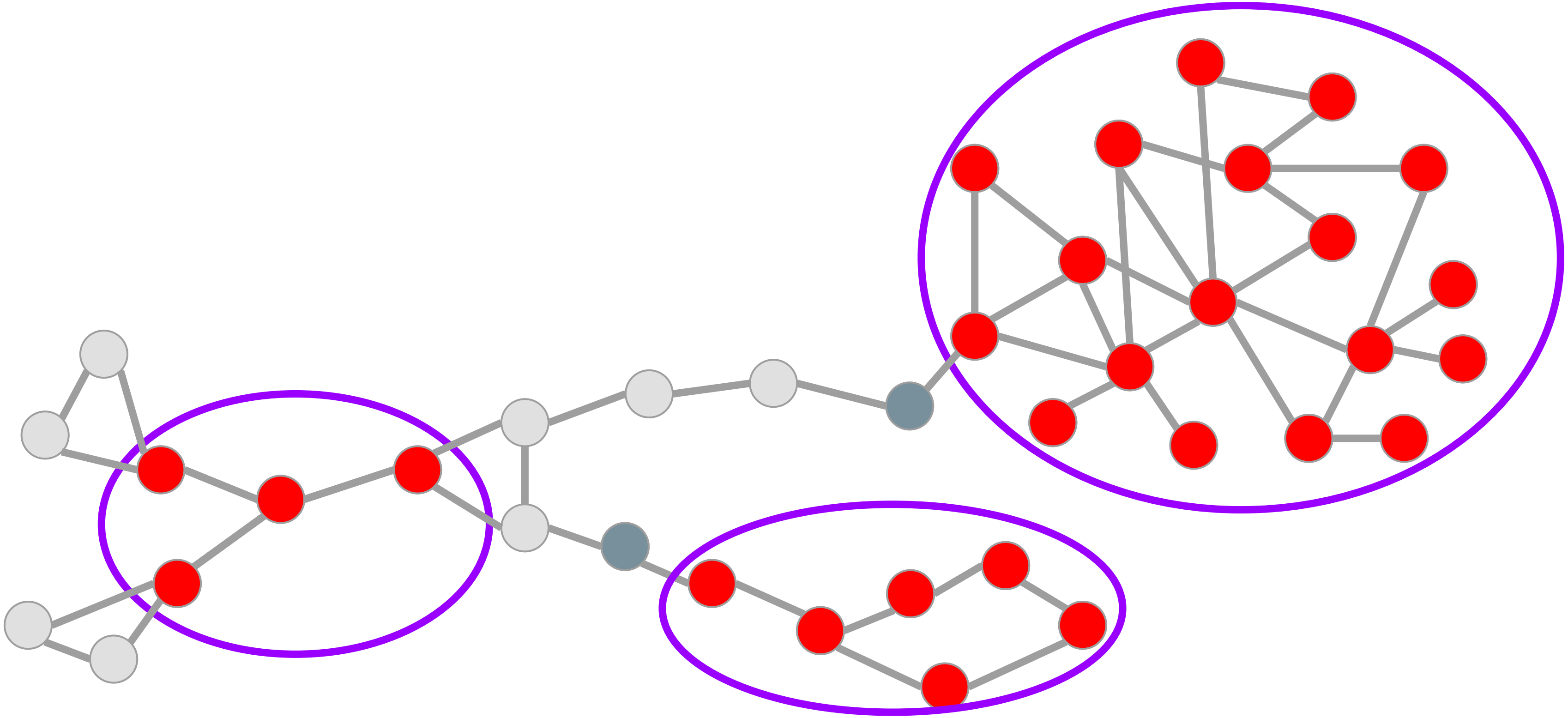
Sequential ball carving



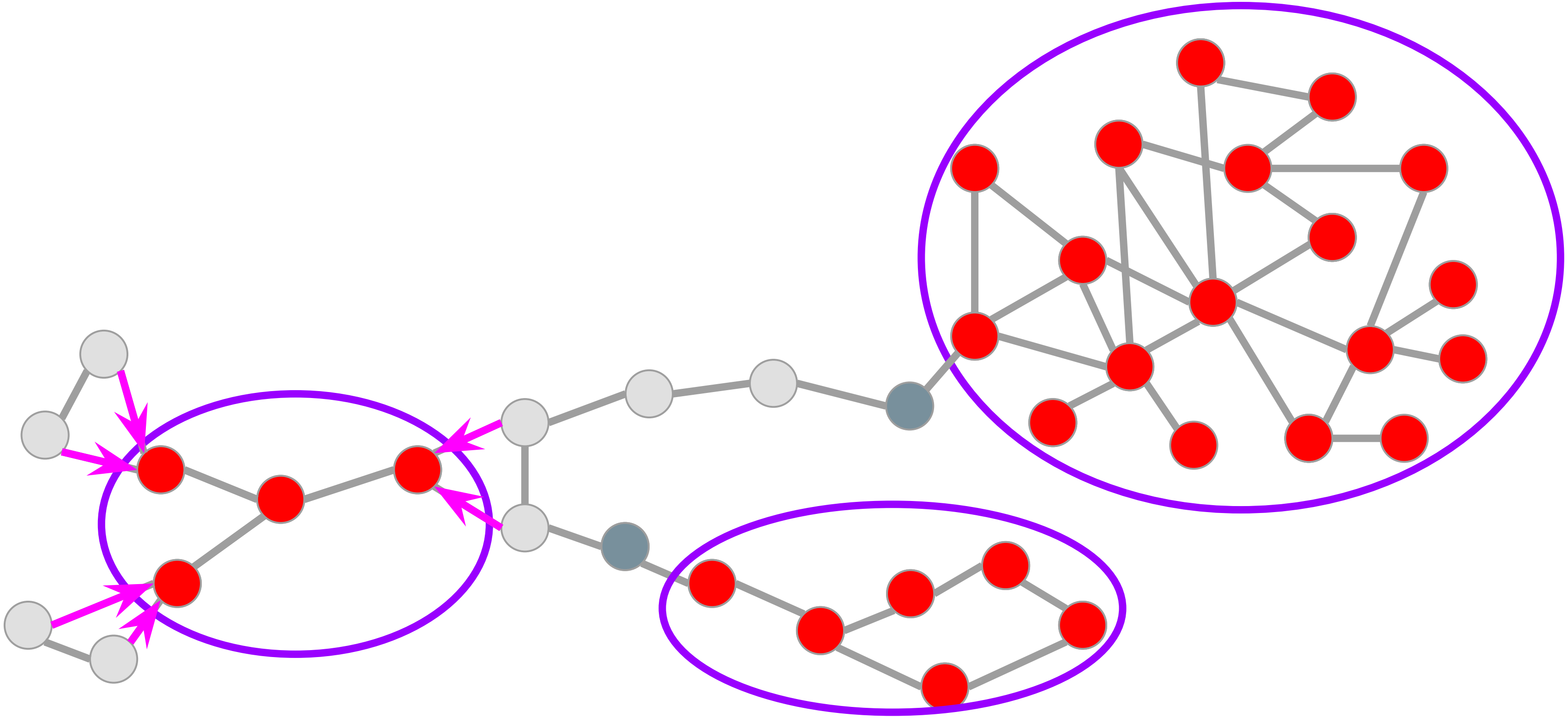
Sequential ball carving



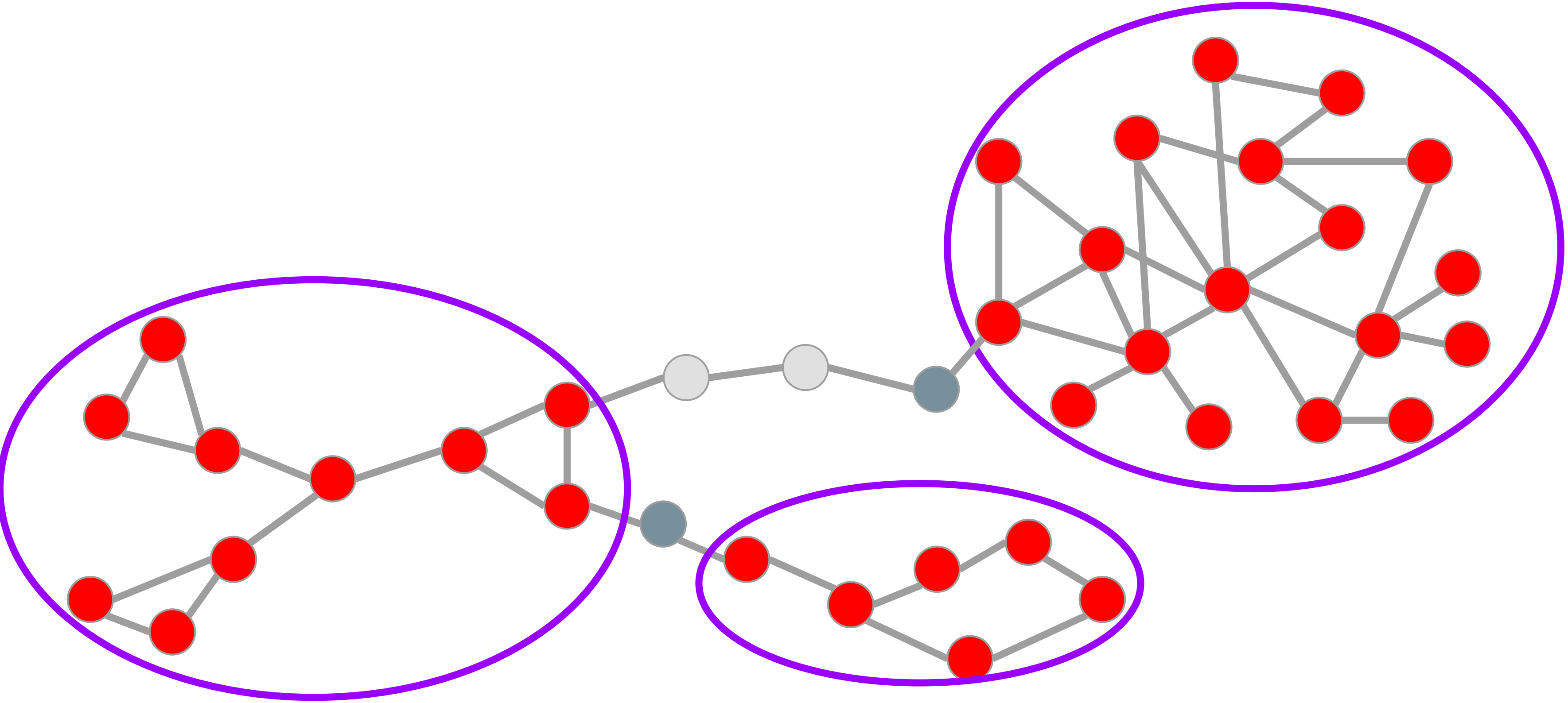
Sequential ball carving



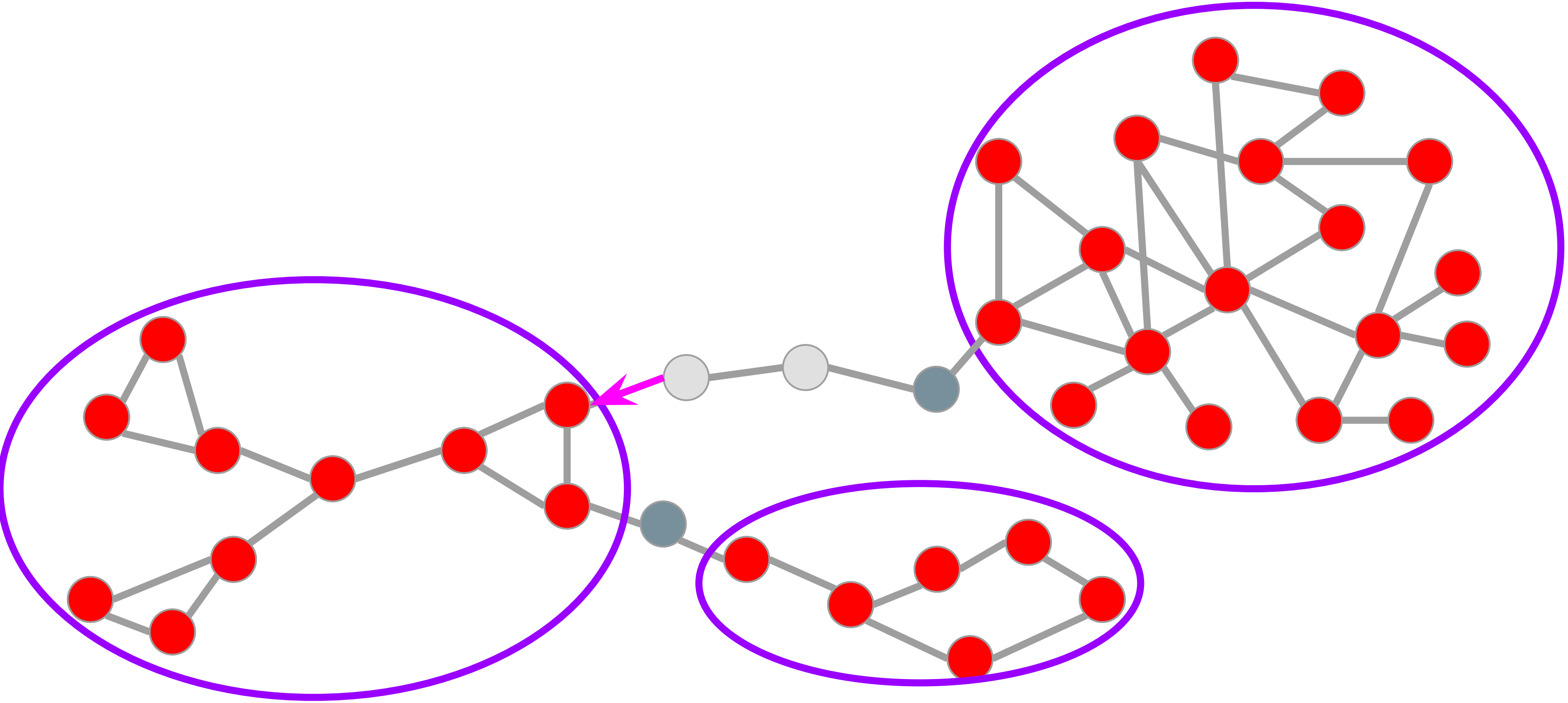
Sequential ball carving



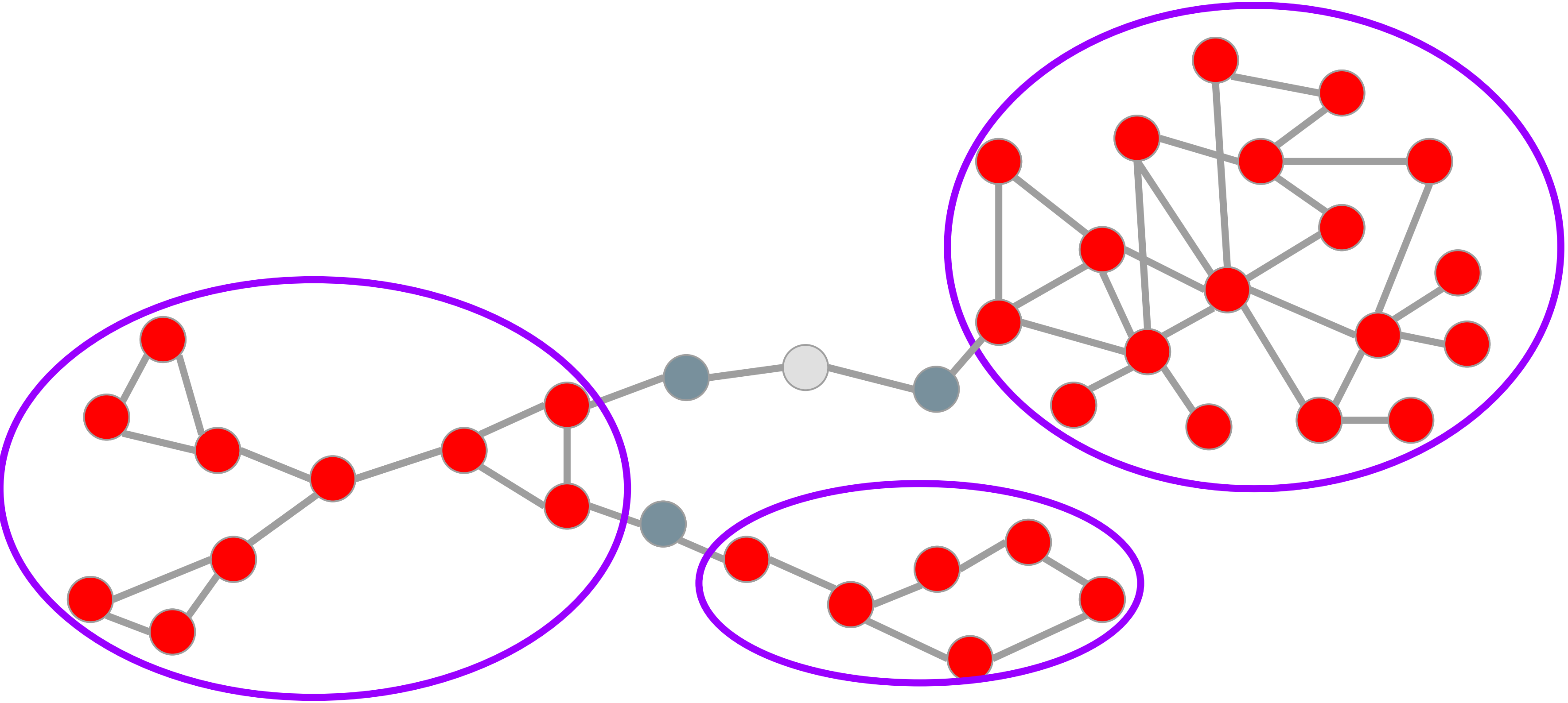
Sequential ball carving



Sequential ball carving



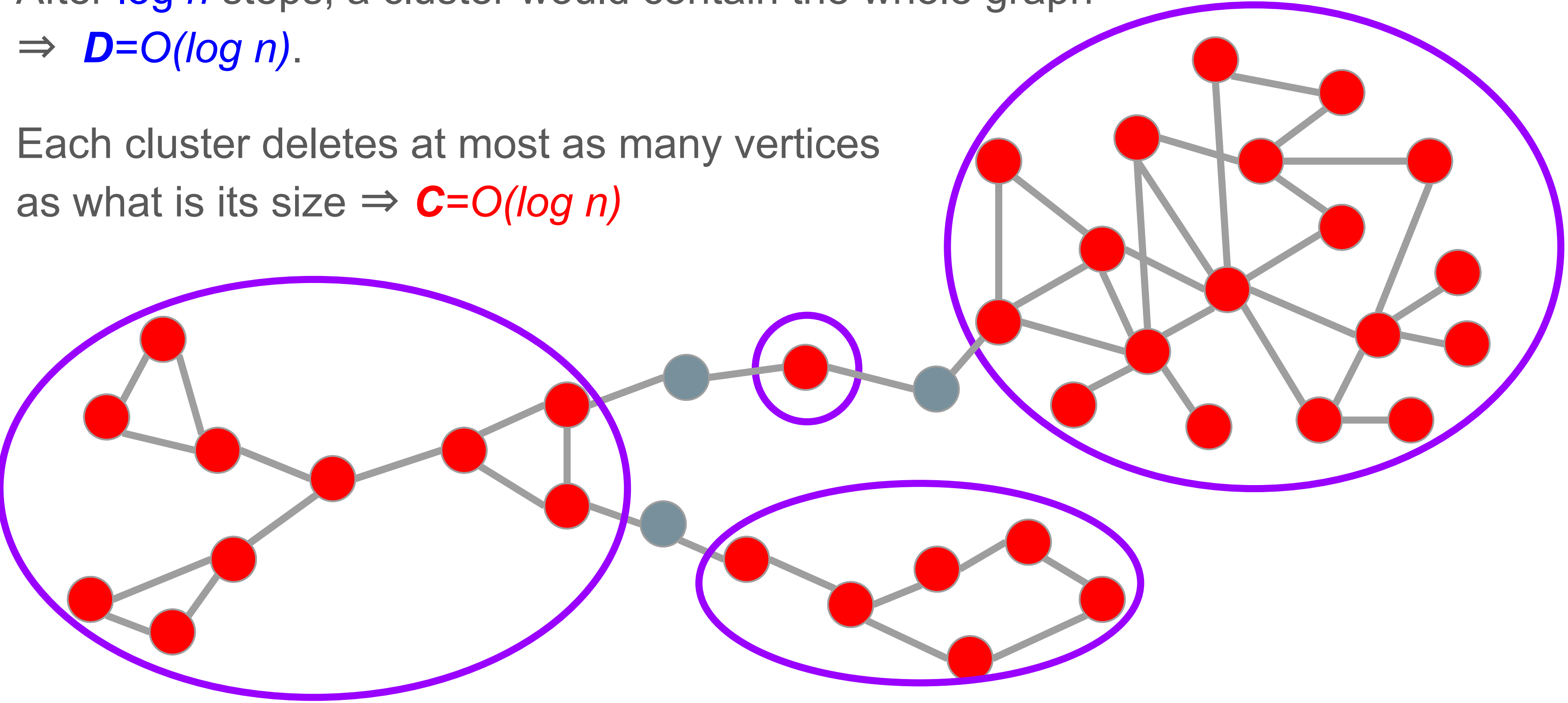
Sequential ball carving



Sequential ball carving

After log n steps, a cluster would contain the whole graph
$\Rightarrow$ $\boldsymbol{D}=O(\log n)$.

Each cluster deletes at most as many vertices as what is its size $\Rightarrow$ $\boldsymbol{C}=O(\log n)$



In general

- This works generally for *Δ+1* coloring, maximal independent set, maximal matching, …
- If the problem has locality k, work in G^k.
- The right level of generality: sequential greedy algorithms [Ghaffari, Kuhn, Maus STOC'17]
- Think of the sequential algorithm for *Δ+1* coloring, maximal independent set, or even *ball carving*!

SLOCAL - sequential variant of the **LOCAL** model

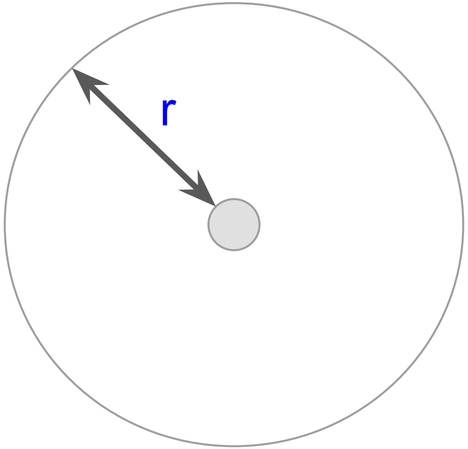
Iterate over nodes in adversarial order.

Decide their label based only on their r-neighbourhood.

SLOCAL - sequential variant of the LOCAL model

Iterate over nodes in adversarial order.

Decide their label based only on their r-neighbourhood.



SLOCAL - sequential variant of the **LOCAL** model

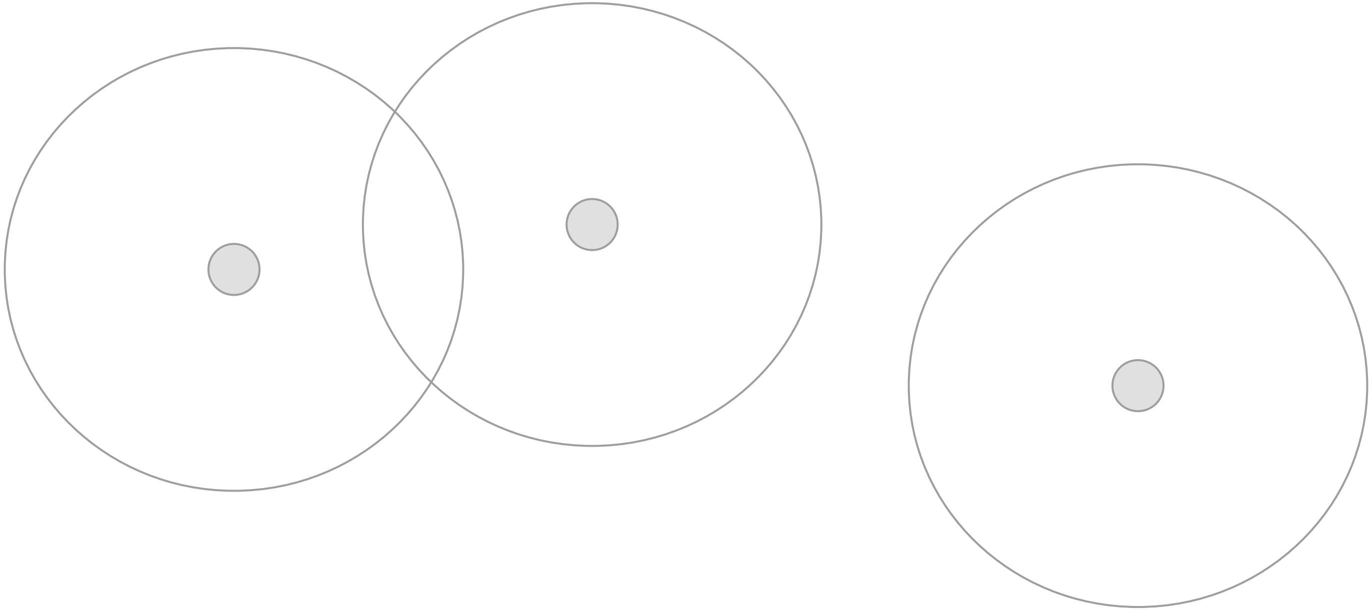
Iterate over nodes in adversarial order.

Decide their label based only on their r-neighbourhood.

SLOCAL - sequential variant of the LOCAL model

Iterate over nodes in adversarial order.

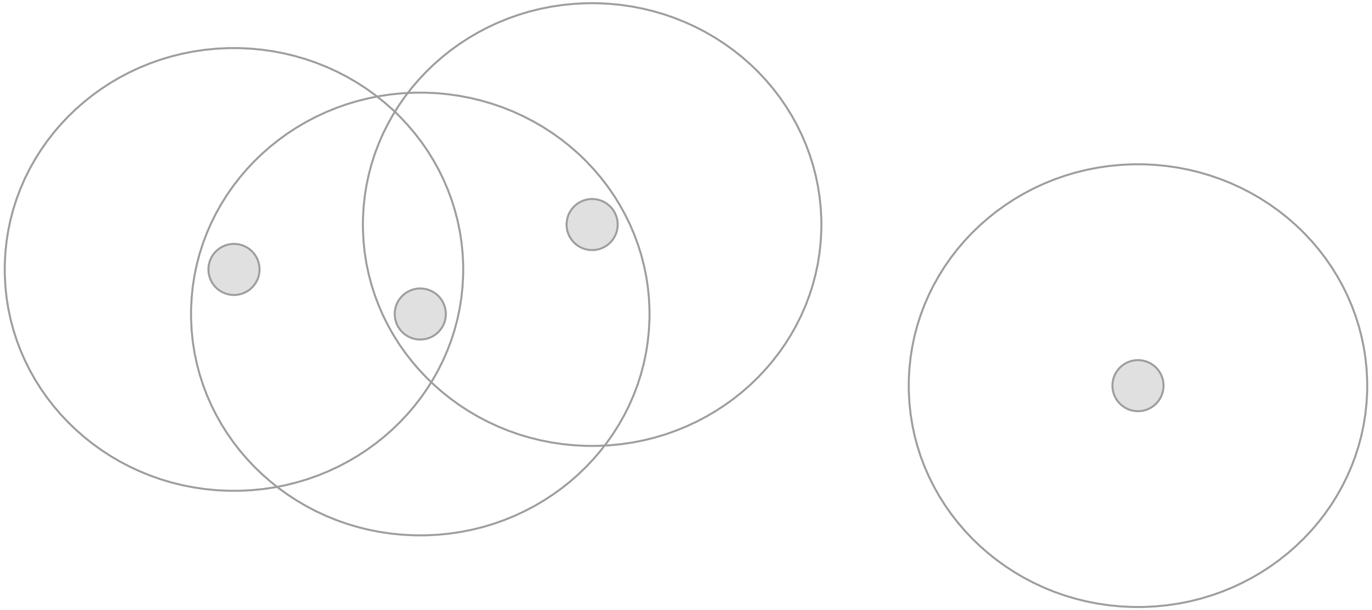
Decide their label based only on their r-neighbourhood.



SLOCAL - sequential variant of the **LOCAL** model

Iterate over nodes in adversarial order.

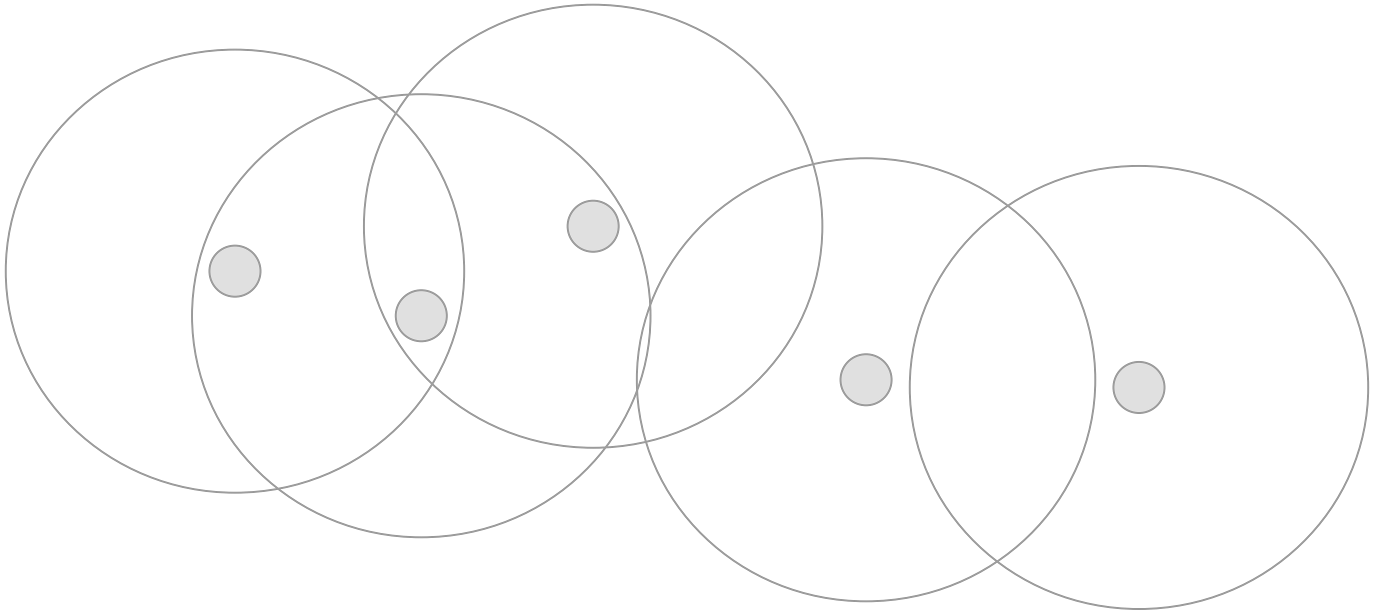
Decide their label based only on their r-neighbourhood.



SLOCAL - sequential variant of the LOCAL model

Iterate over nodes in adversarial order.

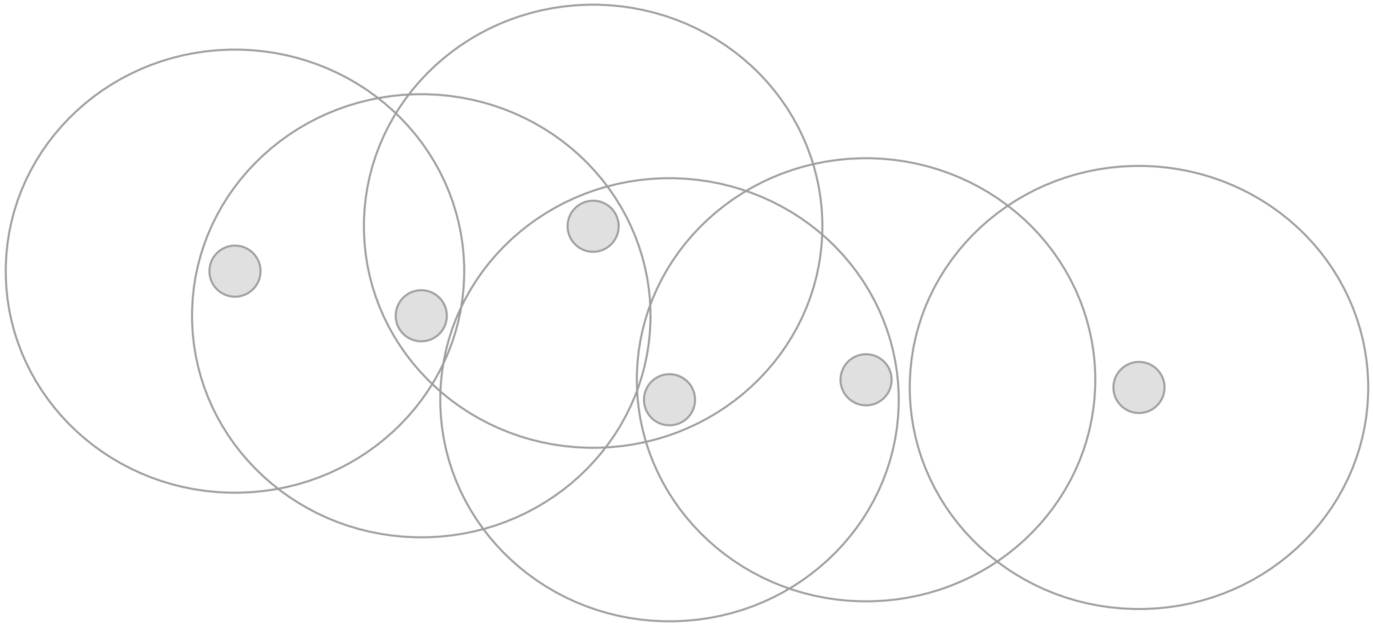
Decide their label based only on their r-neighbourhood.



SLOCAL - sequential variant of the **LOCAL** model

Iterate over nodes in adversarial order.

Decide their label based only on their r-neighbourhood.



P-SLOCAL

"deterministic sequential"

deterministic network decomposition [R., Ghaffari 19+]

direct

P-LOCAL

"deterministic distributed"

P-SLOCAL

"deterministic sequential"

deterministic network decomposition [R., Ghaffari 19+]

P-LOCAL

"deterministic distributed"

Corollary [R., Ghaffari 19+] There is an efficient deterministic algorithm for $\Delta+1$ coloring, maximal independent set, ...

P-SLOCAL

“deterministic sequential”

deterministic network decomposition [R., Ghaffari 19+]

direct

P-LOCAL

“deterministic distributed”

P-RSLOCAL

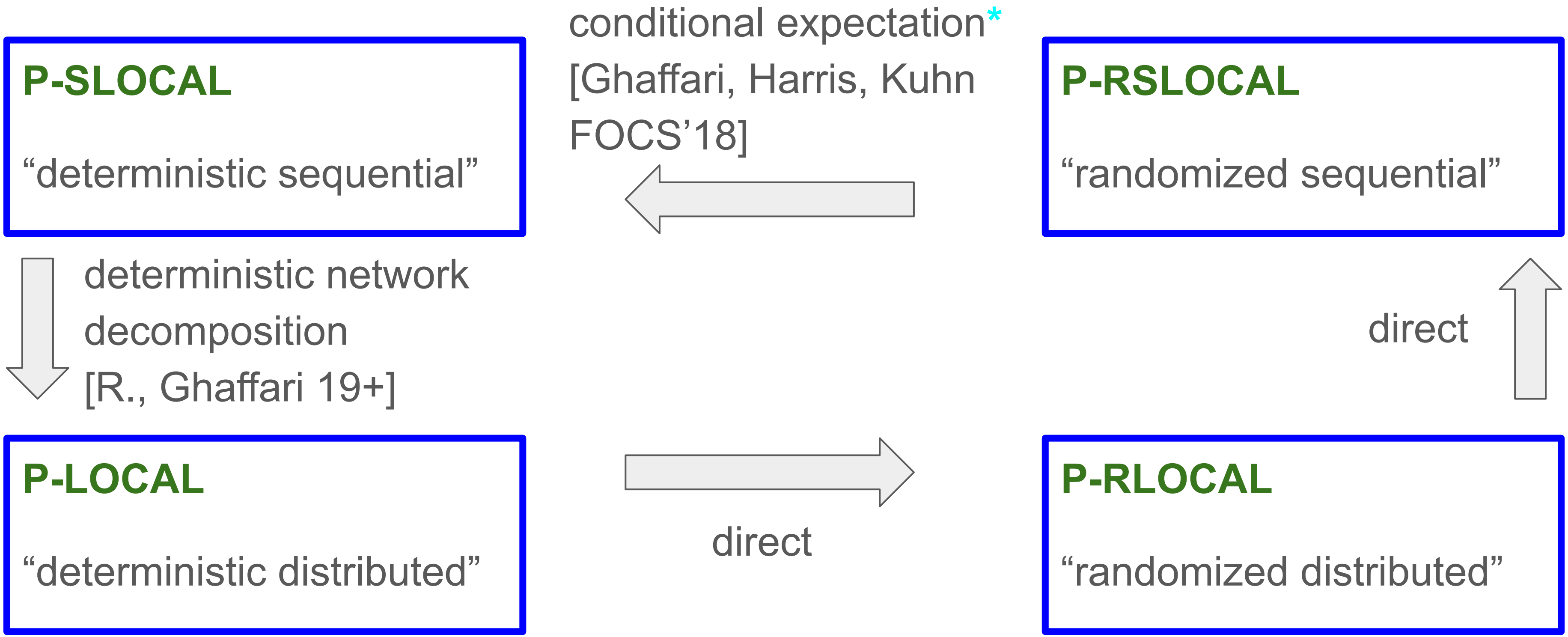
“randomized sequential”

randomized network decomposition [Linial, Saks SODA ‘91]

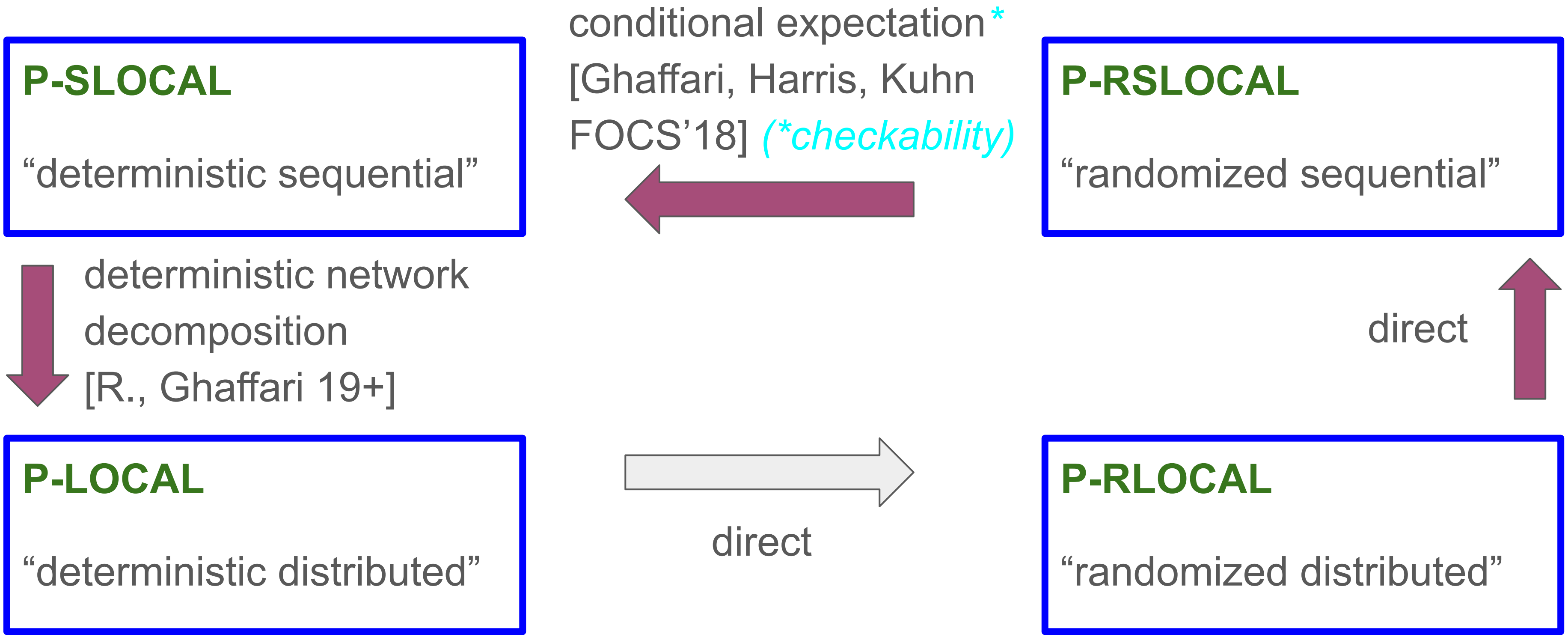
direct

P-RLOCAL

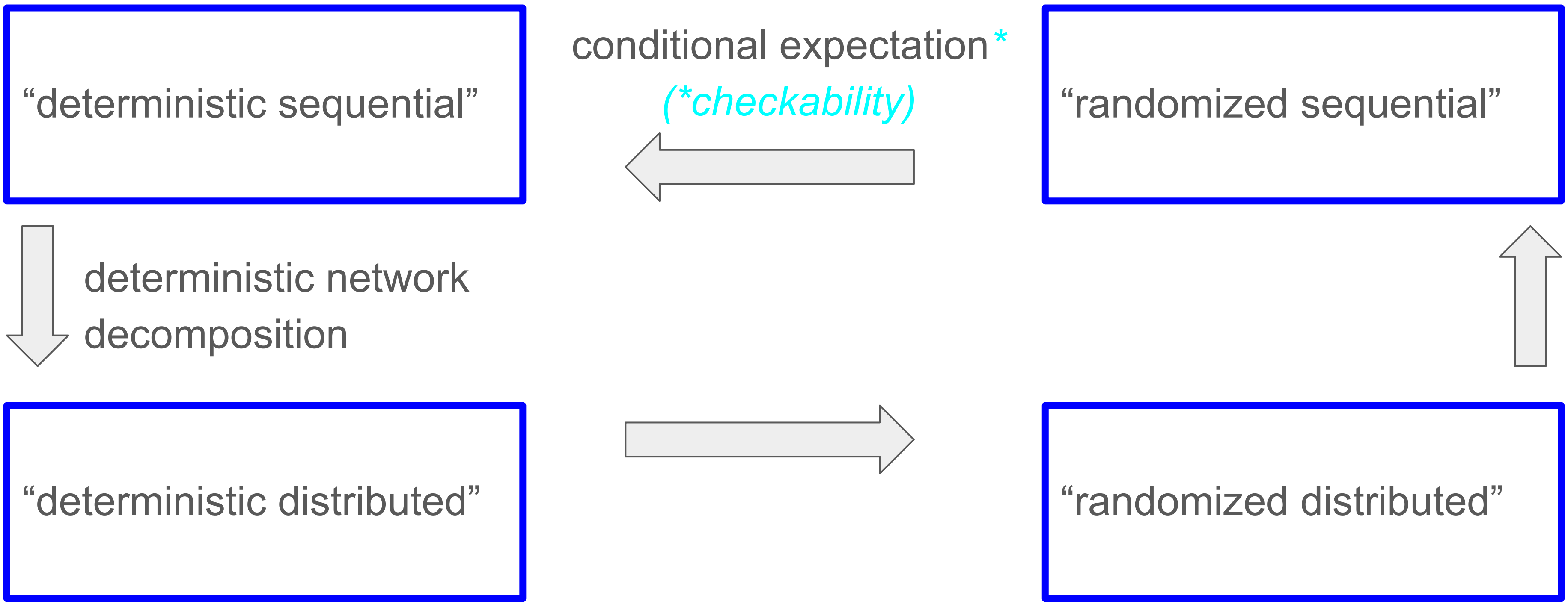
“randomized distributed”



*for problems checkable in poly(log n) rounds



Corollary [R., Ghaffari 19+] There is an efficient deterministic algorithm for hypergraph splitting, Lovász local lemma, ...

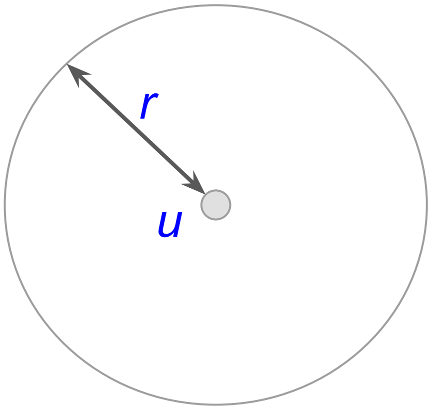


We see a clean first-order theory of the LOCAL model.

Moreover, techniques are simple and principled.

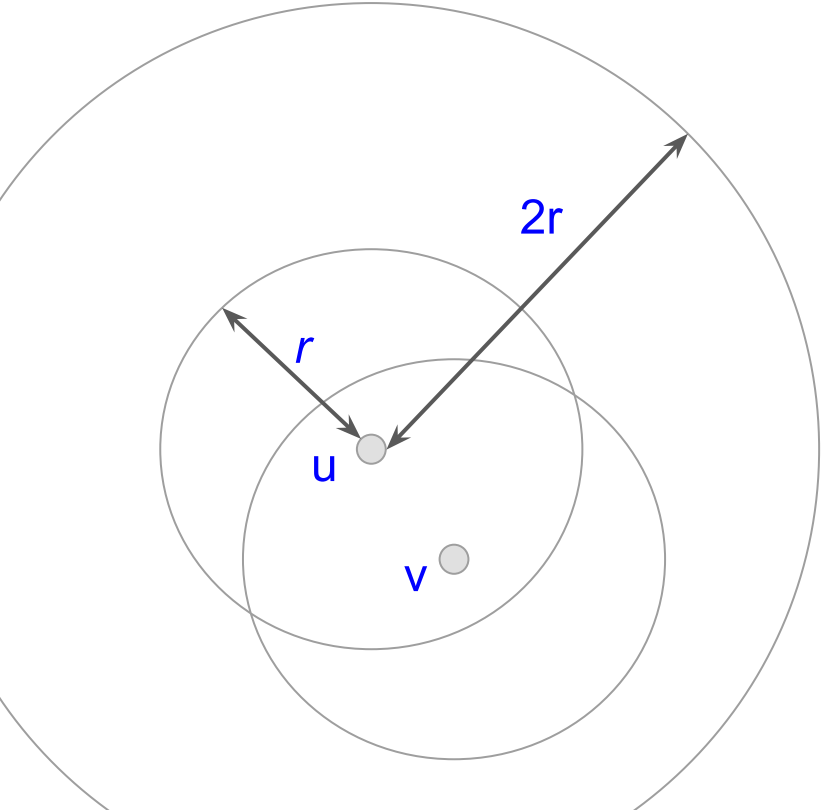
Techniques

Conditional expectation in the **SLOCAL** model



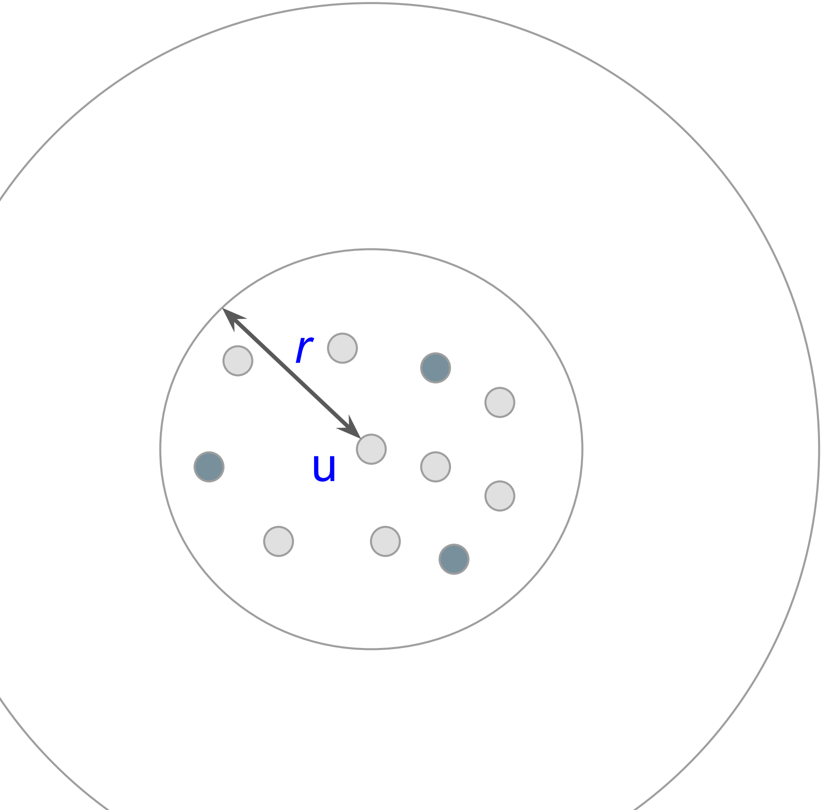
- Original algorithm looks in distance d from the vertex u and can be checked with locality c. In $r=c+d$ rounds we run the algorithm and check for correctness.

Conditional expectation in the SLOCAL model



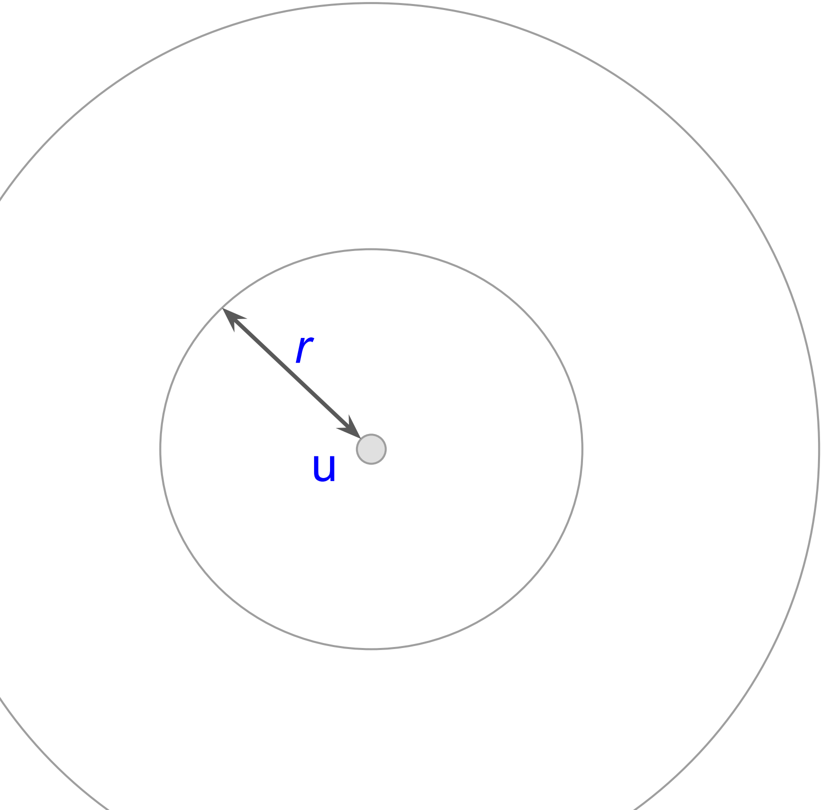
- Original algorithm looks in distance d from the vertex u and can be checked with locality c. In $r=c+d$ rounds we run the algorithm and check for correctness.
- Looking at distance 2r from u, we can compute the probability of failure for any vertex that depends on u.

Conditional expectation in the SLOCAL model



- Original algorithm looks in distance d from the vertex u and can be checked with locality c. In $r=c+d$ rounds we run the algorithm and check for correctness.
- Looking at distance 2r from u, we can compute the probability of failure for any vertex that depends on u.
- For any choice of u's randomness, compute sum of failure probabilities of all these vertices.

Conditional expectation in the **SLOCAL** model



- Original algorithm looks in distance d from the vertex u and can be checked with locality c. In $r=c+d$ rounds we run the algorithm and check for correctness.
- Looking at distance 2r from u, we can compute the probability of failure for any vertex that depends on u.
- For any choice of u's randomness, compute sum of failure probabilities of all these vertices.
- Fix the randomness of u so as to minimize expected sum of failure probabilities; it was $\ll 1$ at the beginning, hence no failure occurs.

Deterministic algorithms for network decomposition

Previous work:

$\boxed{2^{O(\sqrt{\log n \log \log n})} \text{[Awerbuch, Goldberg, Luby, Plotkin FOCS'89]}}$

$2^{O(\sqrt{\log n})}$ [Panconesi, Srinivasan STOC'92]

Deterministic algorithms for network decomposition

Previous work:

$\boxed{2^{O(\sqrt{\log n \log\log n})}\text{ [Awerbuch, Goldberg, Luby, Plotkin FOCS'89]}}$

$2^{O(\sqrt{\log n})}$ [Panconesi, Srinivasan STOC'92]

Our work started with a different algorithm yielding

Deterministic algorithms for network decomposition

Previous work:

$2^{O(\sqrt{\log n \log\log n})}$ [Awerbuch, Goldberg, Luby, Plotkin FOCS'89]

$2^{O(\sqrt{\log n})}$ [Panconesi, Srinivasan STOC'92]

Our work started with a different algorithm yielding

[R., Ghaffari '19+]: *$O(\log^7 n)$* algorithm with *$O(\log^3 n)$* weak diameter and *$O(\log n)$* colors

Deterministic algorithms for network decomposition

Previous work:

$\boxed{2^{O(\sqrt{\log n \log\log n})}\text{ [Awerbuch, Goldberg, Luby, Plotkin FOCS'89]}}$

$2^{O(\sqrt{\log n})}$ [Panconesi, Srinivasan STOC'92]

Our work started with a different algorithm yielding

[R., Ghaffari '19+]: $O(\log^7 n)$ algorithm with $O(\log^3 n)$ weak diameter and $O(\log n)$ colors

Standard reduction: $O(\log^8 n)$ algorithm with $O(\log n)$ strong diameter and $O(\log n)$ colors

Deterministic algorithms for network decomposition

Previous work:

$\boxed{2^{O(\sqrt{\log n \log\log n})} \text{ [Awerbuch, Goldberg, Luby, Plotkin FOCS'89]}}$

$2^{O(\sqrt{\log n})}$ [Panconesi, Srinivasan STOC'92]

Our work started with a different algorithm yielding

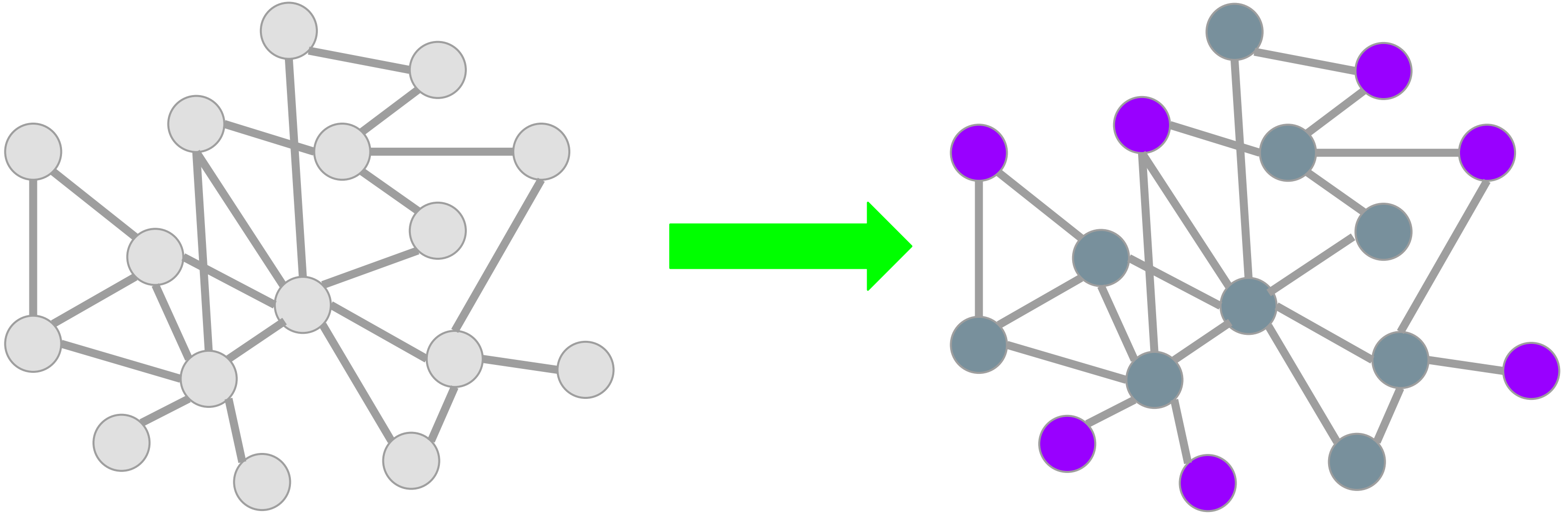
[R., Ghaffari '19+]: $O(\log^7 n)$ algorithm with $O(\log^3 n)$ weak diameter and $O(\log n)$ colors

Standard reduction: $O(\log^8 n)$ algorithm with $O(\log n)$ strong diameter and $O(\log n)$ colors

[Ghaffari, Grunau, R. '19+]: $O(\log^6 n)$ algorithm with $O(\log^2 n)$ weak diameter, $O(\log n)$ colors

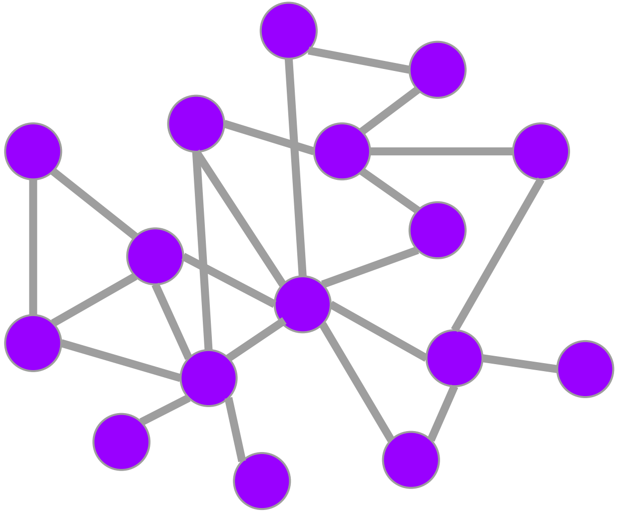
Ruling sets

Goal: find a maximal independent set S such that any vertex of G is of distance at most $O(\log n)$ from S.



Ruling sets

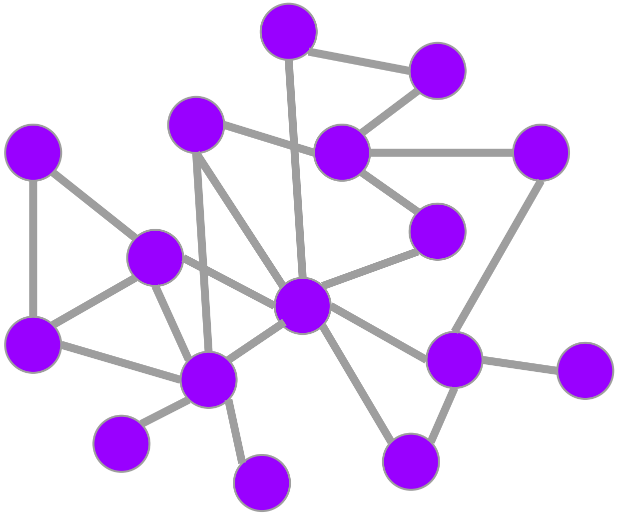
Think about the problem decrementally.



Ruling sets

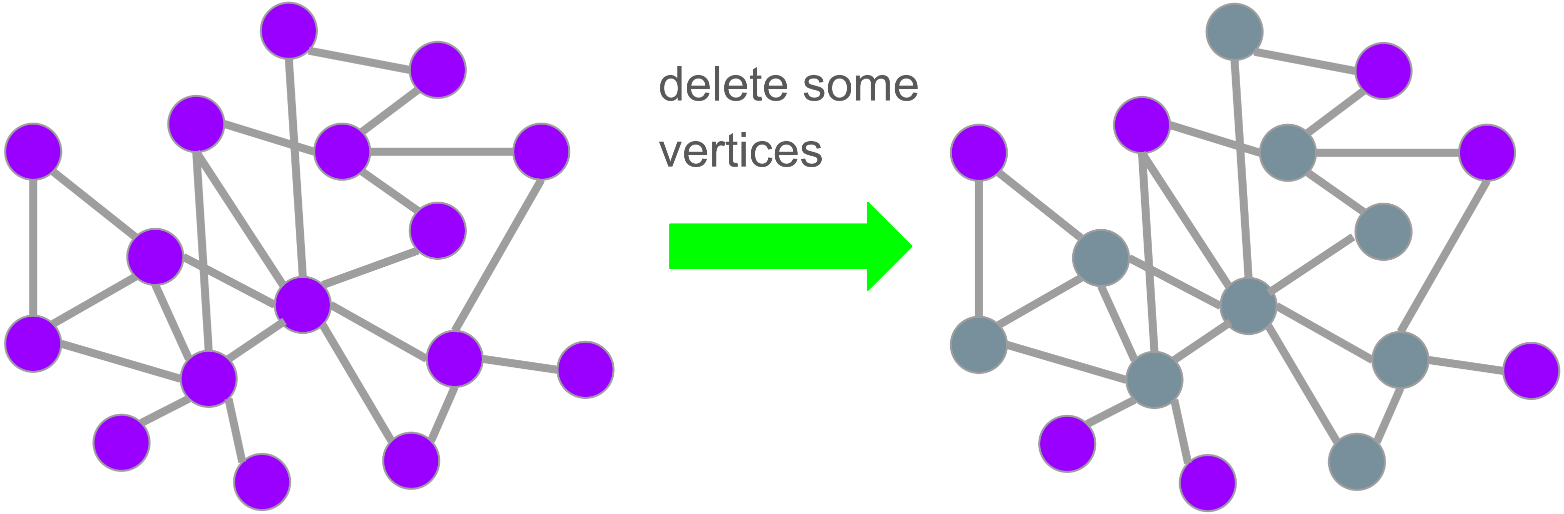
Think about the problem decrementally.

only problem: all edges are "bad"



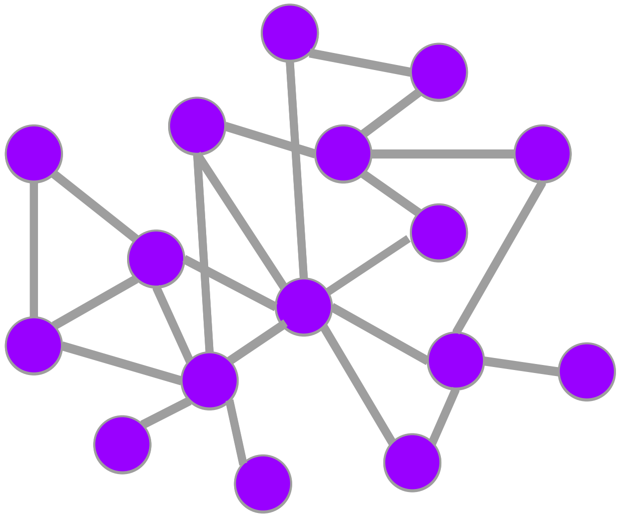
Ruling sets

Think about the problem decrementally.



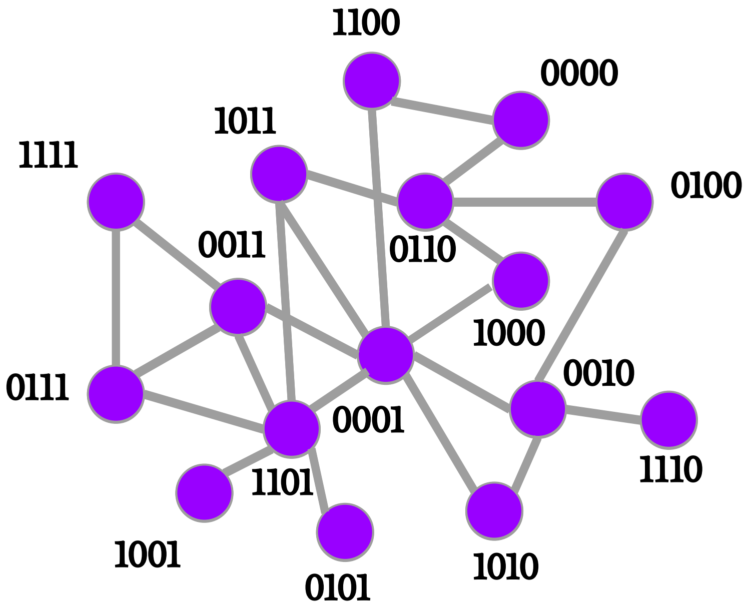
Ruling sets

$O(\log n)$ round algorithm

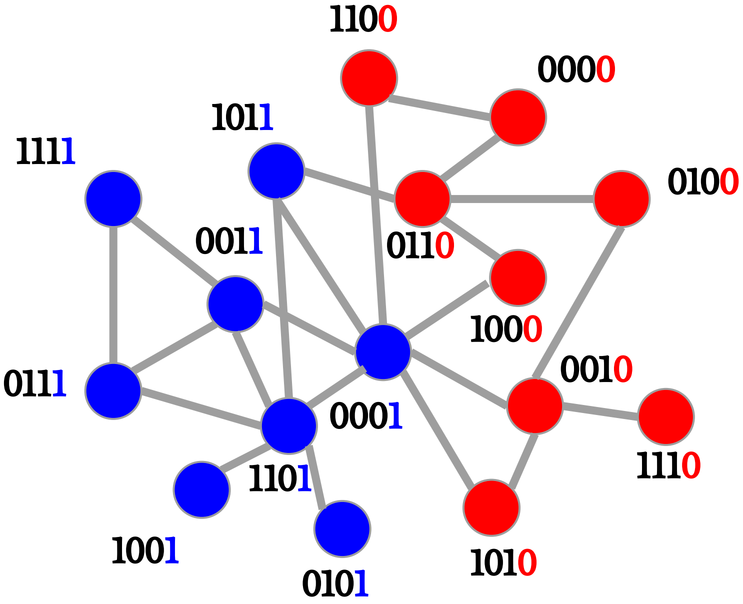


Ruling sets

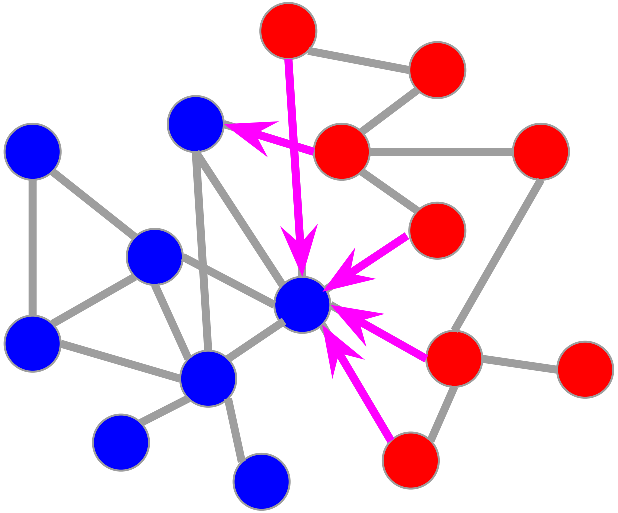
O(log n) bit labels



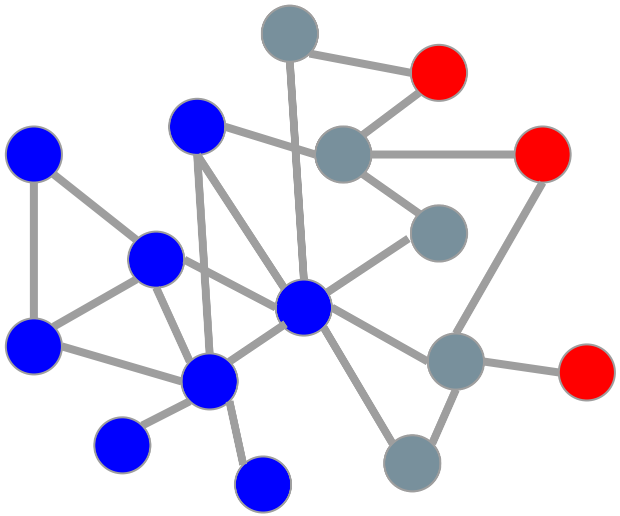
Ruling sets


1100
0000
1011
1111
0100
0011
0110
1000
0010
0111
0001
1110
1101
1001
0101
1010

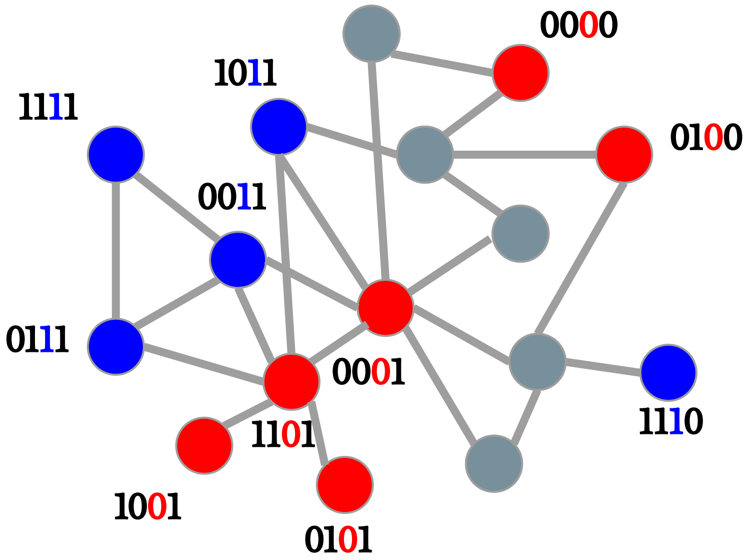
Ruling sets



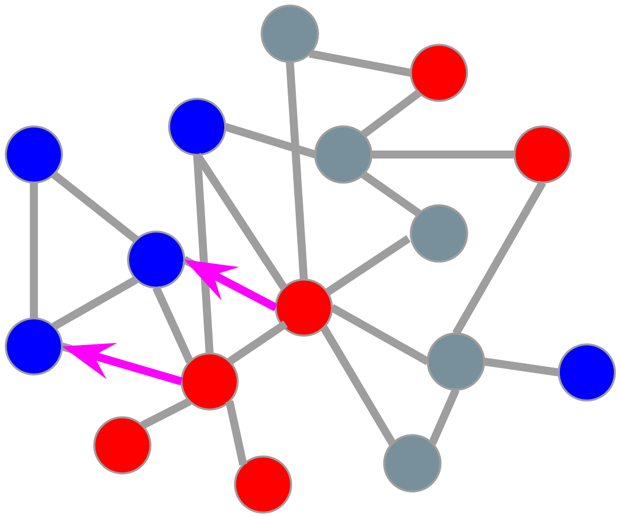
Ruling sets



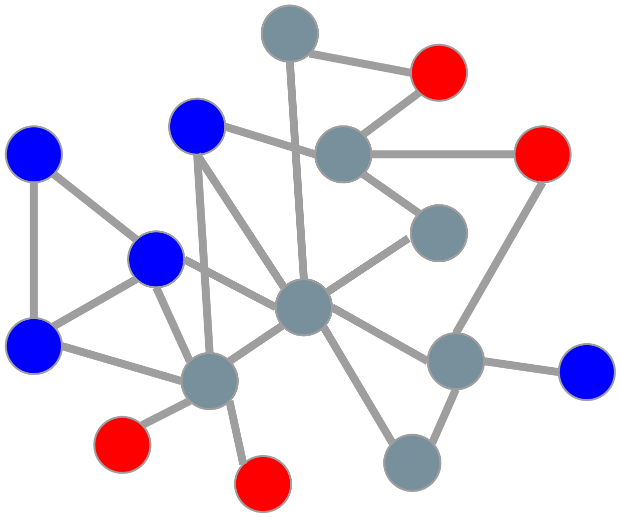
Ruling sets



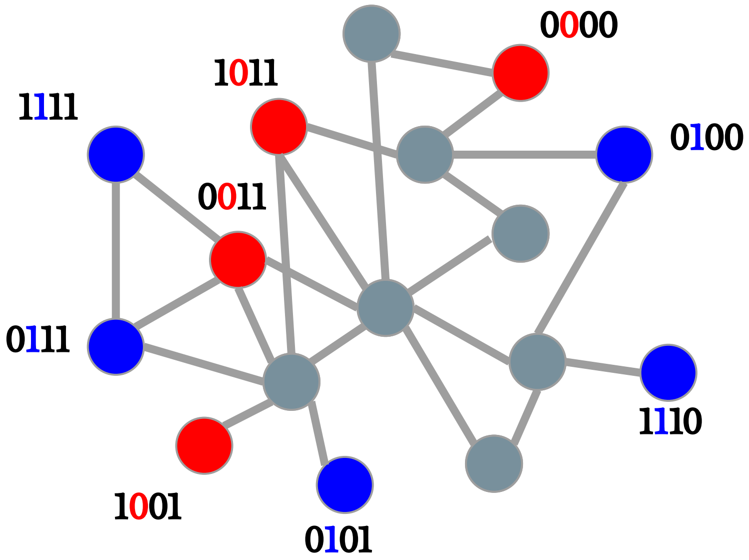
Ruling sets



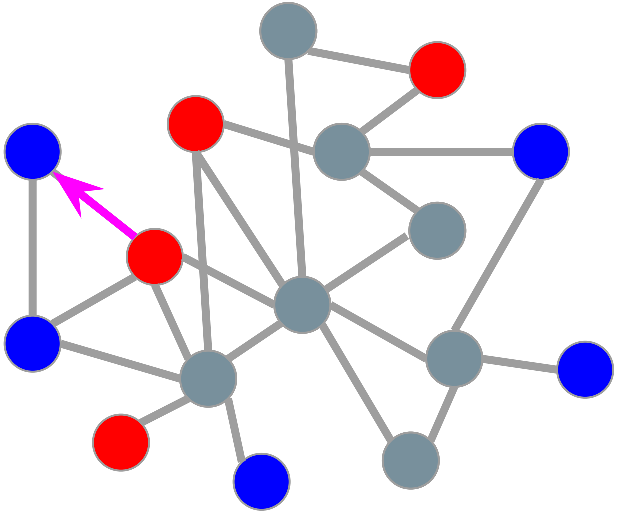
Ruling sets



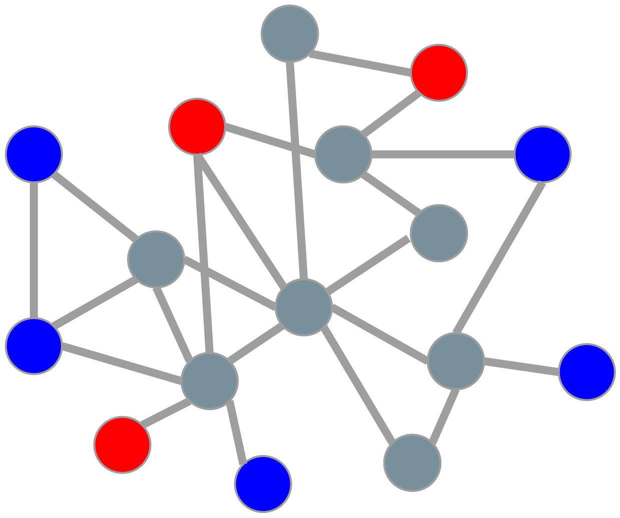
Ruling sets



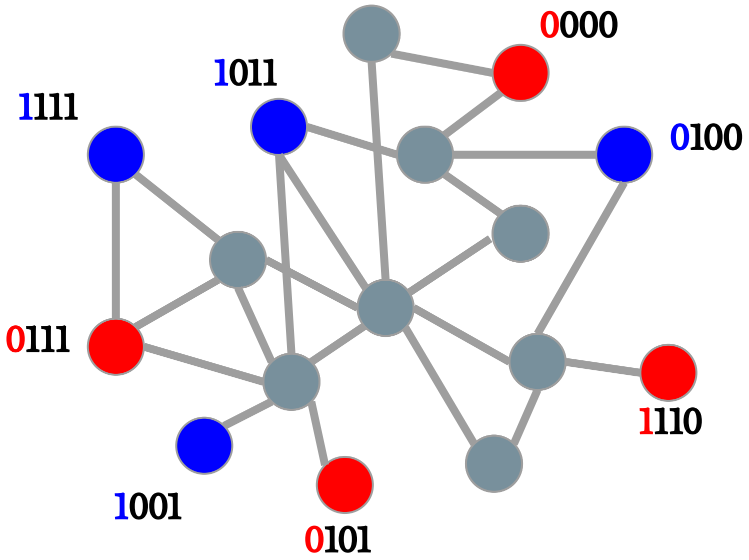
Ruling sets



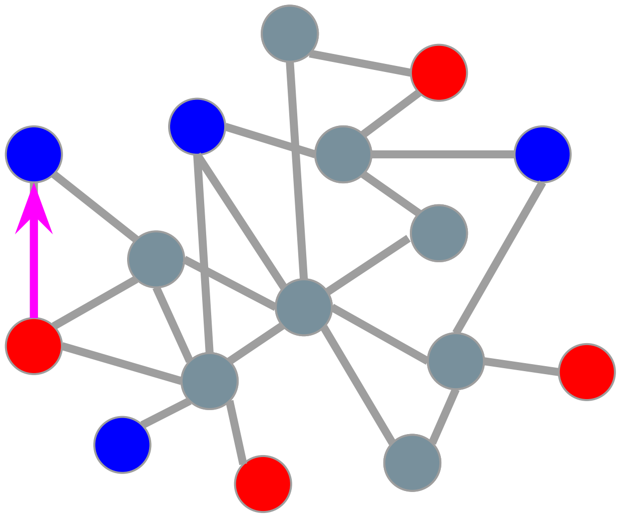
Ruling sets



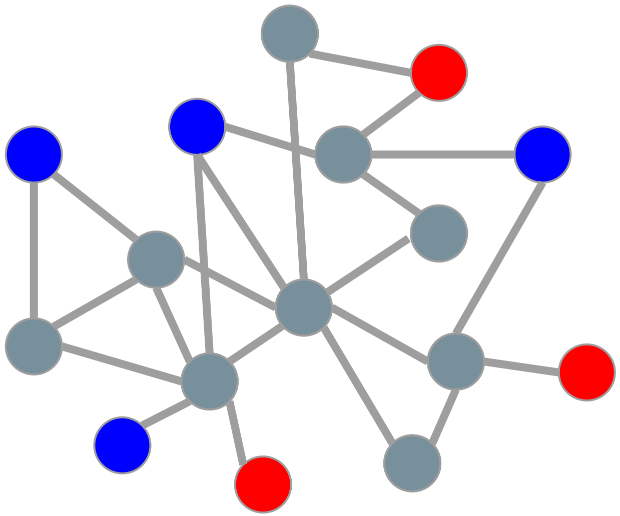
Ruling sets



Ruling sets



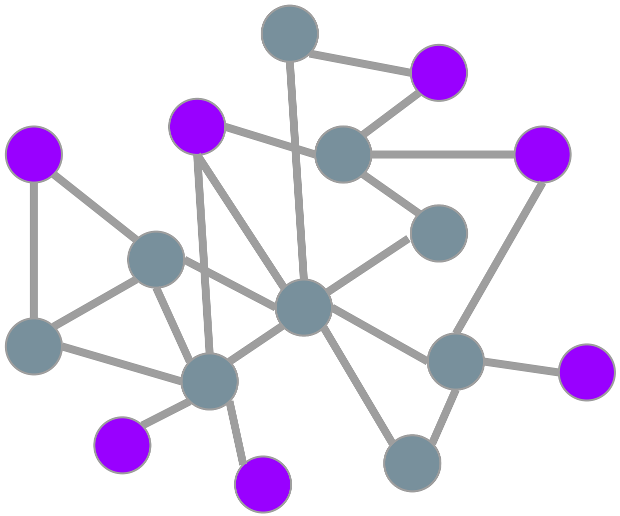
Ruling sets



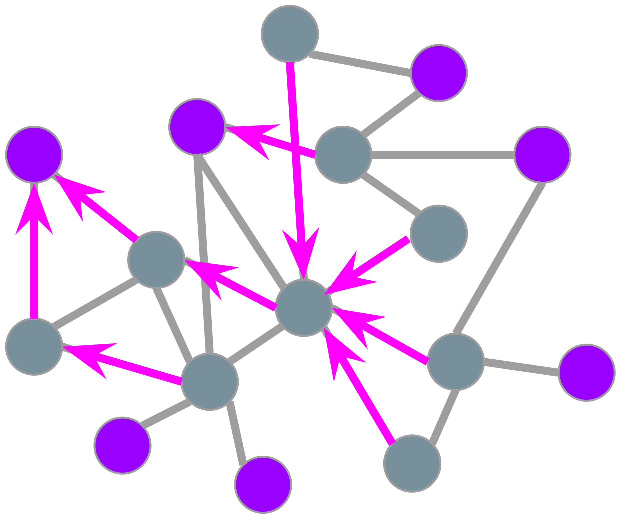
Ruling sets

We got independent set.

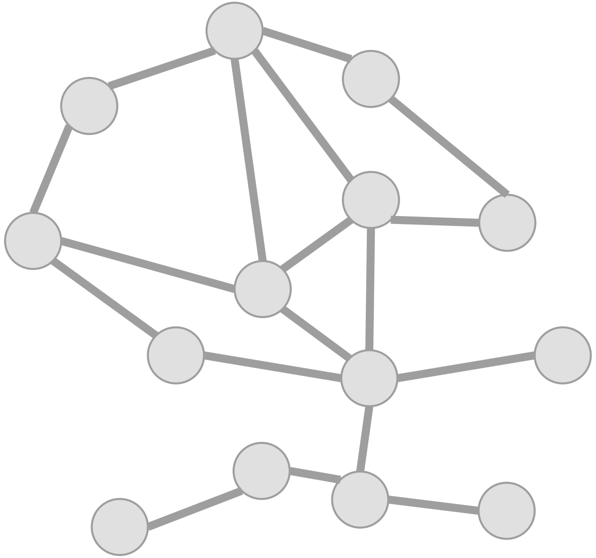
But why are distances to nearest vertex in S of order $O(\log n)$?



Ruling sets



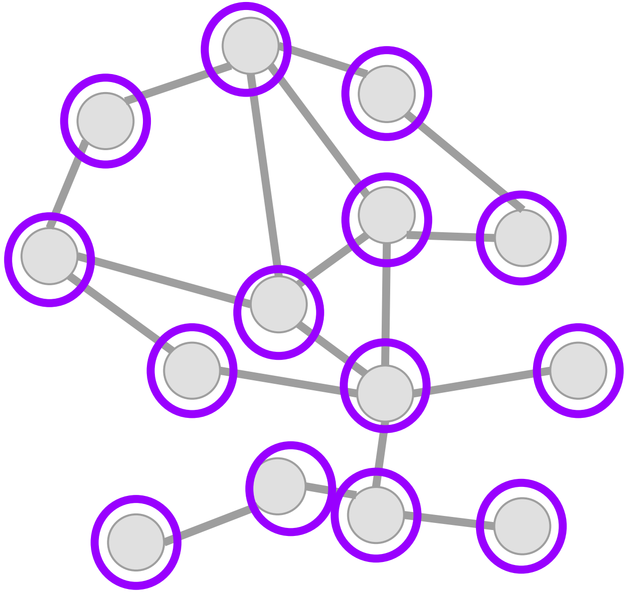
Distributed ball carving



Recall:

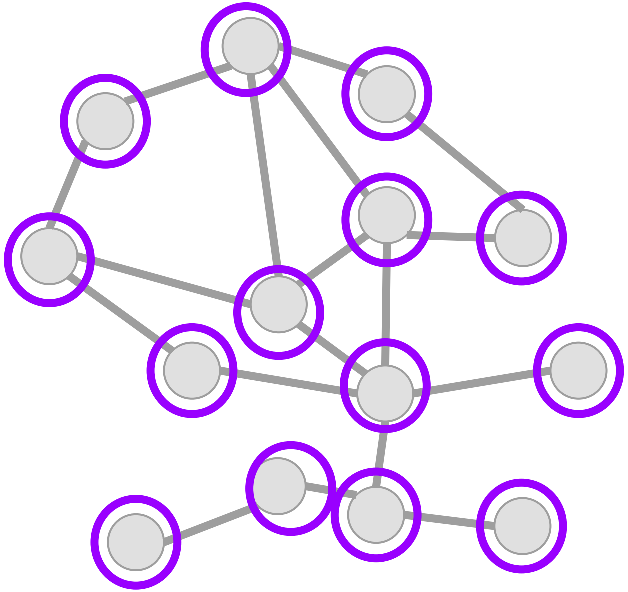
- *poly(log n)* weak-diameter non-neighbouring clusters
- at most ½ fraction of vertices deleted

Distributed ball carving



Each vertex thinks of itself as a root of a cluster

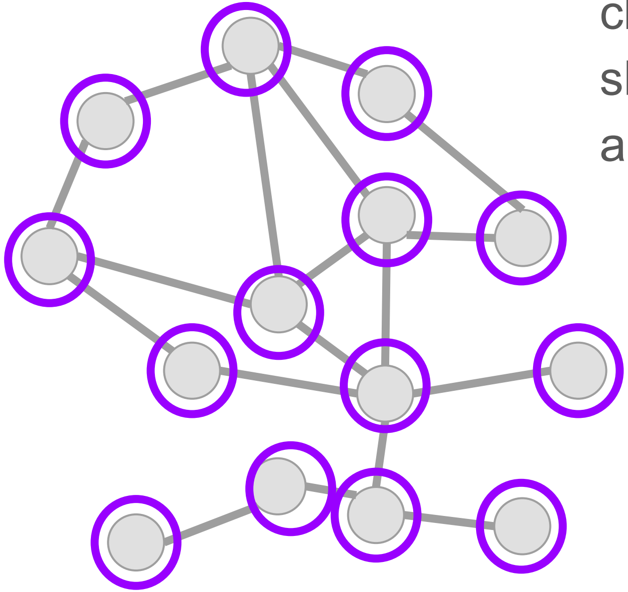
Distributed ball carving



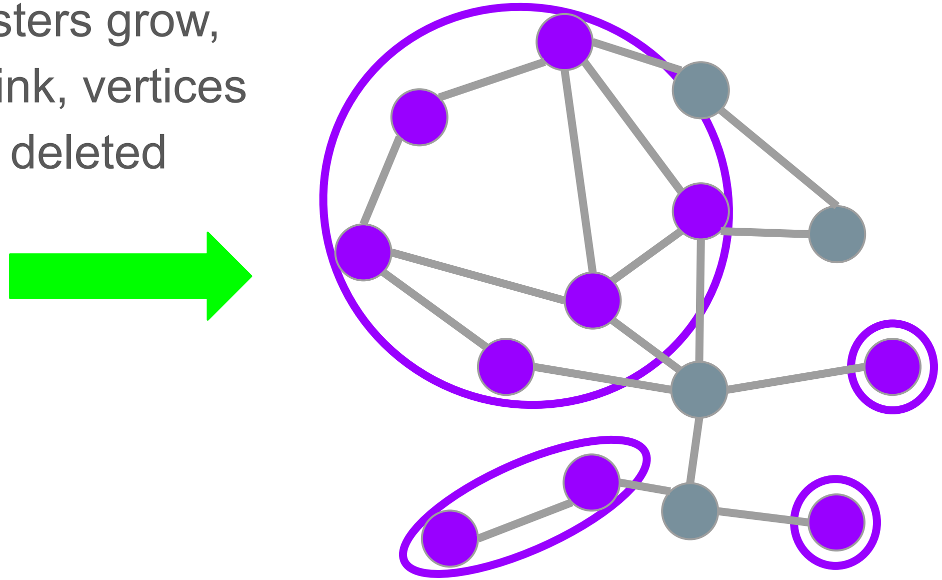
Each vertex thinks of itself as a root of a cluster

only problem: all edges are “bad”

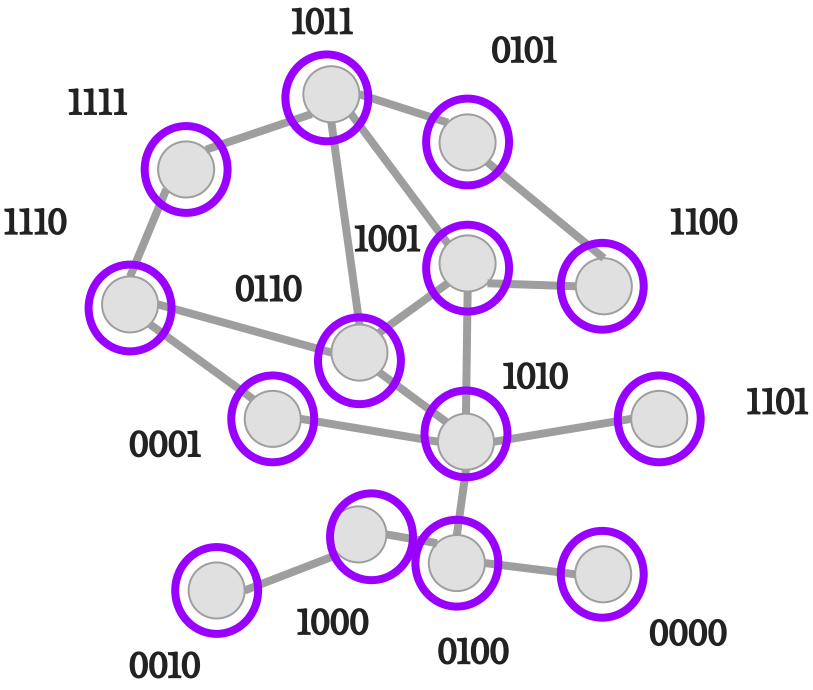
Distributed ball carving



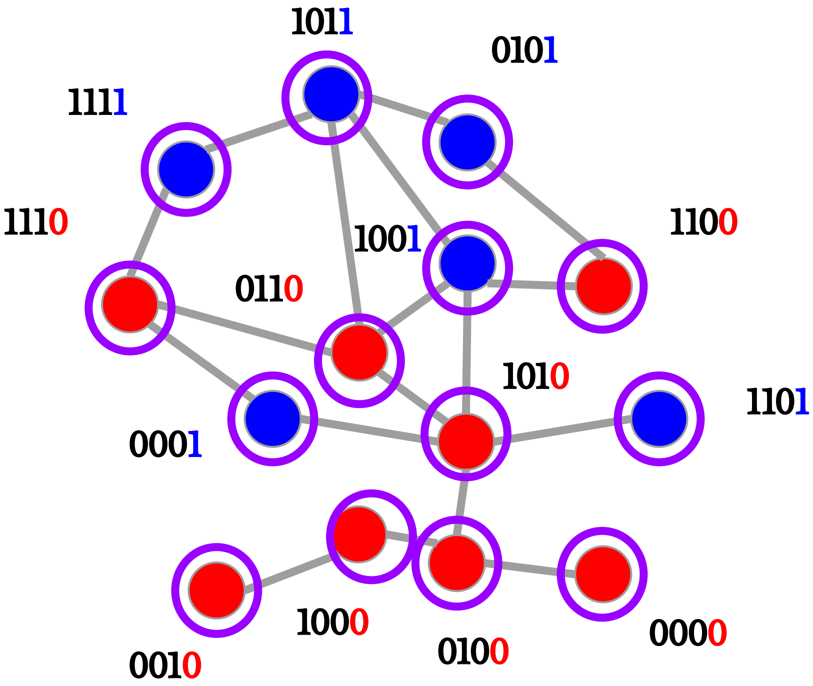
clusters grow,
shrink, vertices
are deleted



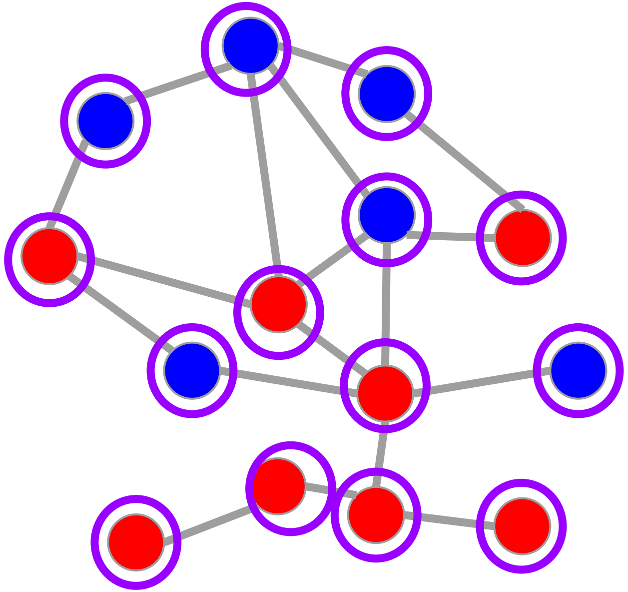
Distributed ball carving



Distributed ball carving



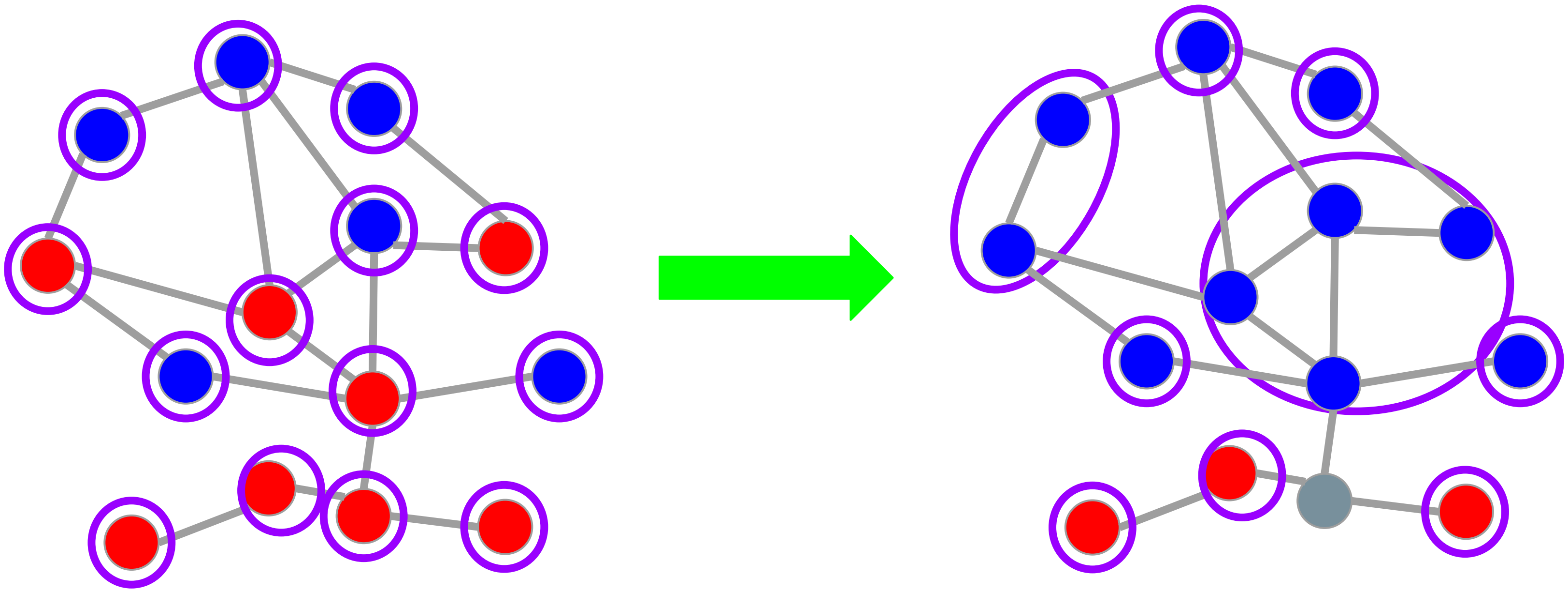
Distributed ball carving



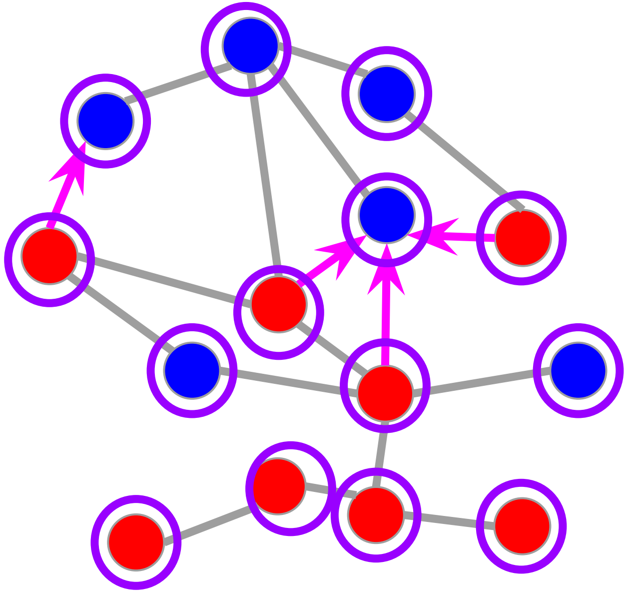
for ruling sets we just delete the boundary, now we have to be more clever

Distributed ball carving

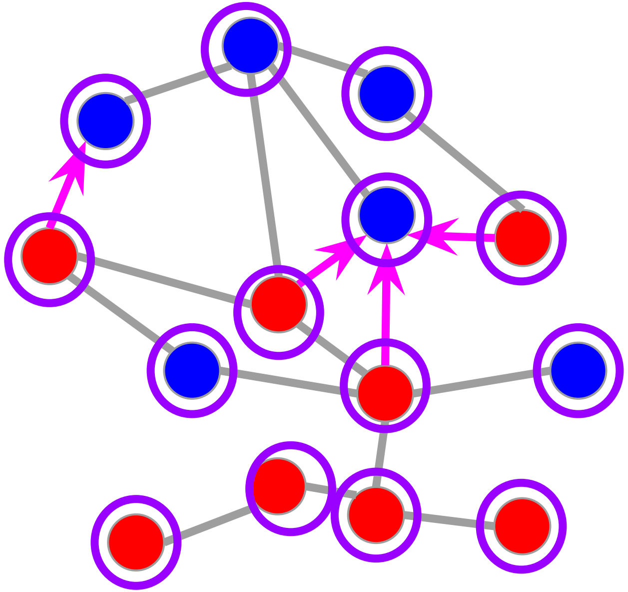
for ruling sets we just delete the boundary,
now we have to be more clever



Distributed ball carving

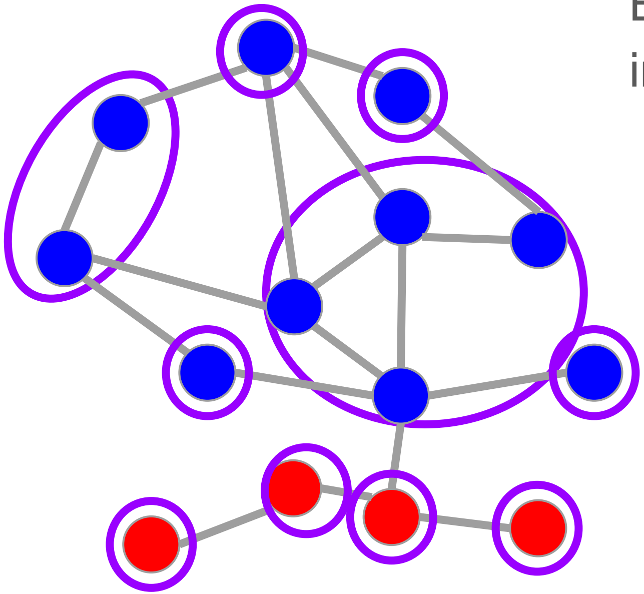


Distributed ball carving



It is time to go back to the sequential algorithm.

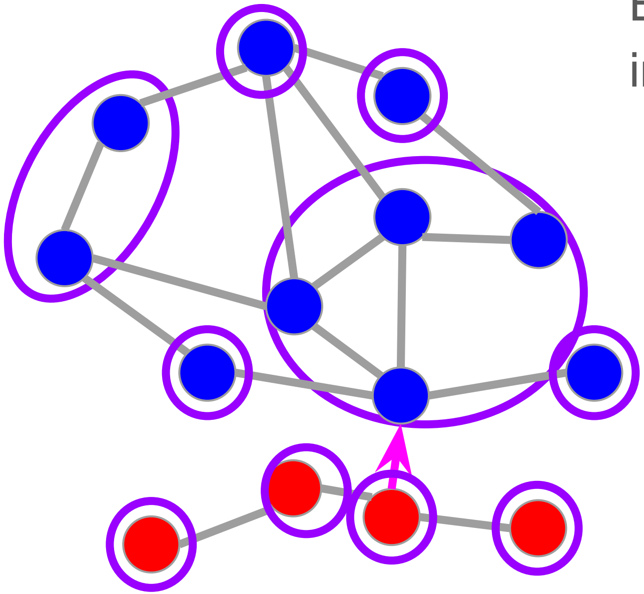
Distributed ball carving



It is time to go back to the sequential algorithm.

Blue clusters grow, if it means multiplicative increase by factor of $1+1/O(\log n)$.

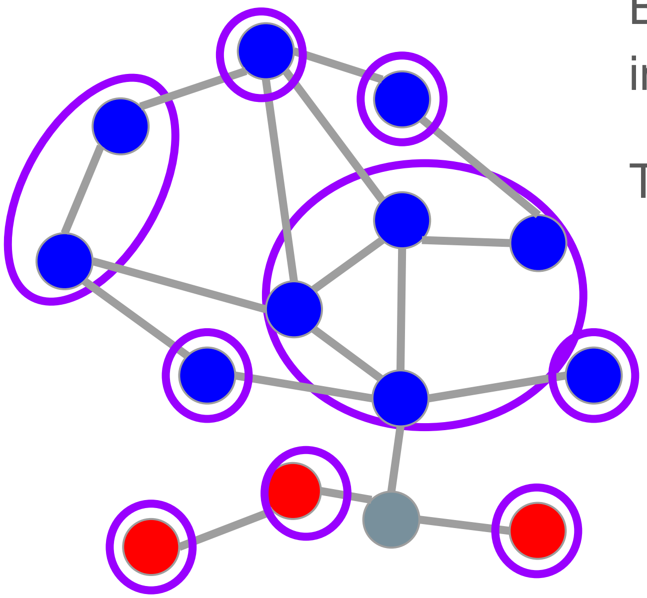
Distributed ball carving



It is time to go back to the sequential algorithm.

Blue clusters grow, if it means multiplicative increase by factor of *$1+1/O(\log n)$.*

Distributed ball carving

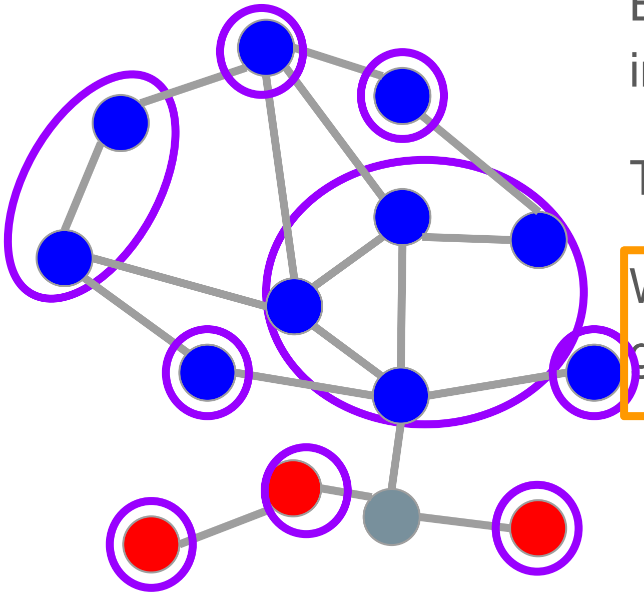


It is time to go back to the sequential algorithm.

Blue clusters grow, if it means multiplicative increase by factor of $1+1/O(\log n)$.

They delete their boundary otherwise.

Distributed ball carving



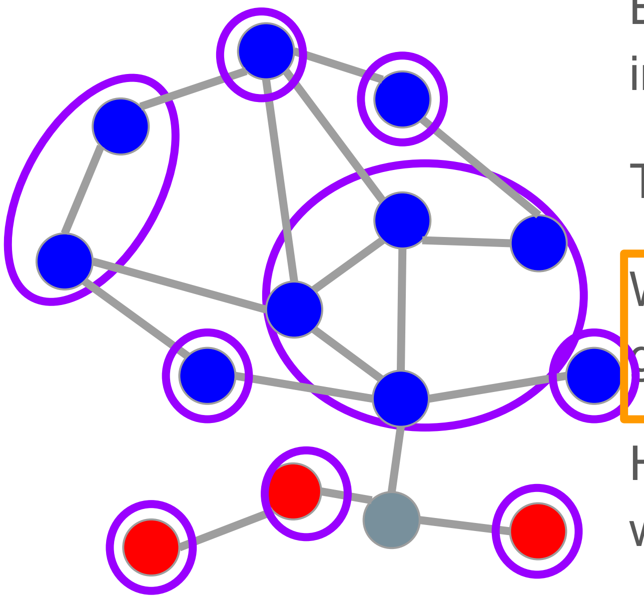
It is time to go back to the sequential algorithm.

Blue clusters grow, if it means multiplicative increase by factor of $1+1/O(\log n)$.

They delete their boundary otherwise.

When blue cluster stops growing, it will never grow again during this phase.

Distributed ball carving



It is time to go back to the sequential algorithm.

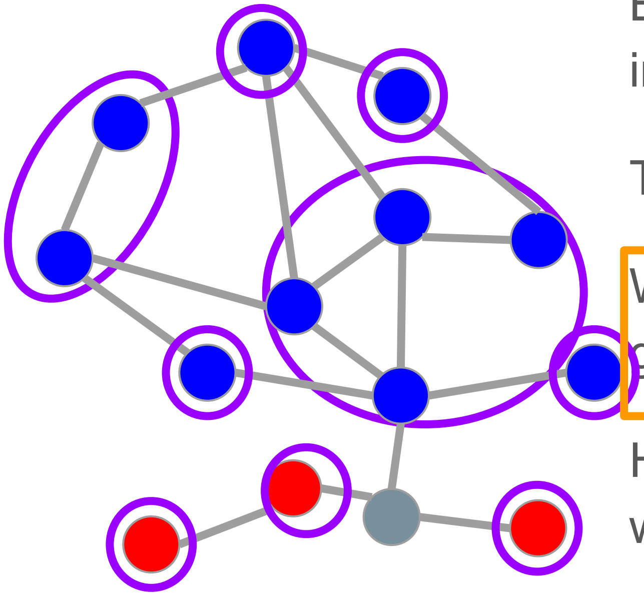
Blue clusters grow, if it means multiplicative increase by factor of $1+1/O(\log n)$.

They delete their boundary otherwise.

When blue cluster stops growing, it will never grow again during this phase.

Hence, if a cluster grows after $\Omega(\log^2 n)$ steps, it was growing for the whole time ⇒ Its size is $>n$.

Distributed ball carving



It is time to go back to the sequential algorithm.

Blue clusters grow, if it means multiplicative increase by factor of $1+1/O(\log n)$.

They delete their boundary otherwise.

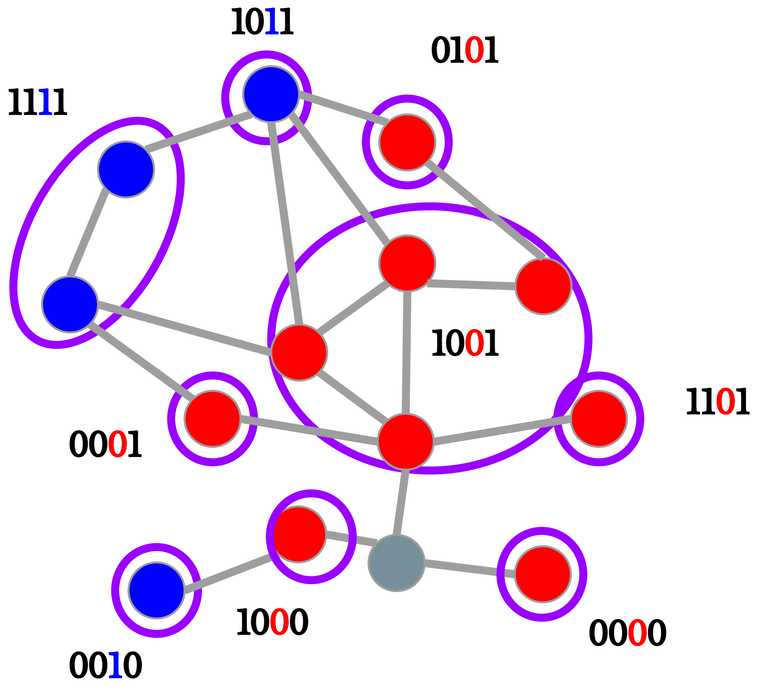
When blue cluster stops growing, it will never grow again during this phase.

Hence, if a cluster grows after $\Omega(\log^2 n)$ steps, it was growing for the whole time ⇒ Its size is $>n$.

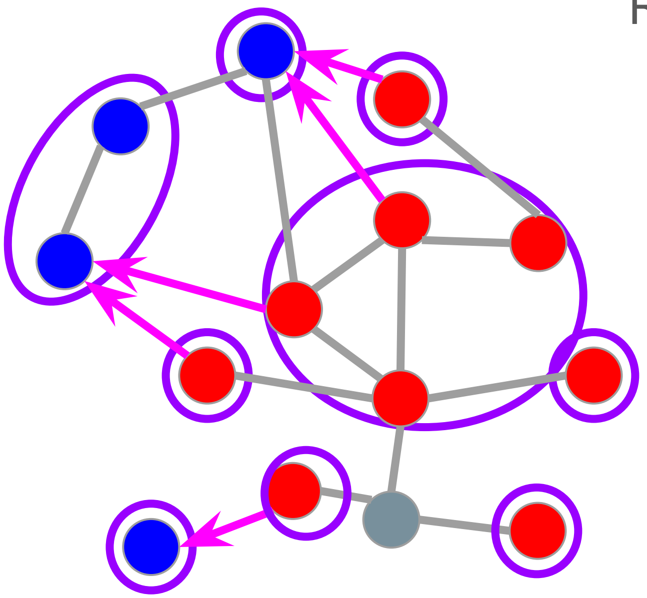
We finish after $O(\log^2 n)$ steps.

Distributed ball carving

Id of a cluster is the id of the original vertex.



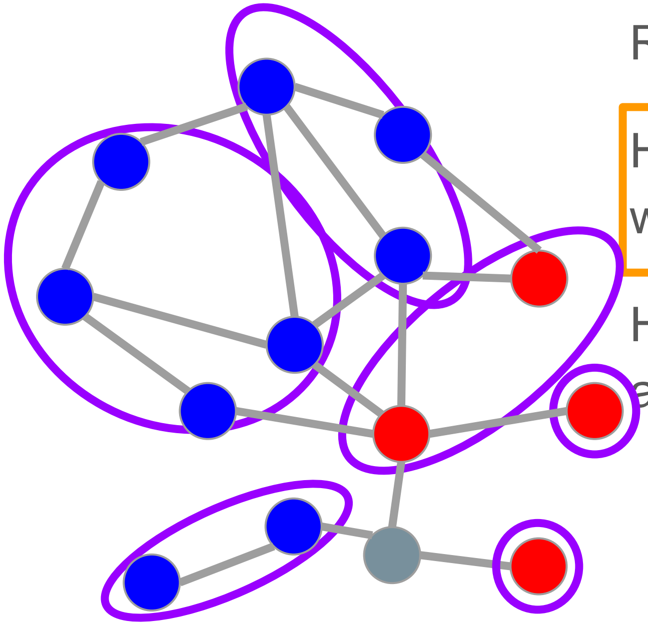
Distributed ball carving



Id of a cluster is the id of the original vertex.

Red vertices propose, not clusters.

Distributed ball carving



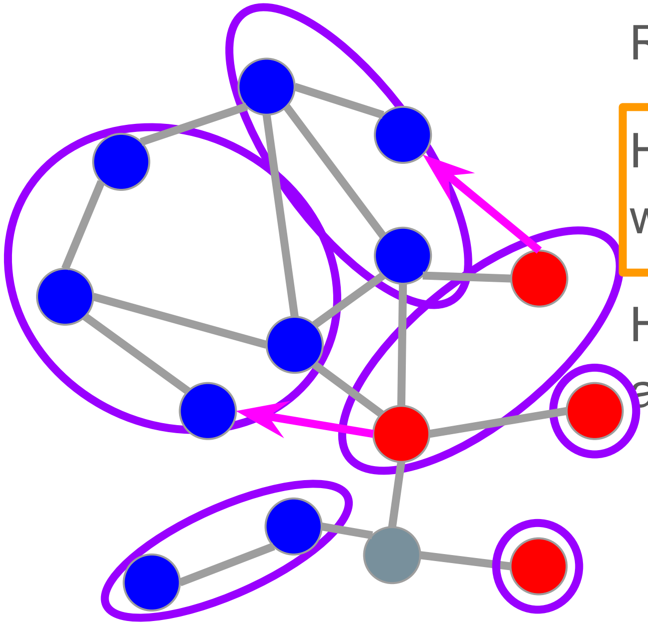
Id of a cluster is the id of the original vertex.

Red vertices propose, not clusters.

Hence, clusters may even disconnect, but weak-diameter increases by at most 2

Hence, the diameter of each cluster grows additively by $O(\log^2 n)$ per phase.

Distributed ball carving



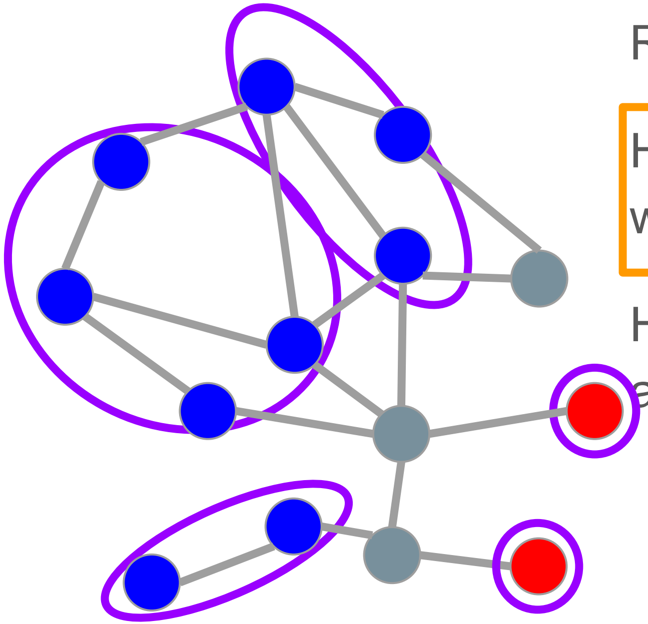
Id of a cluster is the id of the original vertex.

Red vertices propose, not clusters.

Hence, clusters may even disconnect, but weak-diameter increases by at most 2

Hence, the diameter of each cluster grows additively by $O(\log^2 n)$ per phase.

Distributed ball carving



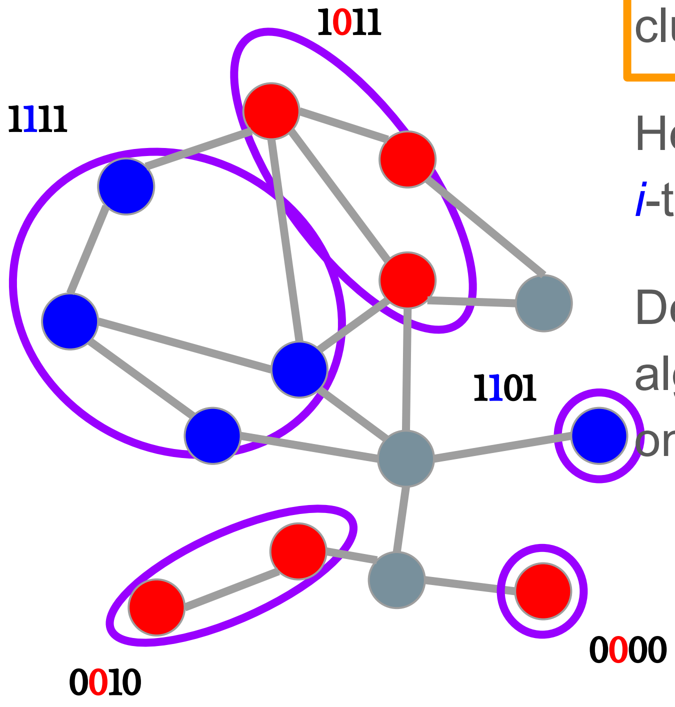
Id of a cluster is the id of the original vertex.

Red vertices propose, not clusters.

Hence, clusters may even disconnect, but weak-diameter increases by at most 2

Hence, the diameter of each cluster grows additively by $O(\log^2 n)$ per phase.

Distributed ball carving

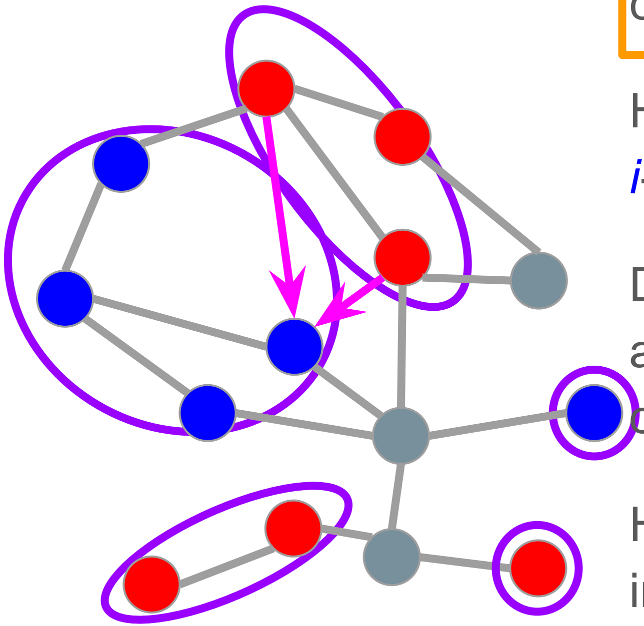


After *i*-th phase, there is no edge between two clusters with different *i*-th bit in their label.

Hence, all connected components agree on the *i*-th bit.

Deleted vertices stay delete for the whole algorithm; hence, all connected component agree on *i* rightmost bits after *i* phases.

Distributed ball carving



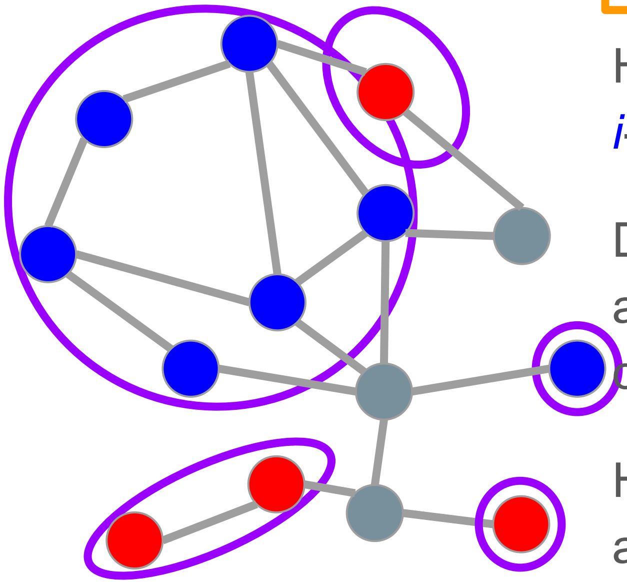
After *i*-th phase, there is no edge between two clusters with different *i*-th bit in their label.

Hence, all connected components agree on the *i*-th bit.

Deleted vertices stay delete for the whole algorithm; hence, all connected component agree on *i* rightmost bits after *i* phases.

Hence, after we finish all remaining clusters are independent.

Distributed ball carving



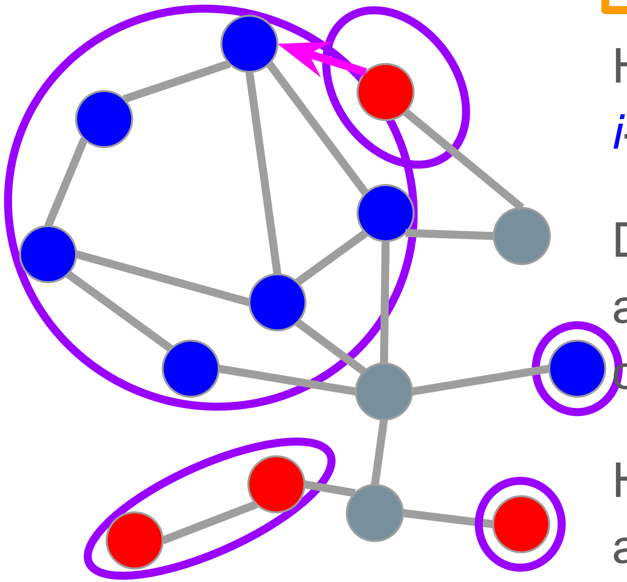
After i-th phase, there is no edge between two clusters with different i-th bit in their label.

Hence, all connected components agree on the i-th bit.

Deleted vertices stay delete for the whole algorithm; hence, all connected component agree on i rightmost bits after i phases.

Hence, after $O(\log n)$ phases remaining clusters are independent.

Distributed ball carving



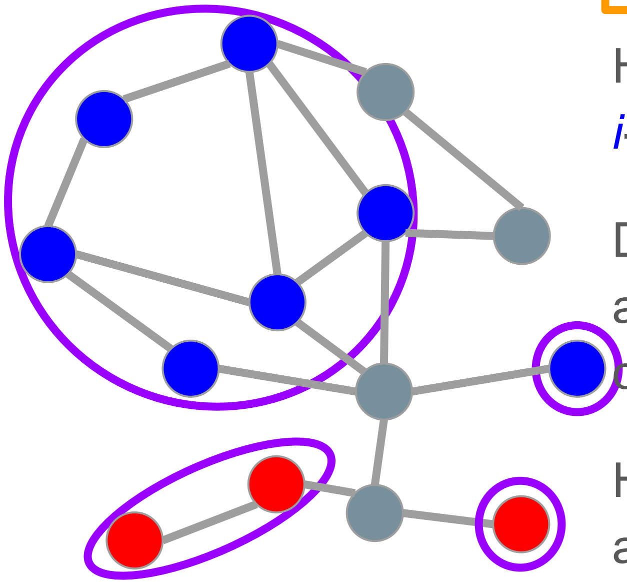
After *i*-th phase, there is no edge between two clusters with different *i*-th bit in their label.

Hence, all connected components agree on the *i*-th bit.

Deleted vertices stay delete for the whole algorithm; hence, all connected component agree on *i* rightmost bits after *i* phases.

Hence, after $O(\log n)$ phases remaining clusters are independent.

Distributed ball carving



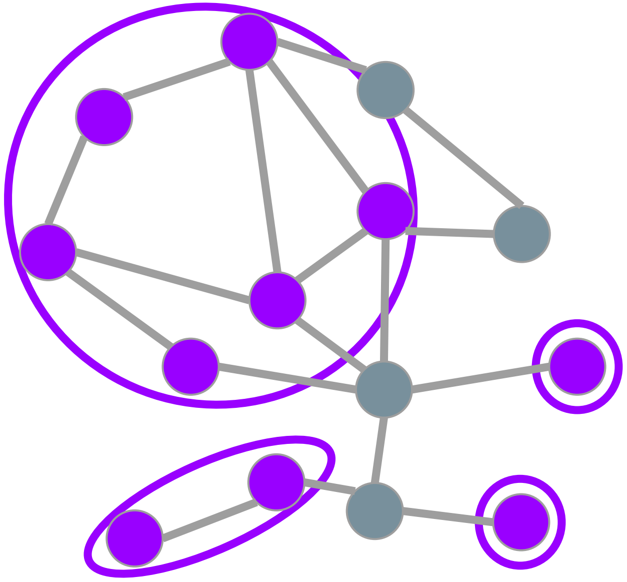
After *i*-th phase, there is no edge between two clusters with different *i*-th bit in their label.

Hence, all connected components agree on the *i*-th bit.

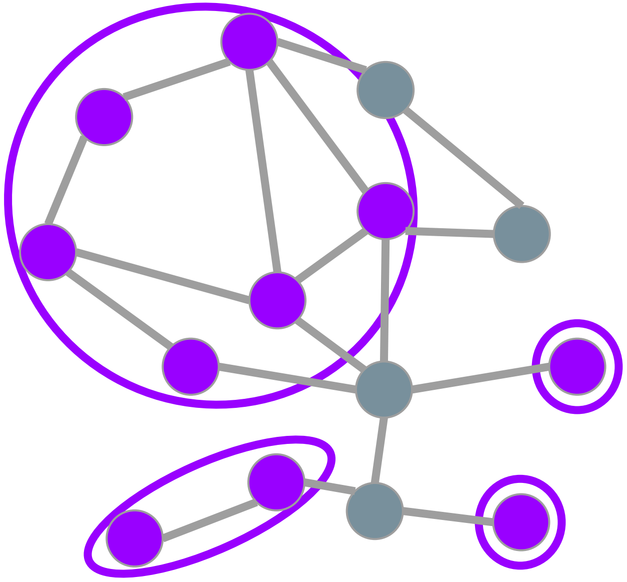
Deleted vertices stay delete for the whole algorithm; hence, all connected component agree on *i* rightmost bits after *i* phases.

Hence, after *O(*log *n)* phases remaining clusters are independent.

Distributed ball carving

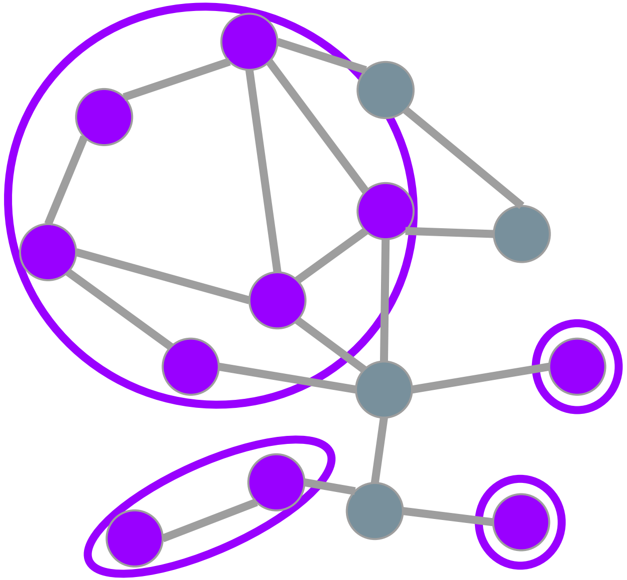


Distributed ball carving



In each of $O(\log n)$ phases, we deleted $1/O(\log n)$ fraction of vertices.

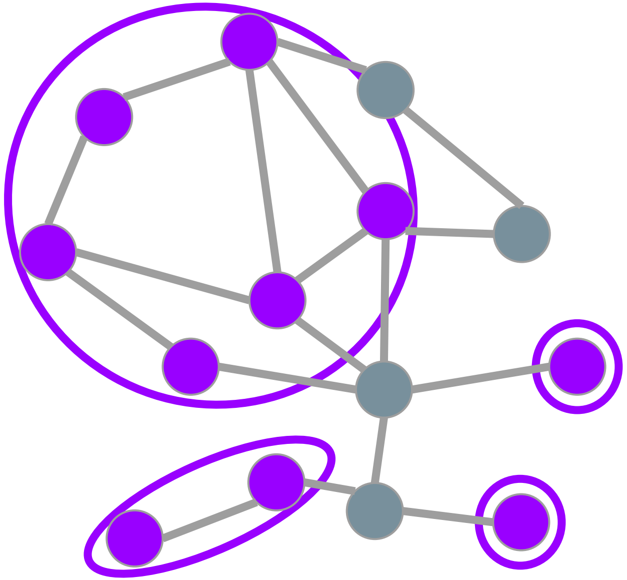
Distributed ball carving



In each of $O(\log n)$ phases, we deleted $1/O(\log n)$ fraction of vertices.

Hence, in total we deleted at most ½ of the vertices (if we set up right constant).

Distributed ball carving

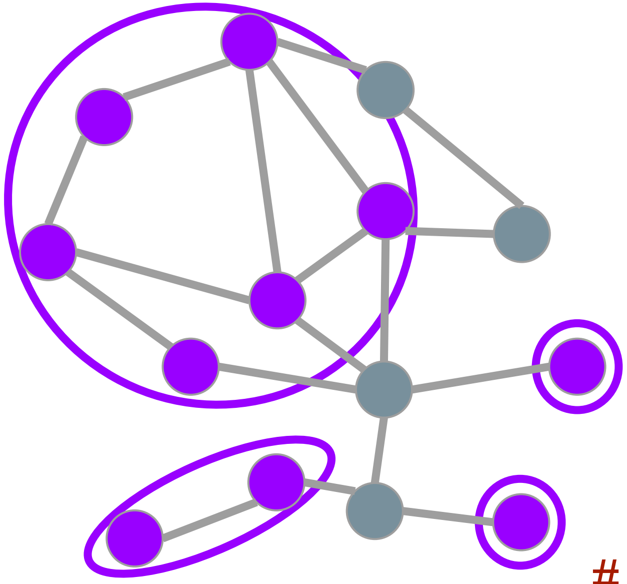


In each of $O(\log n)$ phases, we deleted $1/O(\log n)$ fraction of vertices.

Hence, in total we deleted at most ½ of the vertices (if we set up right constant).

Each phase took $O(\log^2 n)$ steps, hence, diameter of each cluster is $O(\log^3 n)$.

Distributed ball carving



In each of $O(\log n)$ phases, we deleted $1/O(\log n)$ fraction of vertices.

Hence, in total we deleted at most ½ of the vertices (if we set up right constant).

Each phase took $O(\log^2 n)$ steps, hence, diameter of each cluster is $O(\log^3 n)$.

Running time is $O(\log^7 n) =$

$\underbrace{O(\log n)}_{\text{\# of colors}} \cdot \underbrace{O(\log n)}_{\text{\# of phases}} \cdot \underbrace{O(\log^2 n)}_{\text{steps per phase}} \cdot \underbrace{O(\log^3 n)}_{\text{complexity of one step}}$

Outlook to **CONGEST** and randomized **LOCAL**/**MPC**

Our algorithm in **CONGEST**

Since our algorithm works also in the **CONGEST** model, we get some more results:

[Censor-Hillel, Parter, Schwartzman DISC‘17]:

“There is deterministic poly(log n)-round algorithm for maximal independent set in the **CONGEST** model. “

[Bamberger, Kuhn, Maus ‘19+]

“The same holds for Δ+1 coloring. “

Important: the algorithm also gives you underlying broadcast trees.

Our algorithm in **CONGEST**

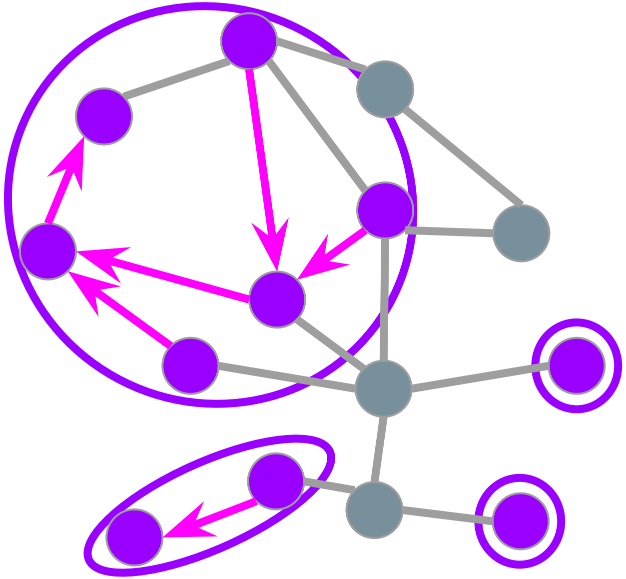
The algorithm generates broadcast trees.

Each edge is in $O(\log n)$ of them.

Our algorithm in **CONGEST**

The algorithm generates broadcast trees.

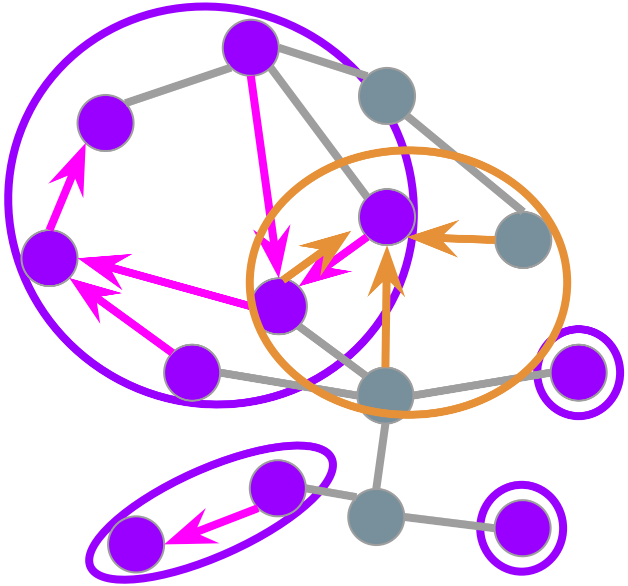
Each edge is in $O(\log n)$ of them.



Our algorithm in **CONGEST**

The algorithm generates broadcast trees.

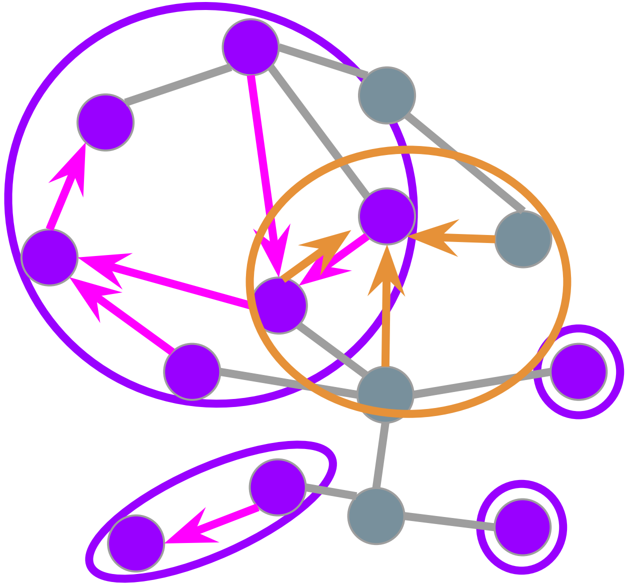
Each edge is in $O(\log n)$ of them.



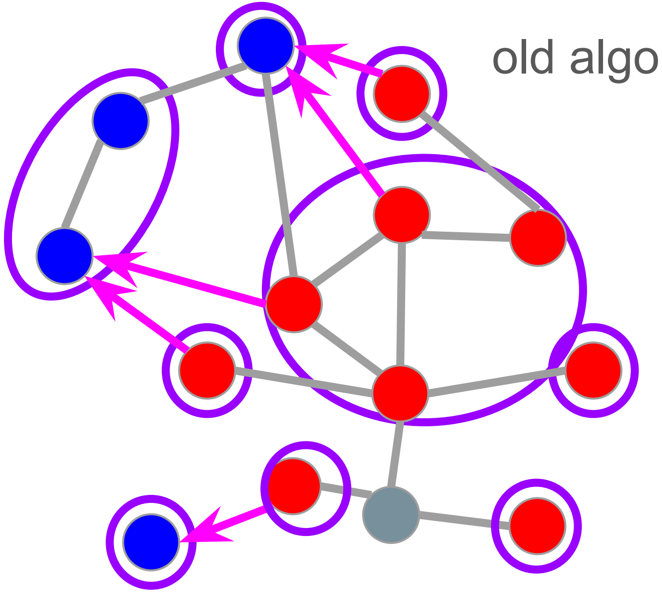
Our algorithm in **CONGEST**

The algorithm generates broadcast trees.

Each edge is in $O(\log n)$ of them.



Natural extension of the algorithm gives decomposition of G^k.

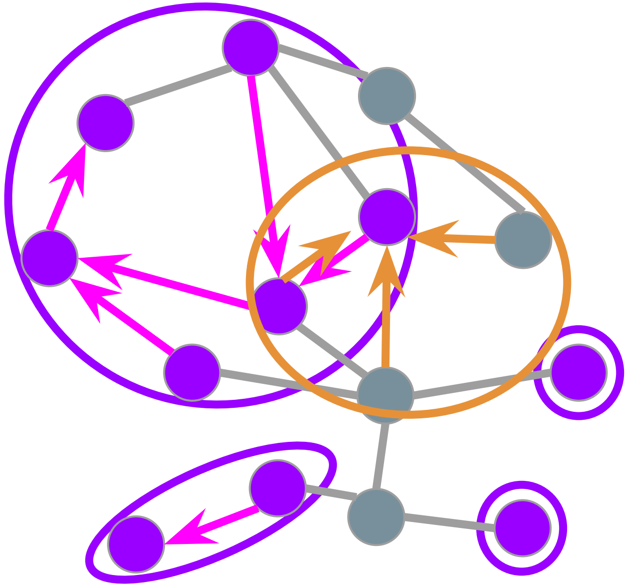


old algorithm

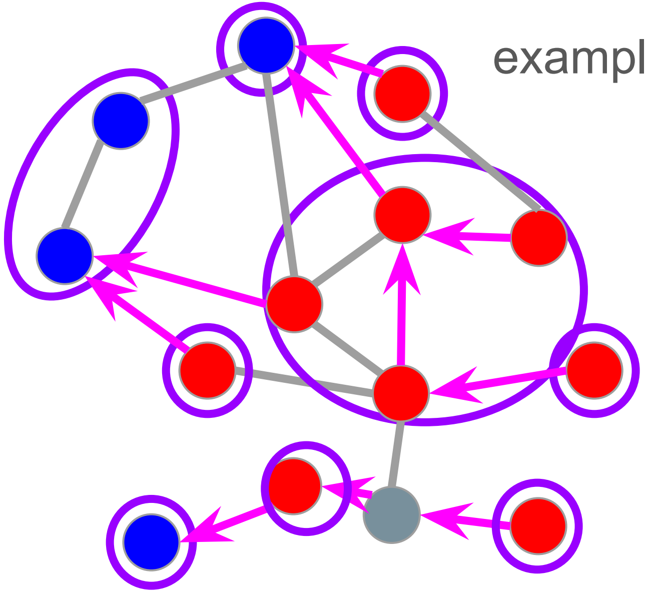
Our algorithm in **CONGEST**

The algorithm generates broadcast trees.

Each edge is in $O(\log n)$ of them.



Natural extension of the algorithm gives decomposition of G^k.



example for G^3

Outlook: Beyond deterministic & local

Δ+1 coloring in different models

LOCAL, deterministic	LOCAL, randomized	MPC, randomized
poly(log n)		

Outlook: Beyond deterministic & local

Δ+1 coloring in different models

LOCAL, deterministic		LOCAL, randomized	MPC, randomized
poly(log n)	→	poly(log log n)	

shattering [Chang, Li, Pettie STOC'18]
+network decomposition [R., Ghaffari 19+]

Outlook: Beyond deterministic & local

Δ+1 coloring in different models

amplification of success probability
[Chang, Kopelowitz, and Pettie FOCS'16]

LOCAL, deterministic	LOCAL, randomized	MPC, randomized
poly(log n)	poly(log log n)	

shattering [Chang, Li, Pettie STOC'18]
+network decomposition [R., Ghaffari 19+]

Outlook: Beyond deterministic & local

Δ+1 coloring in different models

amplification of success probability
[Chang, Kopelowitz, and Pettie FOCS'16]

LOCAL, deterministic

poly(log n)

LOCAL, randomized

poly(log log n)

MPC, randomized

O(log log log n)

shattering [Chang, Li, Pettie STOC'18]
+network decomposition [R., Ghaffari 19+]

graph exponentiation [Chang, Fischer, Ghaffari, Uitto, Zheng PODC'19]

Outlook: Beyond deterministic & local

Δ+1 coloring in different models

amplification of success probability
[Chang, Kopelowitz, and Pettie FOCS'16]

conditioned on hardness of connectivity
in MPC [Ghaffari, Kuhn, Uitto '19+]

LOCAL, deterministic

poly(log n)

LOCAL, randomized

poly(log log n)

MPC, randomized

O(log log log n)

shattering [Chang, Li, Pettie STOC'18]
+network decomposition [R., Ghaffari 19+]

graph exponentiation [Chang, Fischer,
Ghaffari, Uitto, Zheng PODC'19]

Open problems

Give $o(\log^6 n)$ **LOCAL** algorithm for net. decomposition, or even, say, $\Delta+1$ coloring.

Give poly(log n) **CONGEST** algorithm for strong diameter decomposition.

Find a *natural* deterministic poly(log n) algorithm for MIS/coloring in the **CONGEST** model.