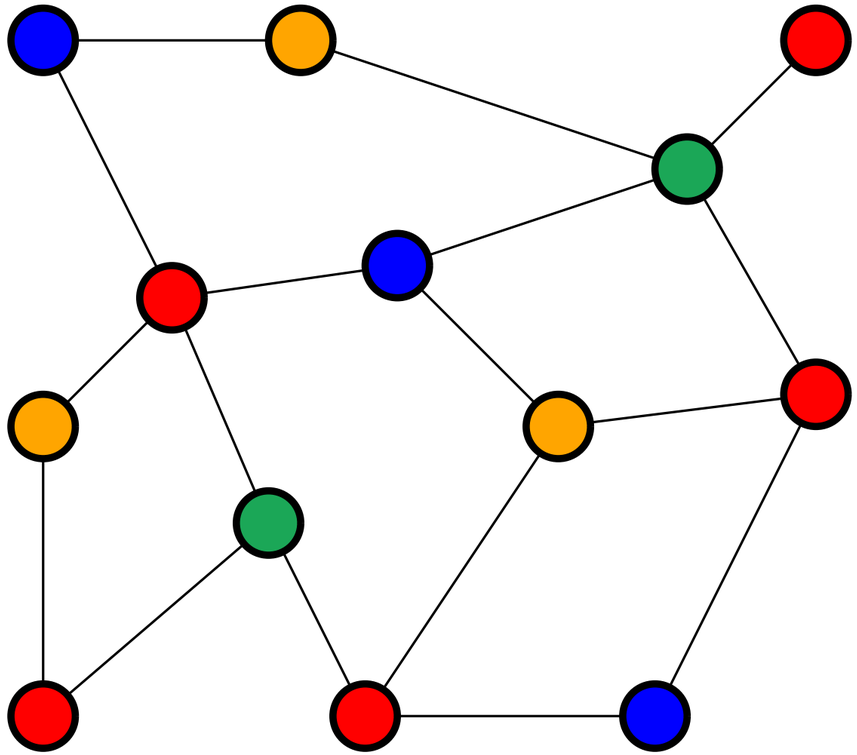
Recent Developments in Distributed Δ-coloring

Manuel Jakob

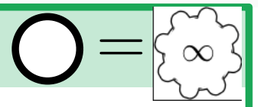
ATCS Seminar

Setup: distributed Δ-Coloring



Δ :=max. degree (here: $\Delta = 4$)

LOCAL-model



- unbounded communication in synchrounouse rounds
- each vertex needs to output its part of the solution
- complexity is the number of communication rounds

Δ-Coloring

- color vertices with colors $\{1, \ldots, \Delta\}$ s..t. no adjacent vertices get the same color

Assumptions::

- # vertices and Δ is known to all vertices
- max. degree Δ is constant

Previous Work on distributed Δ-Coloring

Previous Work on distributed Δ-Coloring

Lower Bound:

$\Omega(\log n)$ rounds [Chang, Kopelowitz, Pettie '16]

Previous Work on distributed Δ-Coloring

Lower Bound:

$\Omega(\log n)$ rounds — [Chang, Kopelowitz, Pettie '16]

Upper Bound:

$O(\log^3 n)$ rounds — [Panconesi, Sirinivasan '93]

$O(\log^2 n)$ rounds — [Ghaffari, Hirvonen, Kuhn, Maus '18]

Previous Work on distributed Δ-Coloring

Lower Bound:

$\Omega(\log n)$ rounds	[Chang, Kopelowitz, Pettie '16]

Upper Bound:

$O(\log^3 n)$ rounds	[Panconesi, Sirinivasan '93]
$O(\log^2 n)$ rounds	[Ghaffari, Hirvonen, Kuhn, Maus '18]

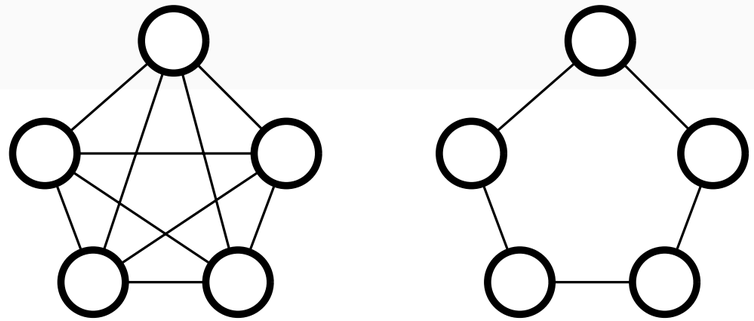
$O(\log n \cdot \log^* n)$ rounds	[Bourreau, Brandt, Nolin '25]
$O(\log n)$ rounds on dense graphs	[**J**, Maus '25]
$O(\log n)$ rounds	[Bourreau, Brandt, Nolin '26]

Preliminaries/Tools

Preliminaries/Tools

Brooks Theorem: [Brooks '41]

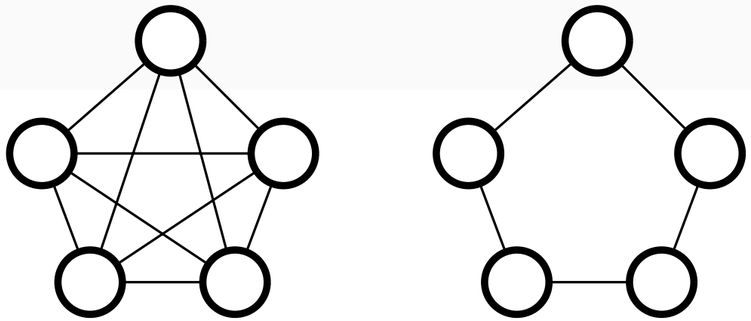
A graph G can be Δ-colored if it doesn't contain a clique of size $\Delta + 1$ or an odd cycle with $\Delta = 2$



Preliminaries/Tools

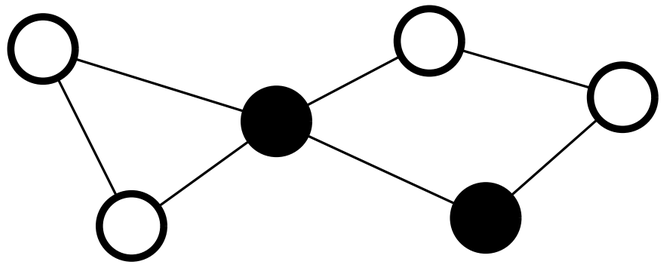
Brooks Theorem: [Brooks '41]

A graph G can be Δ-colored if it doesn't contain a clique of size $\Delta + 1$ or an odd cycle with $\Delta = 2$



Maximal Independent Set (MIS):

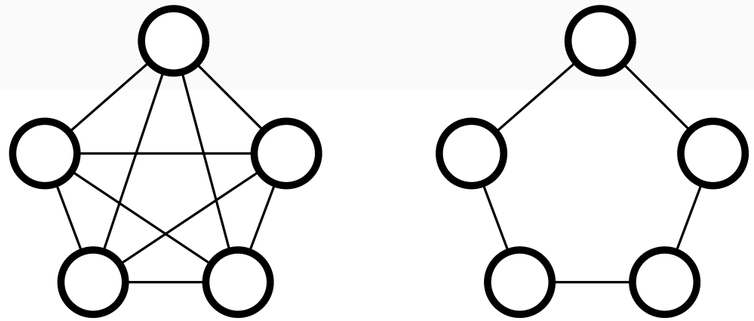
A set of non-adjacent vertices that cannot be extended further. $O(\log^* n)$ [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]



Preliminaries/Tools

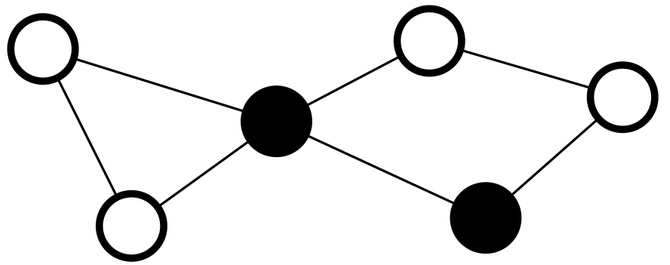
Brooks Theorem: [Brooks '41]

A graph G can be Δ-colored if it doesn't contain a clique of size $\Delta + 1$ or an odd cycle with $\Delta = 2$



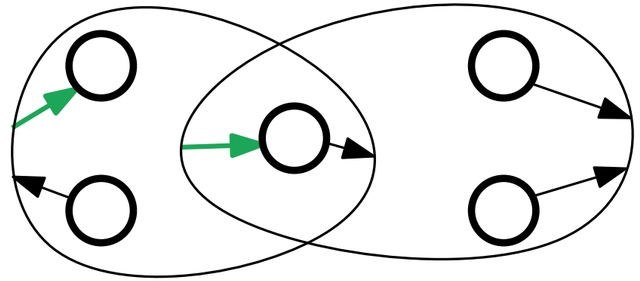
Maximal Independent Set (MIS):

A set of non-adjacent vertices that cannot be extended further. $O(\log^* n)$ [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]



(Hyper-)Edge Sinkless Orientation (H)SO:

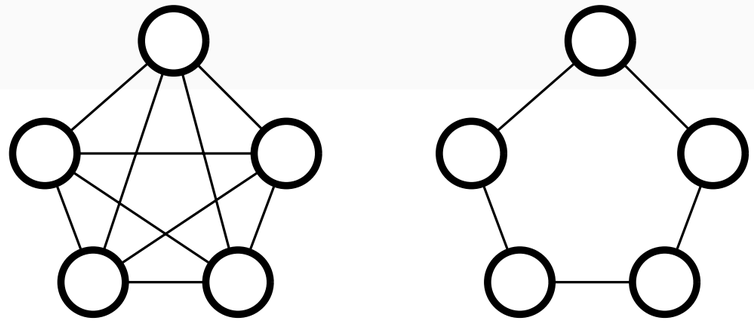
An orientation of (hyper-)edges where no vertex is a sink. $O(\log_{\delta/r} n)$ [Brandt, Maus, Narayanan, Schager '25]



Preliminaries/Tools

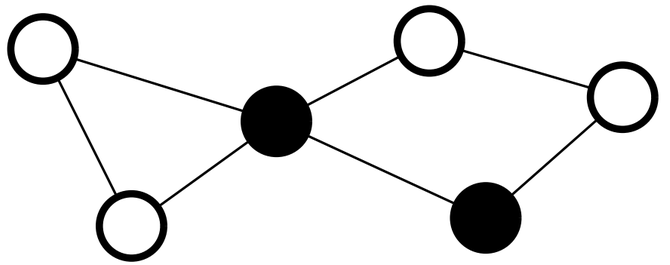
Brooks Theorem: [Brooks '41]

A graph G can be Δ-colored if it doesn't contain a clique of size $\Delta + 1$ or an odd cycle with $\Delta = 2$



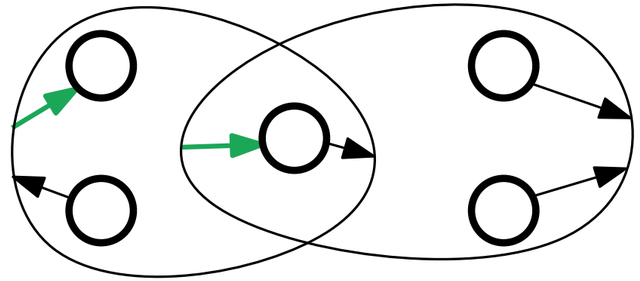
Maximal Independent Set (MIS):

A set of non-adjacent vertices that cannot be extended further. $O(\log^* n)$ [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]



(Hyper-)Edge Sinkless Orientation (H)SO:

An orientation of (hyper-)edges where no vertex is a sink. $O(\log_{\delta/r} n)$ [Brandt, Maus, Narayanan, Schager '25]



$(\Delta + 1)$ - Coloring / $deg + 1$ - Coloring

$O(\log^* n)$ [Linial '92]

Why is $(\Delta + 1)$-coloring so much faster?

Why is $(\Delta + 1)$-coloring so much faster?

Definition: $slack(v)$

$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

Why is $(\Delta + 1)$-coloring so much faster?

Definition: $slack(v)$

$slack(v) := \#$ available colors to $v - \#$ uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

Why is $(\Delta + 1)$-coloring so much faster?

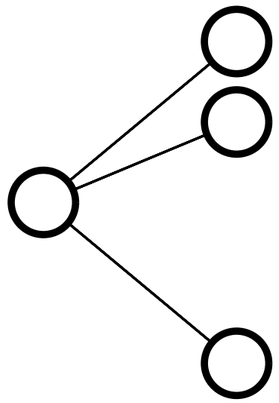
Definition: $slack(v)$

$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

natural slack



Why is $(\Delta + 1)$-coloring so much faster?

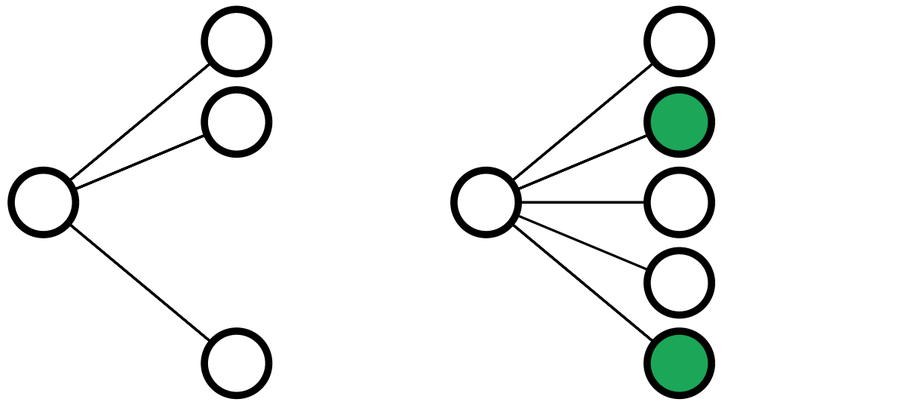
Definition: $slack(v)$

$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

natural slack



permanent slack

Why is $(\Delta + 1)$-coloring so much faster?

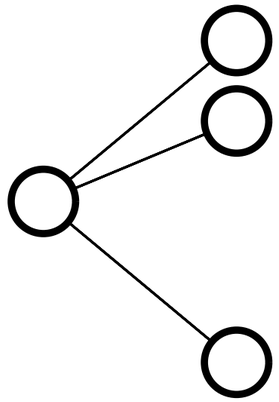
Definition: $slack(v)$

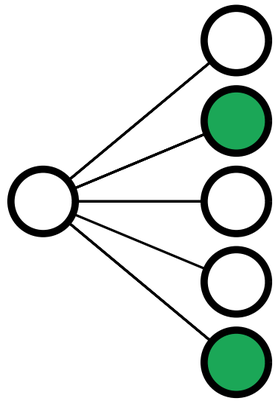
$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

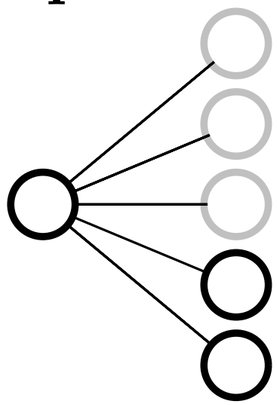
Δ-coloring: $\forall v \in V : slack(v) \geq 0$

natural slack





temporal slack



permanent slack

Why is $(\Delta + 1)$-coloring so much faster?

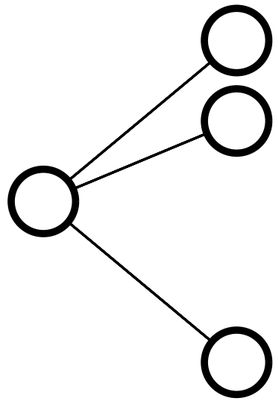
Definition: $slack(v)$

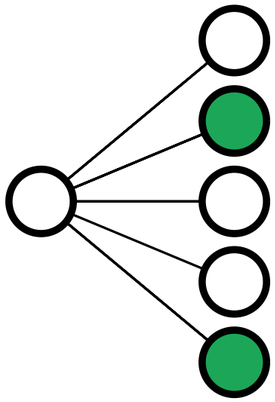
$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

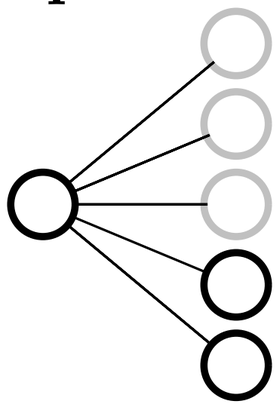
natural slack

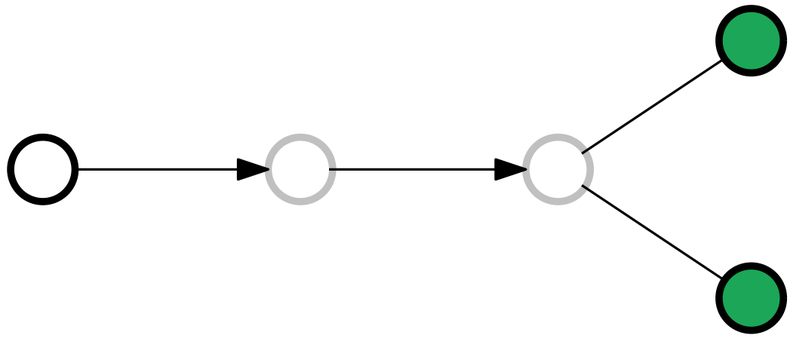




permanent slack

temporal slack





propagation of slack

Why is $(\Delta + 1)$-coloring so much faster?

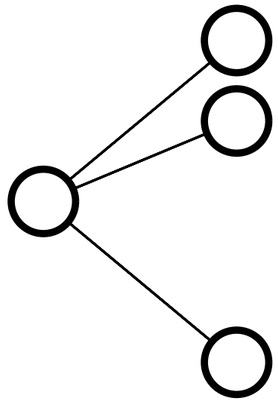
Definition: $slack(v)$

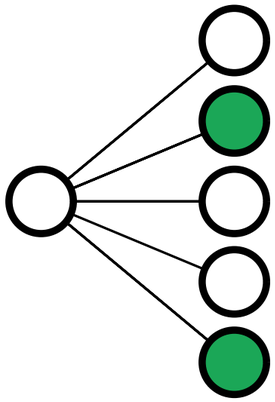
$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

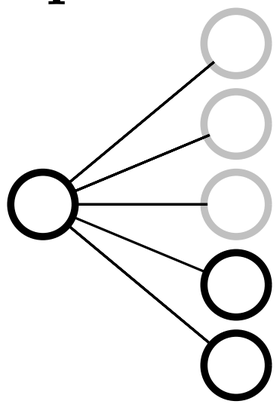
natural slack

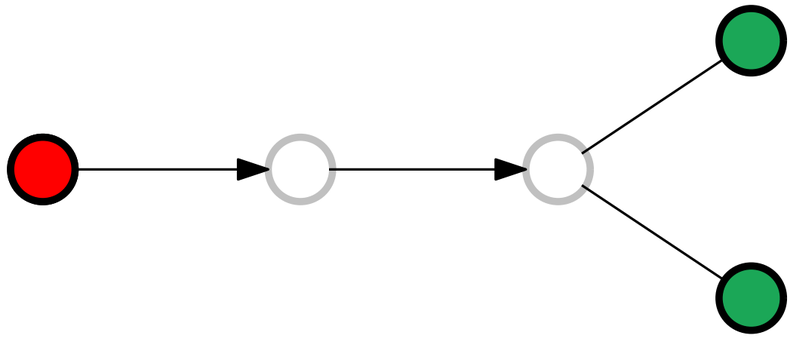




permanent slack

temporal slack





propagation of slack

Why is $(\Delta + 1)$-coloring so much faster?

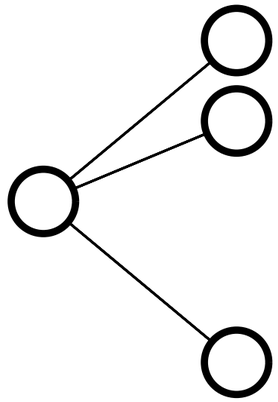
Definition: $slack(v)$

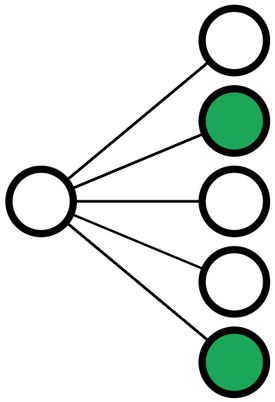
$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

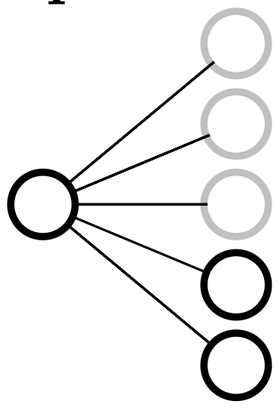
natural slack

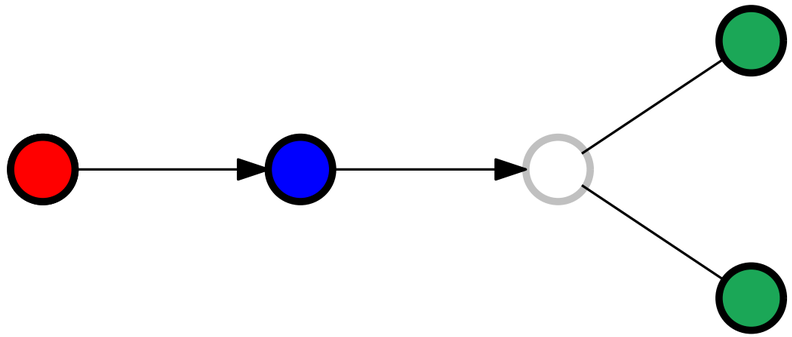




permanent slack

temporal slack





propagation of slack

Why is $(\Delta + 1)$-coloring so much faster?

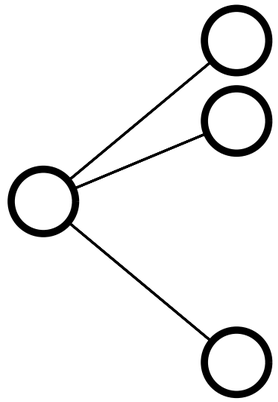
Definition: $slack(v)$

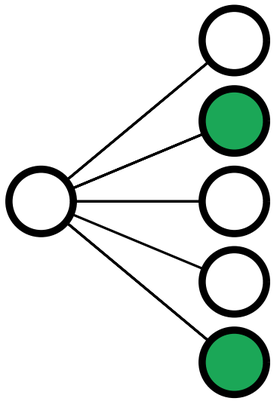
$slack(v) :=$ # available colors to v $-$ # uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

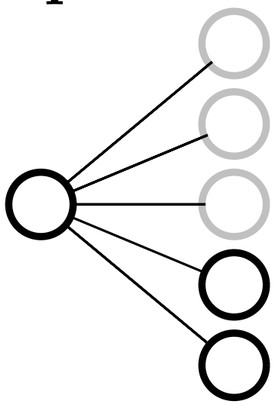
natural slack

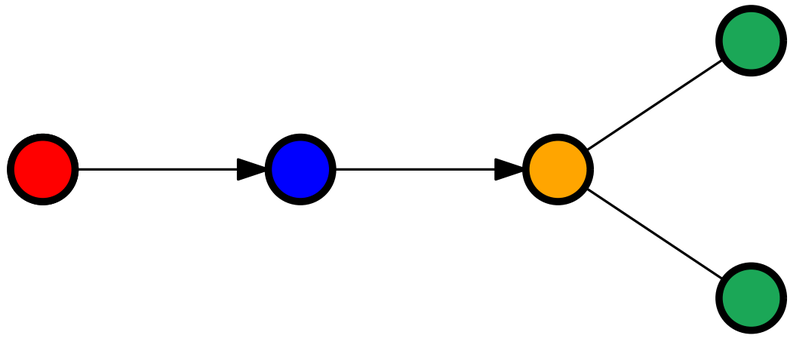




permanent slack

temporal slack





propagation of slack

Why is $(\Delta + 1)$-coloring so much faster?

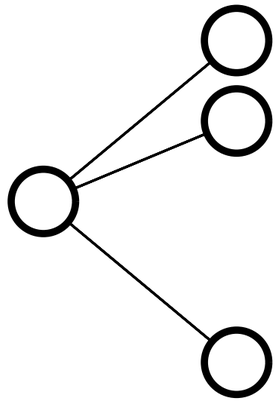
Definition: $slack(v)$

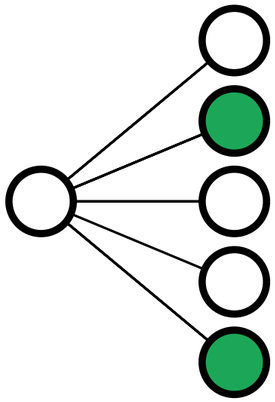
$slack(v) := \#$ available colors to $v - \#$ uncolored vertices in $N(v)$

$(\Delta + 1)$-coloring: $\forall v \in V : slack(v) \geq 1$ $\Rightarrow$ greedy solvable

Δ-coloring: $\forall v \in V : slack(v) \geq 0$

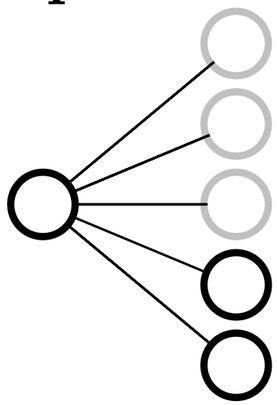
natural slack



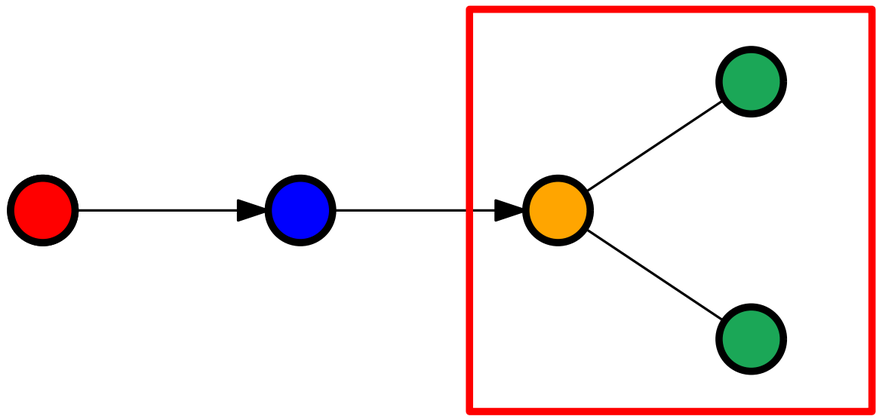


permanent slack

temporal slack



Slack Triad



propagation of slack

Δ-Coloring via Ruling Subgraphs

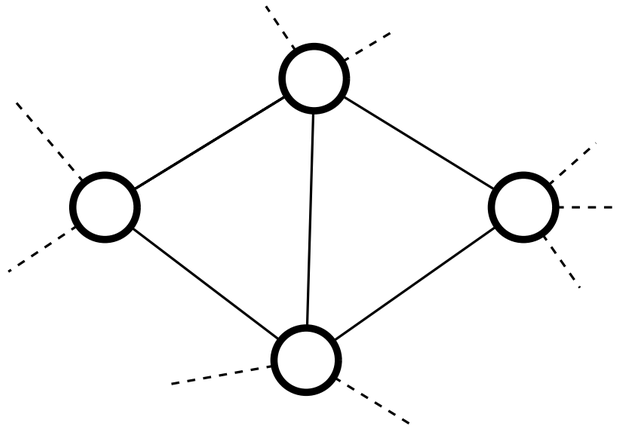
[Bourreau, Brandt, Nolin '25]

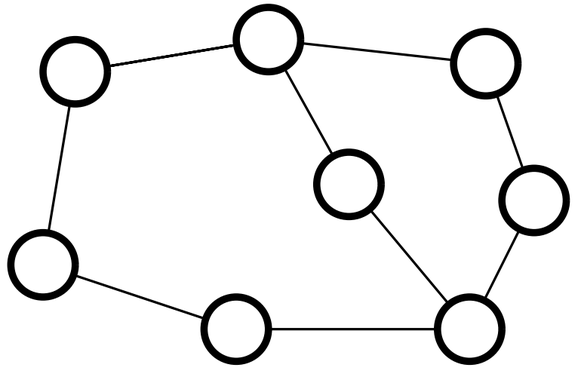
Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

Definition: Degree Choosable Component (DCC)

is a graph for which it is always possible to complete a proper Δ-coloring



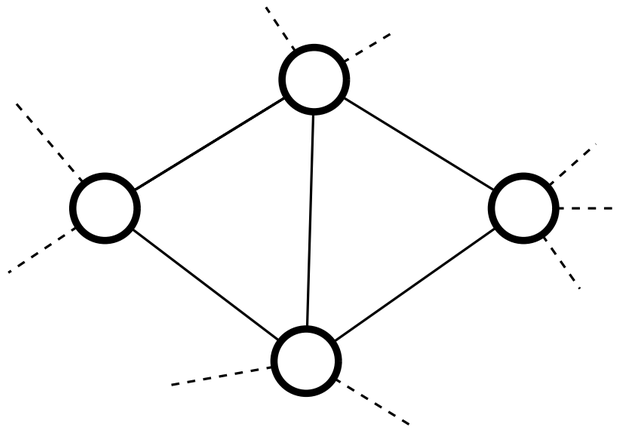


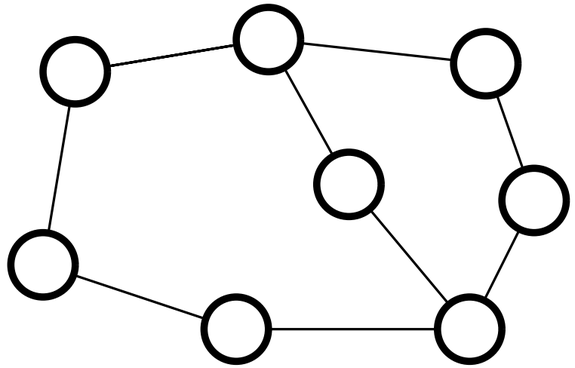
Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

Definition: Degree Choosable Component (DCC)

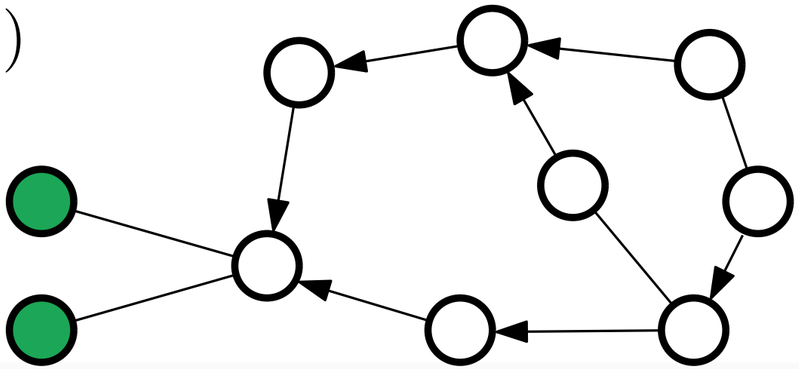
is a graph for which it is always possible to complete a proper Δ-coloring



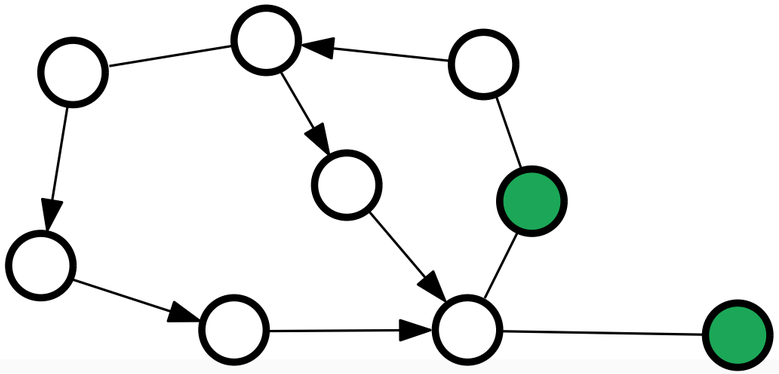


Proof:

(1)



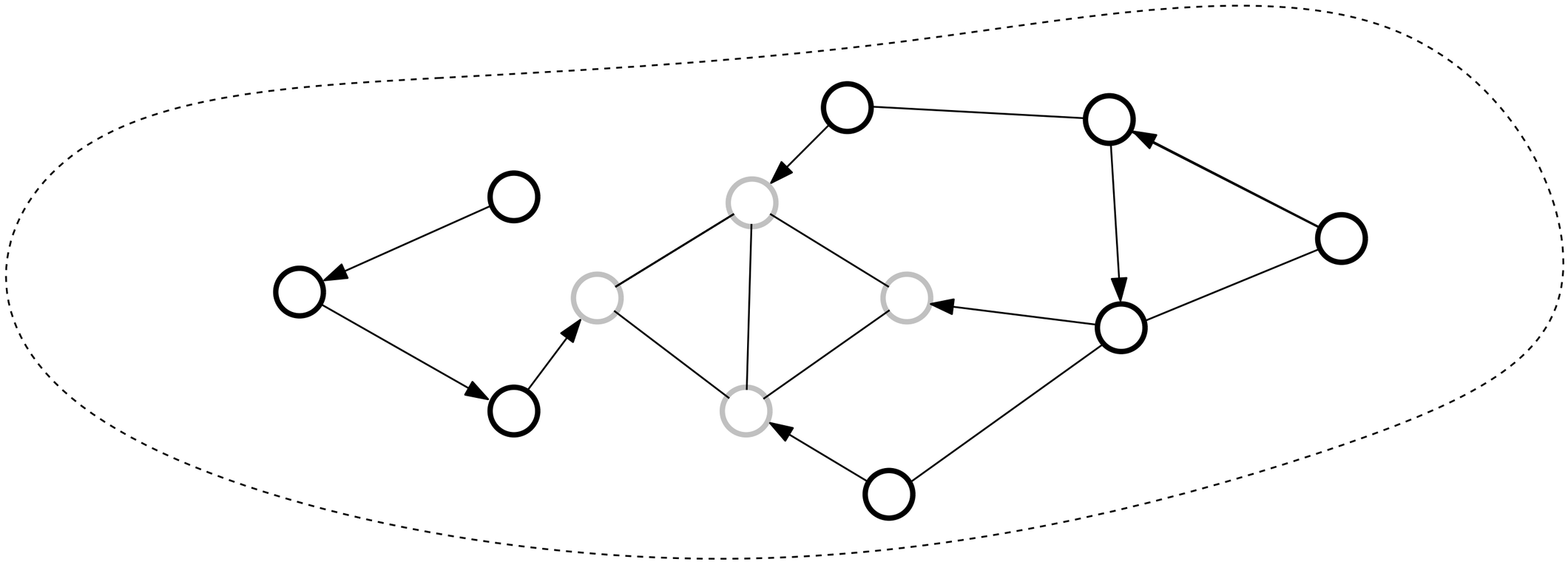
(2)



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

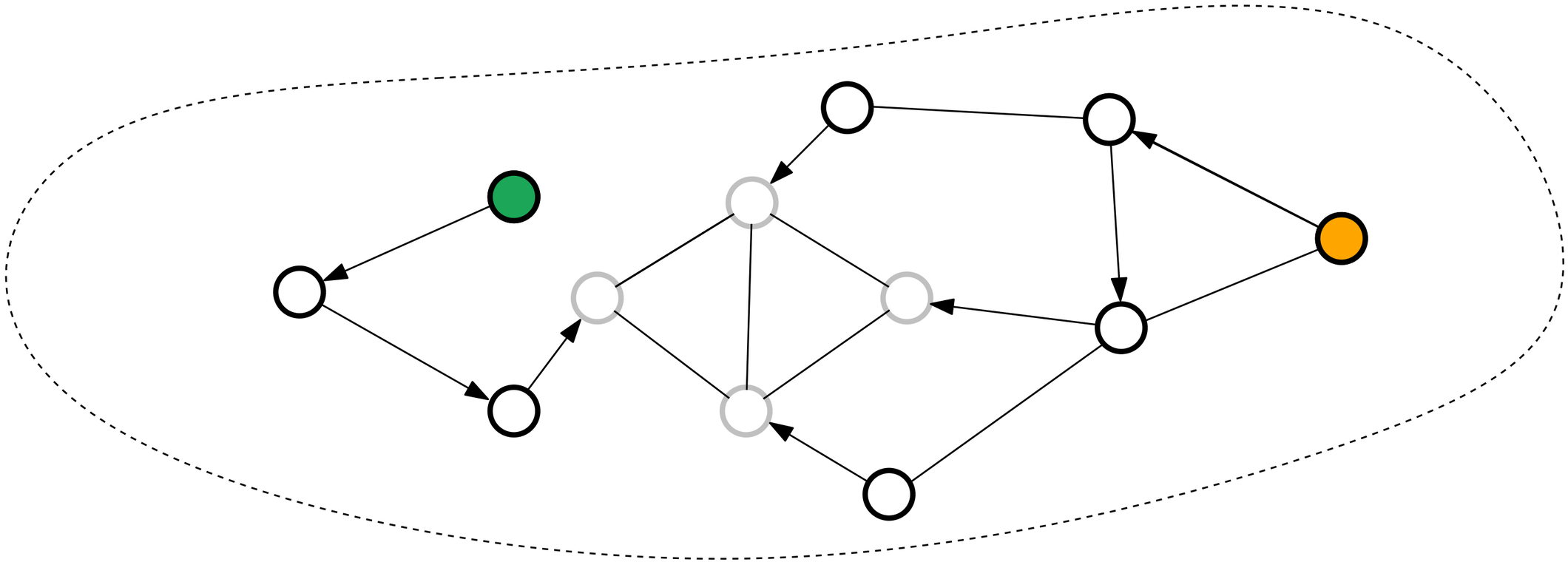
Approach:



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

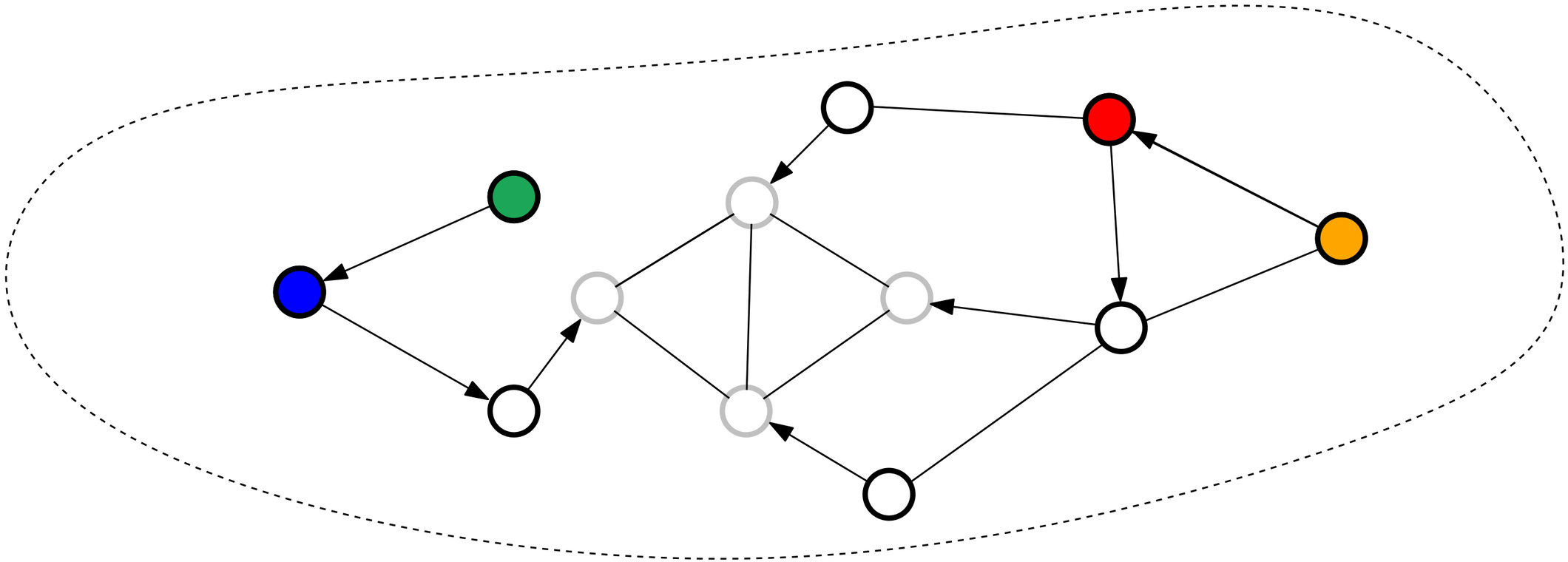
Approach:



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

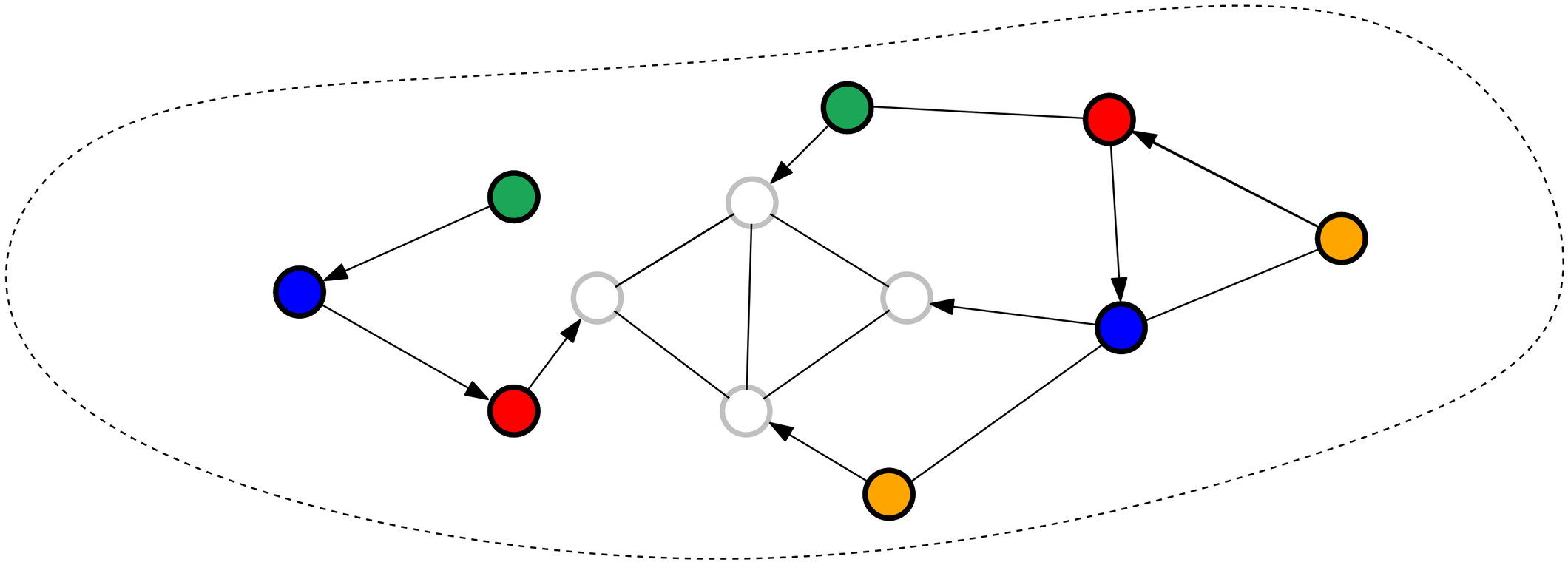
Approach:



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

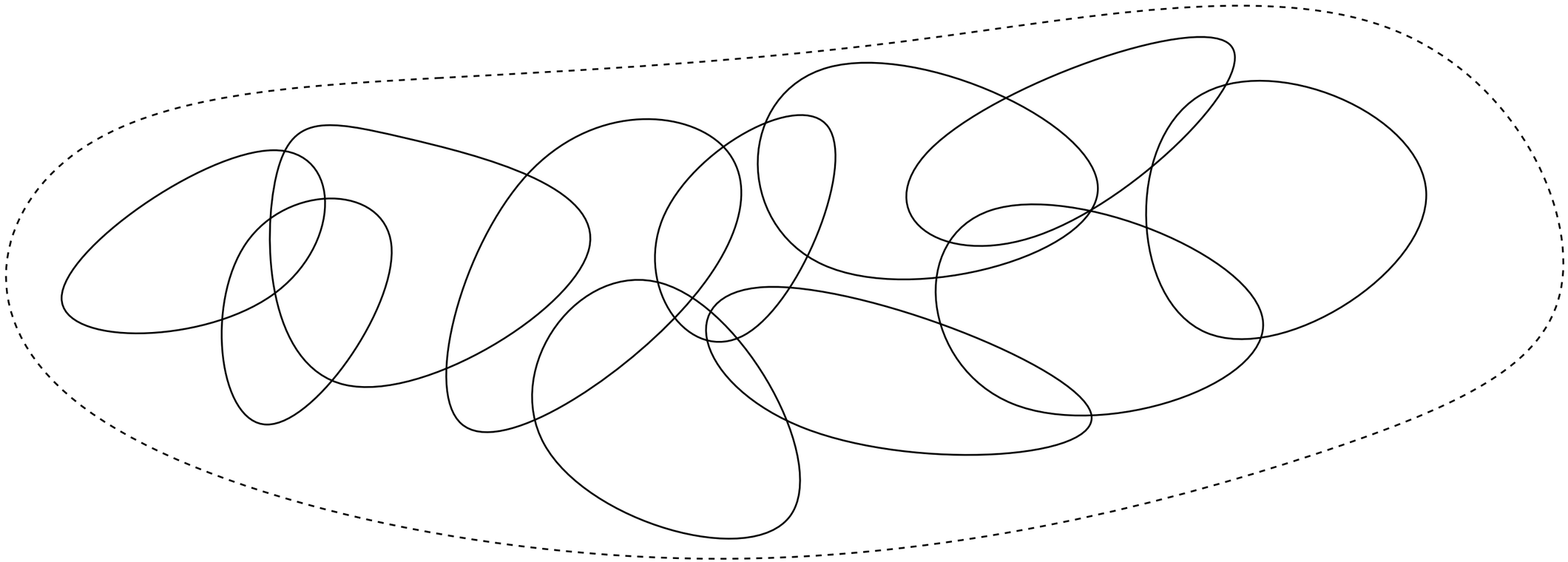
Approach:



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

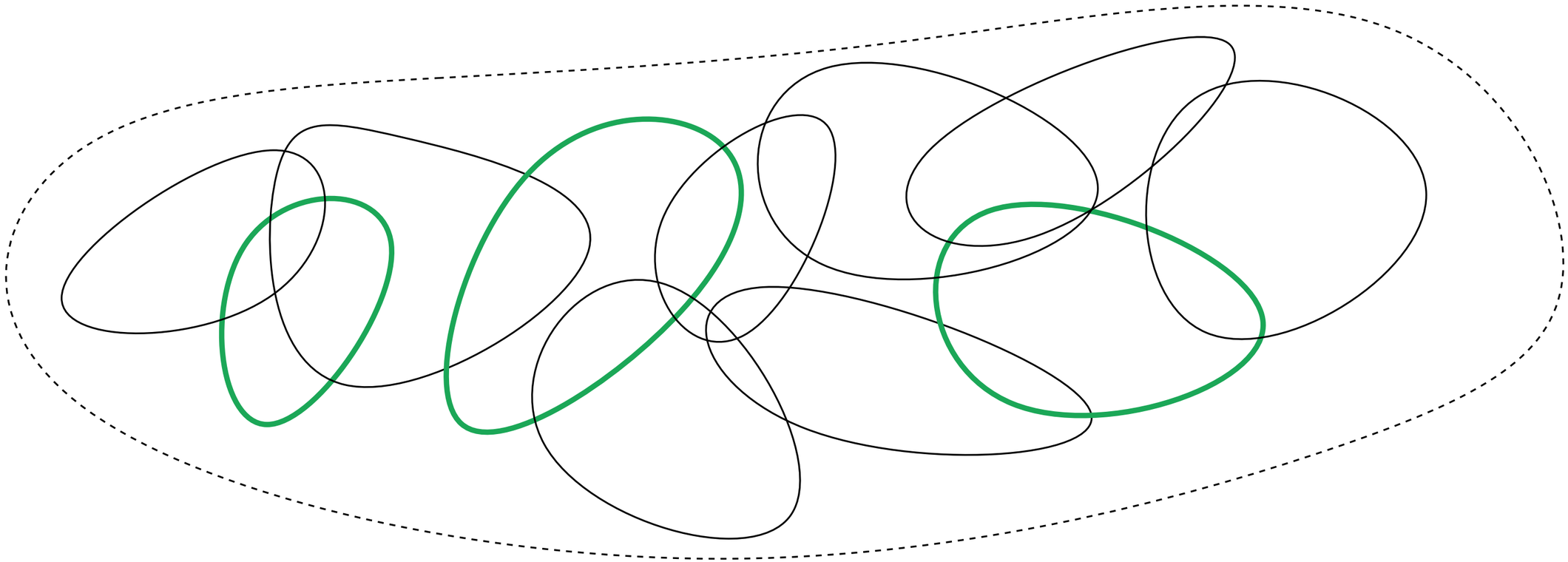
Problem: DCCs can overlap...



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

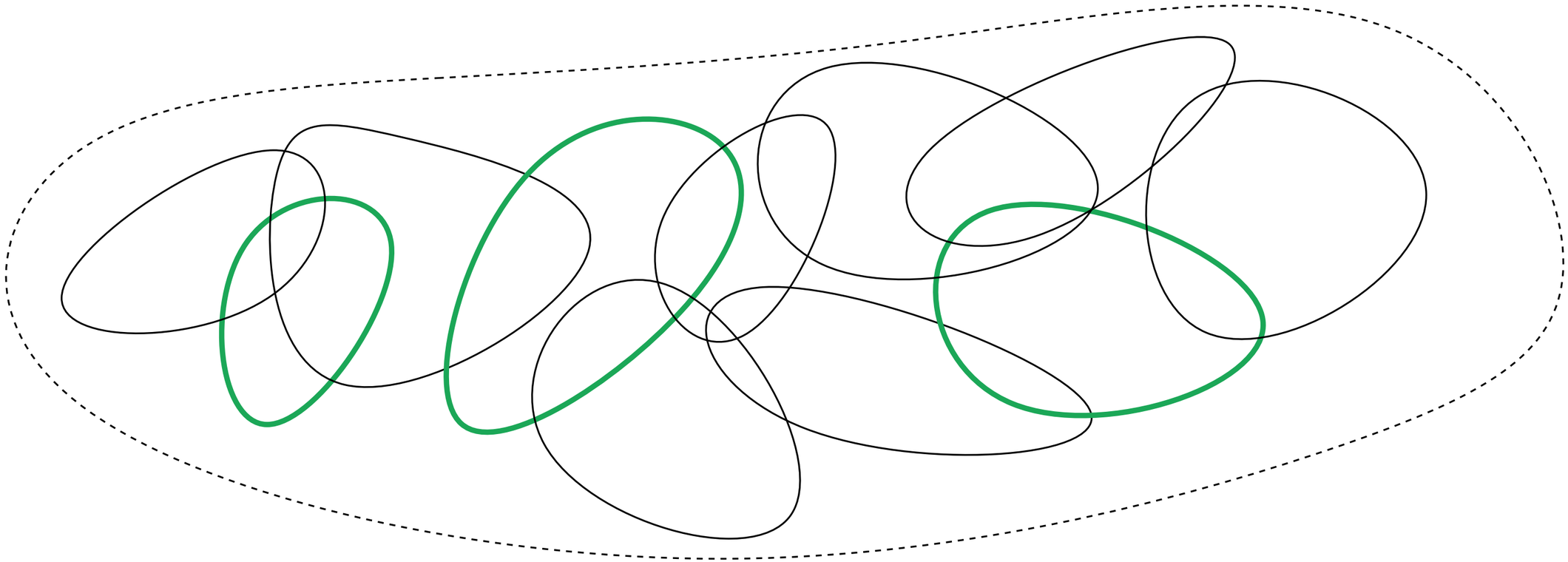
Problem: DCCs can overlap...



Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

Problem: DCCs can overlap...



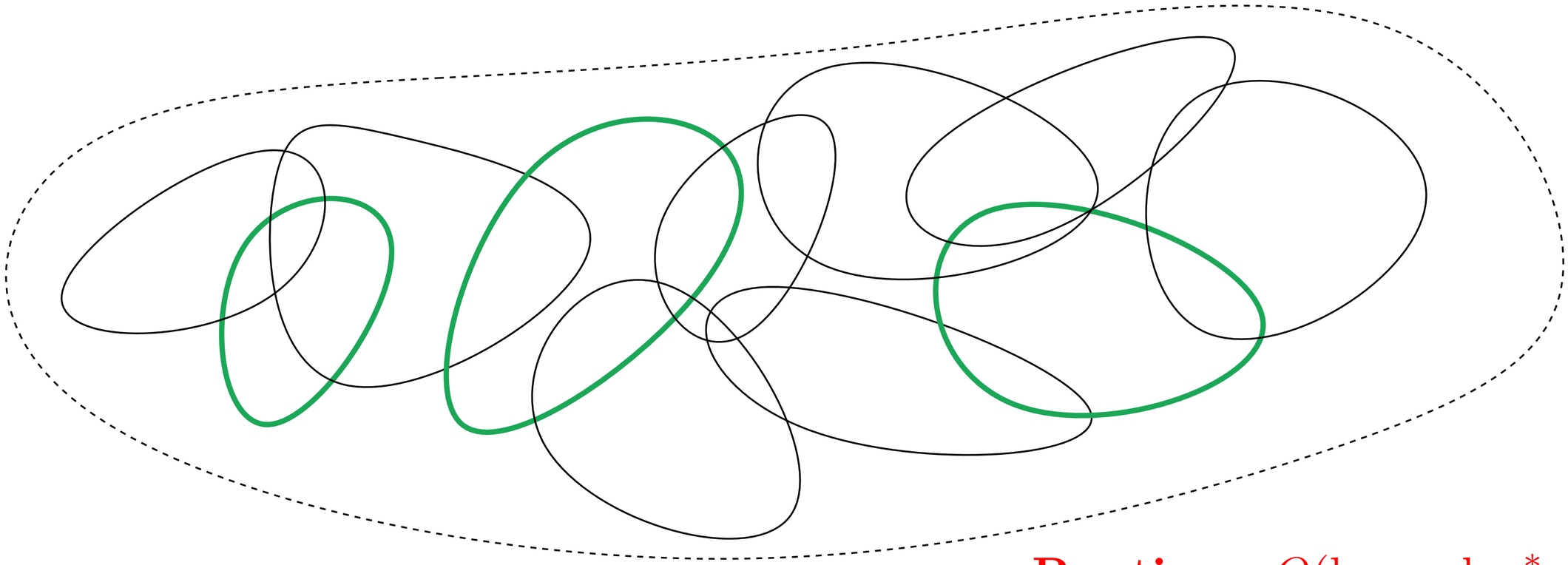
Algorithm (simplified):

1. each vertex finds all DCCs in a $\log n$ neighborhood and selects one arbitrary
2. compute an independent set of DCCs
3. color vertices in decreasing distance to the closest DCC ($(\Delta+1)$-coloring)

Δ-Coloring via Ruling Subgraphs

[Bourreau, Brandt, Nolin '25]

Problem: DCCs can overlap...



Runtime: $O(\log n \cdot \log^* n)$

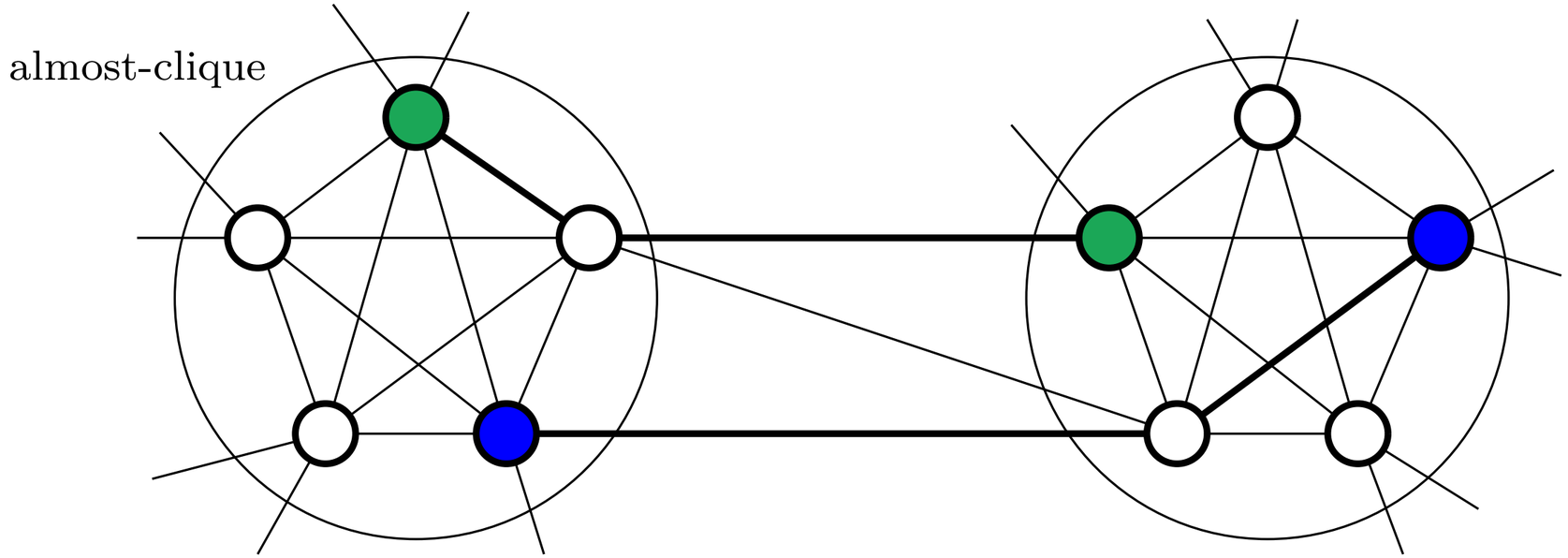
Algorithm (simplified):

1. each vertex finds all DCCs in a $\log n$ neighborhood and selects one arbitrary
2. compute an independent set of DCCs
3. color vertices in decreasing distance to the closest DCC ($(\Delta+1)$-coloring)

Towards an optimal algorithm

[**J**, Maus '25]

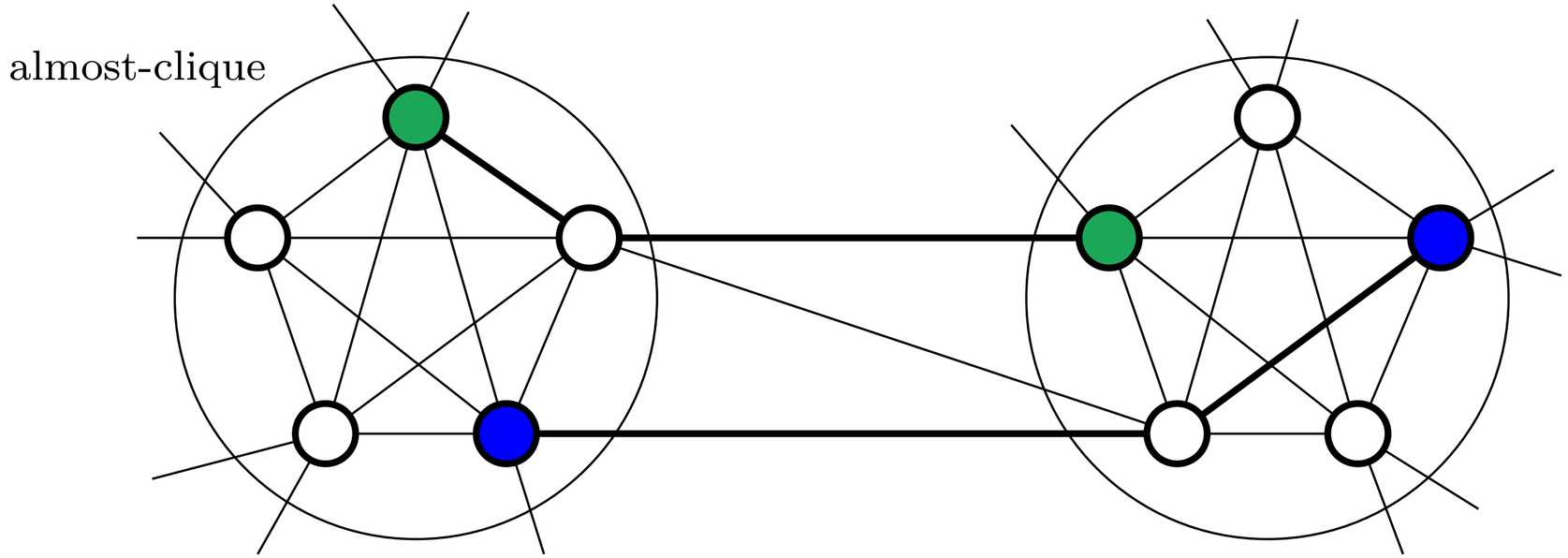
locally-dense graph:



Towards an optimal algorithm

[**J**, Maus '25]

locally-dense graph:



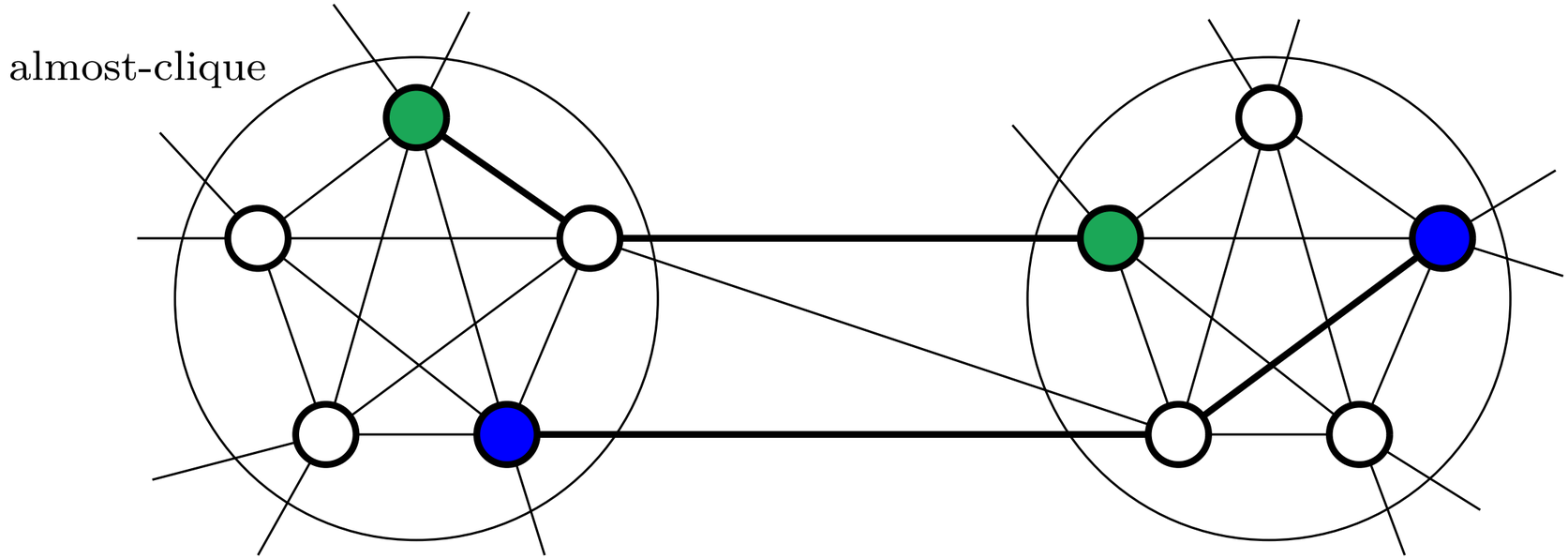
Algorithm (simplified):

1. compute a Maximal Matching of the edges between clusters
2. choose one edge for each cluster (HSO)
3. color the slack triads
4. color remaining vertices of the cliques ($(\Delta+1)$-coloring)

Towards an optimal algorithm

[**J**, Maus '25]

locally-dense graph:



Algorithm (simplified):

1. compute a Maximal Matching of the edges between clusters
2. choose one edge for each cluster (HSO)
3. color the slack triads
4. color remaining vertices of the cliques ($(\Delta+1)$-coloring)

Runtime: $O(\log n)$

Towards an optimal algorithm

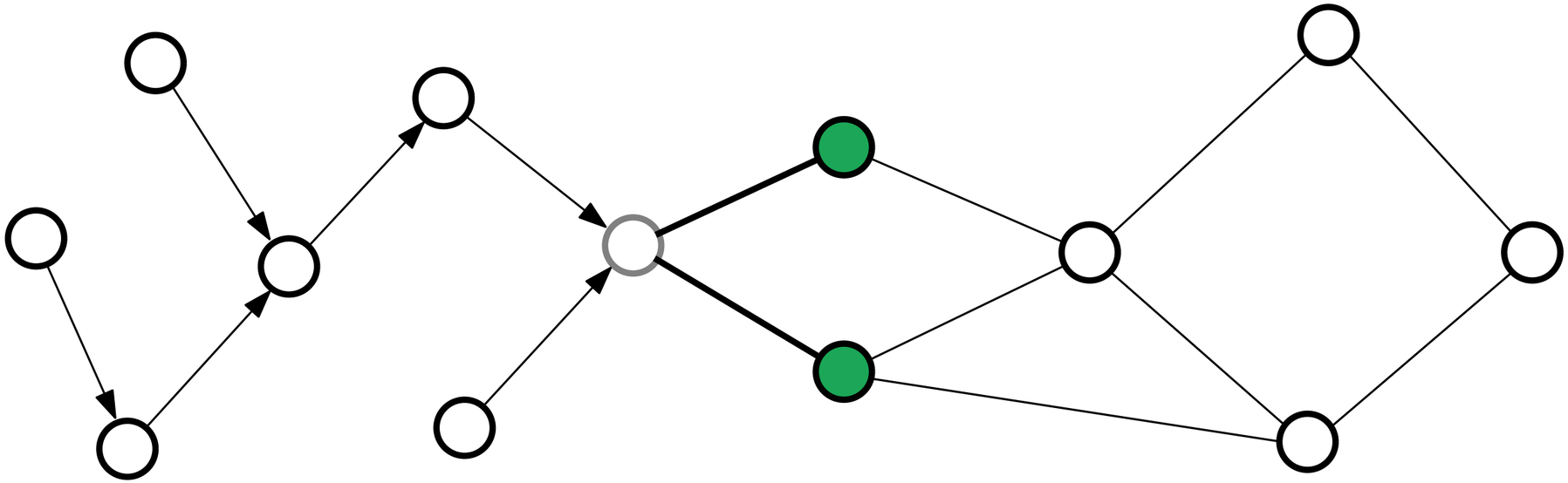
[**J**, Maus '25]

Why is this not applicable to all graphs?

Towards an optimal algorithm

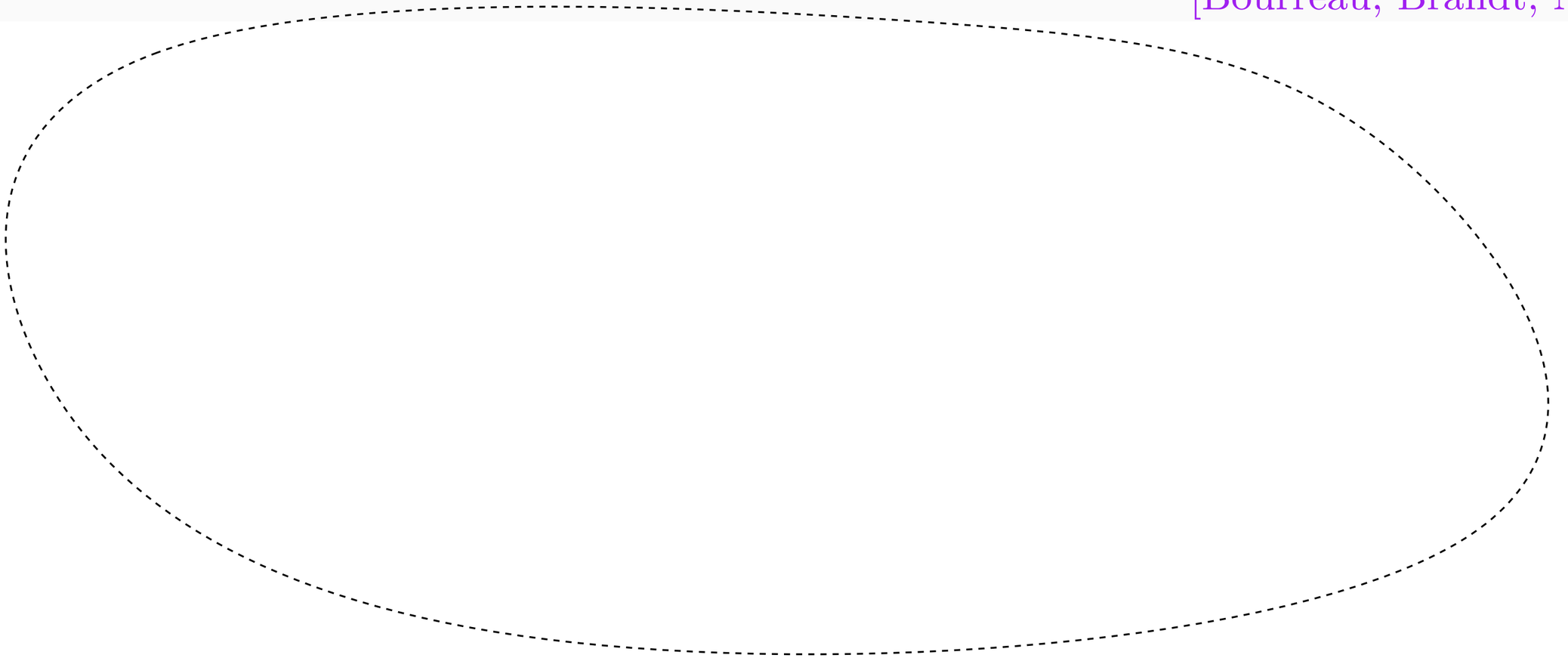
[**J**, Maus '25]

Why is this not applicable to all graphs?



Reduction to MIS

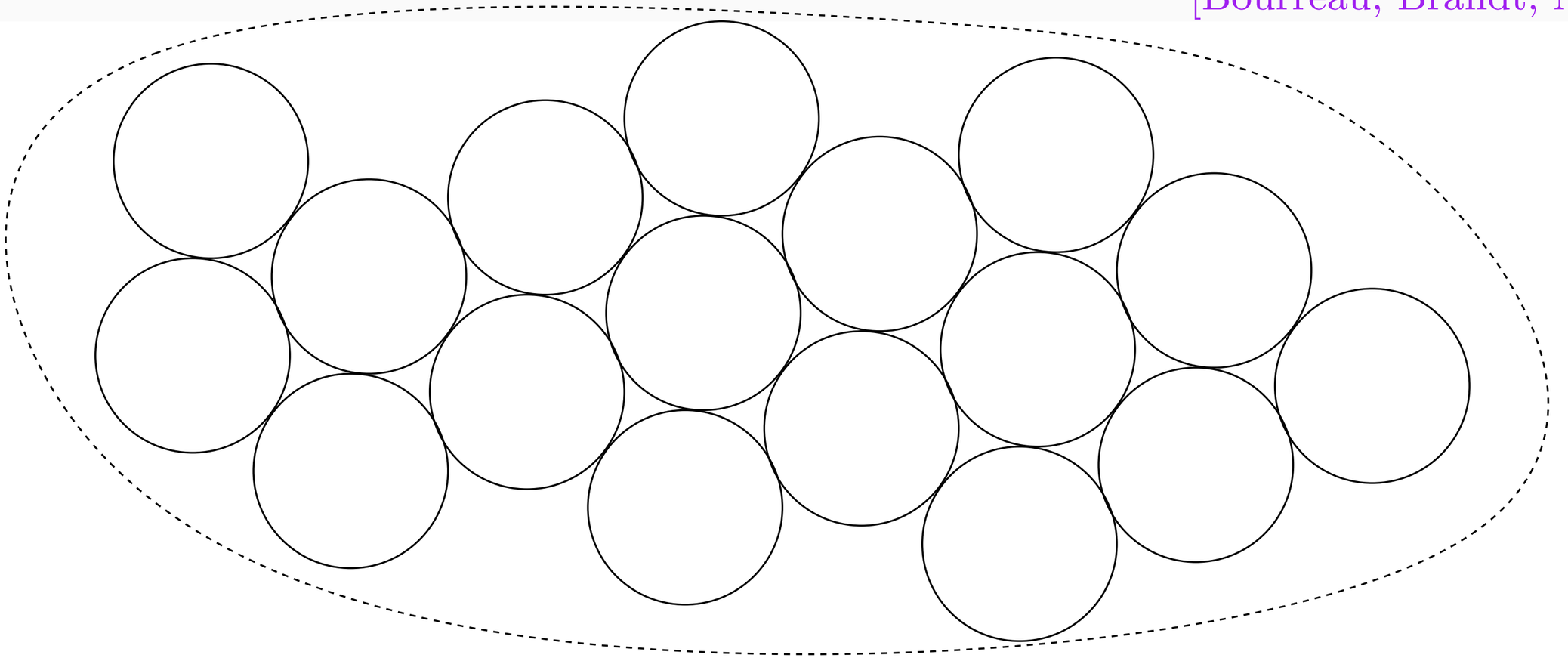
[Bourreau, Brandt, Nolin '26]



Algorithm (simplified):

Reduction to MIS

[Bourreau, Brandt, Nolin '26]

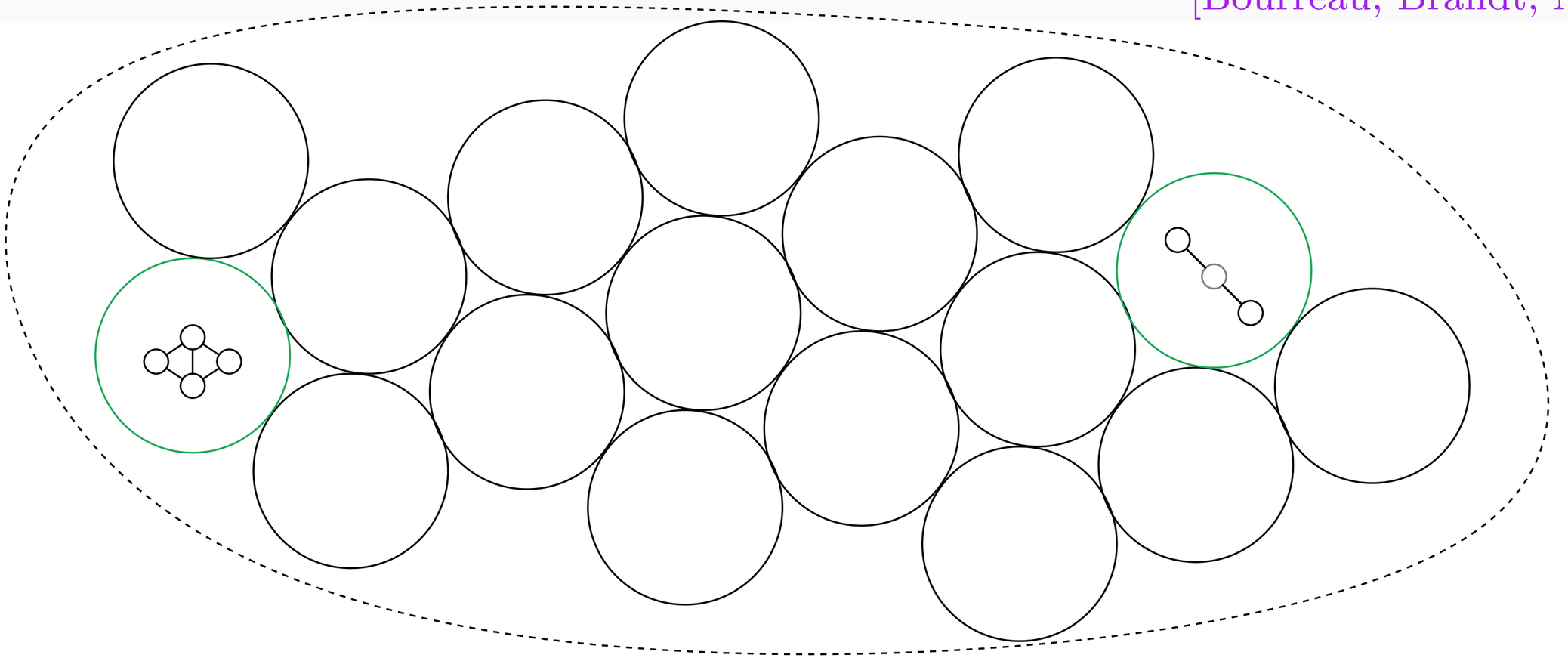


Algorithm (simplified):

1. compute a MIS on G^c

Reduction to MIS

[Bourreau, Brandt, Nolin '26]

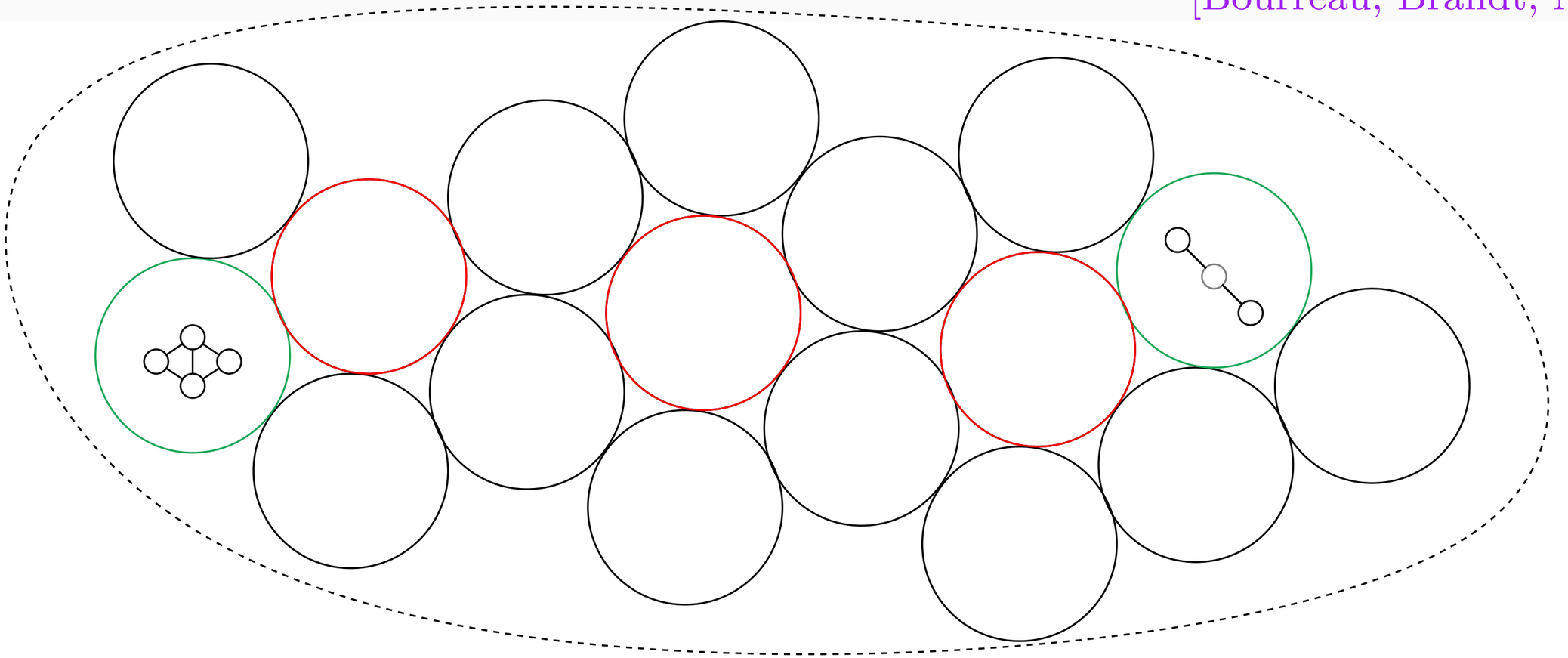


Algorithm (simplified):

1. compute a MIS on G^c
2. define C_{DCC} clusters

Reduction to MIS

[Bourreau, Brandt, Nolin '26]

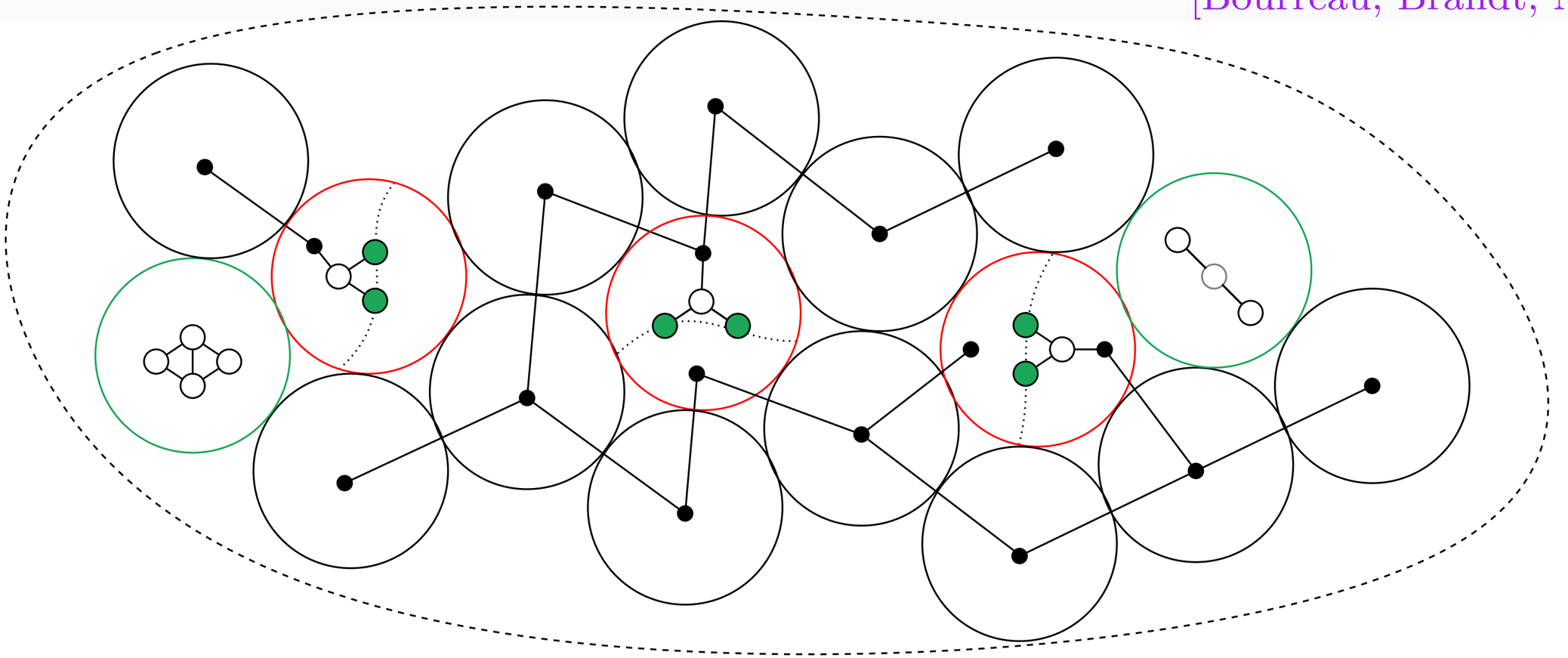


Algorithm (simplified):

1. compute a MIS on G^c
2. define C_{DCC} clusters
3. compute MIS to get C_{flex} and C_{link}

Reduction to MIS

[Bourreau, Brandt, Nolin '26]

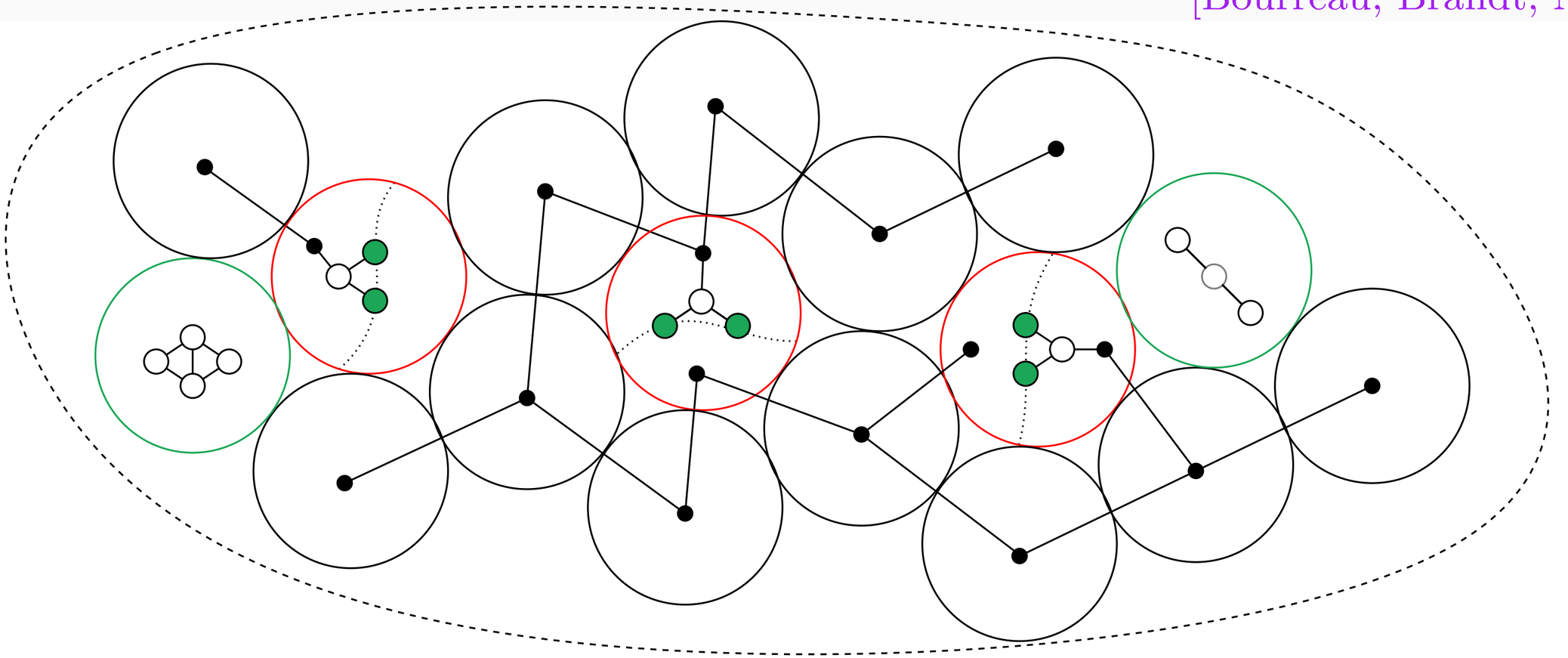


Algorithm (simplified):

1. compute a MIS on G^c
2. define C_{DCC} clusters
3. compute MIS to get C_{flex} and C_{link}
4. define Slack Triads in C_{flex} (HSO)
5. $(\Delta+1)$-coloring

Reduction to MIS

[Bourreau, Brandt, Nolin '26]



Algorithm (simplified):

1. compute a MIS on G^c
2. define C_{DCC} clusters
3. compute MIS to get C_{flex} and C_{link}
4. define Slack Triads in C_{flex} (HSO)
5. $(\Delta+1)$-coloring

Runtime: $O(\log n)$

Summary

Summary

Conclusion

Δ-coloring is tight $\Theta(\log n)$ rounds for const. Δ

Summary

Conclusion

Δ-coloring is tight $\Theta(\log n)$ rounds for const. Δ

What was missing?

Δ-coloring for $\Delta = f(n)$

randomized Δ-coloring

Summary

Conclusion

Δ-coloring is tight $\Theta(\log n)$ rounds for const. Δ

What was missing?

Δ-coloring for $\Delta = f(n)$

randomized Δ-coloring

Thanks!