Distributed Algorithms

for Wireless Multihop Networks

Roger Wattenhofer

Overview

Distributed Algorithms …

MIS

Local Model

Time Complexity

Randomized Algorithm

Applications

Ring Lower Bound

Ring Upper Bound

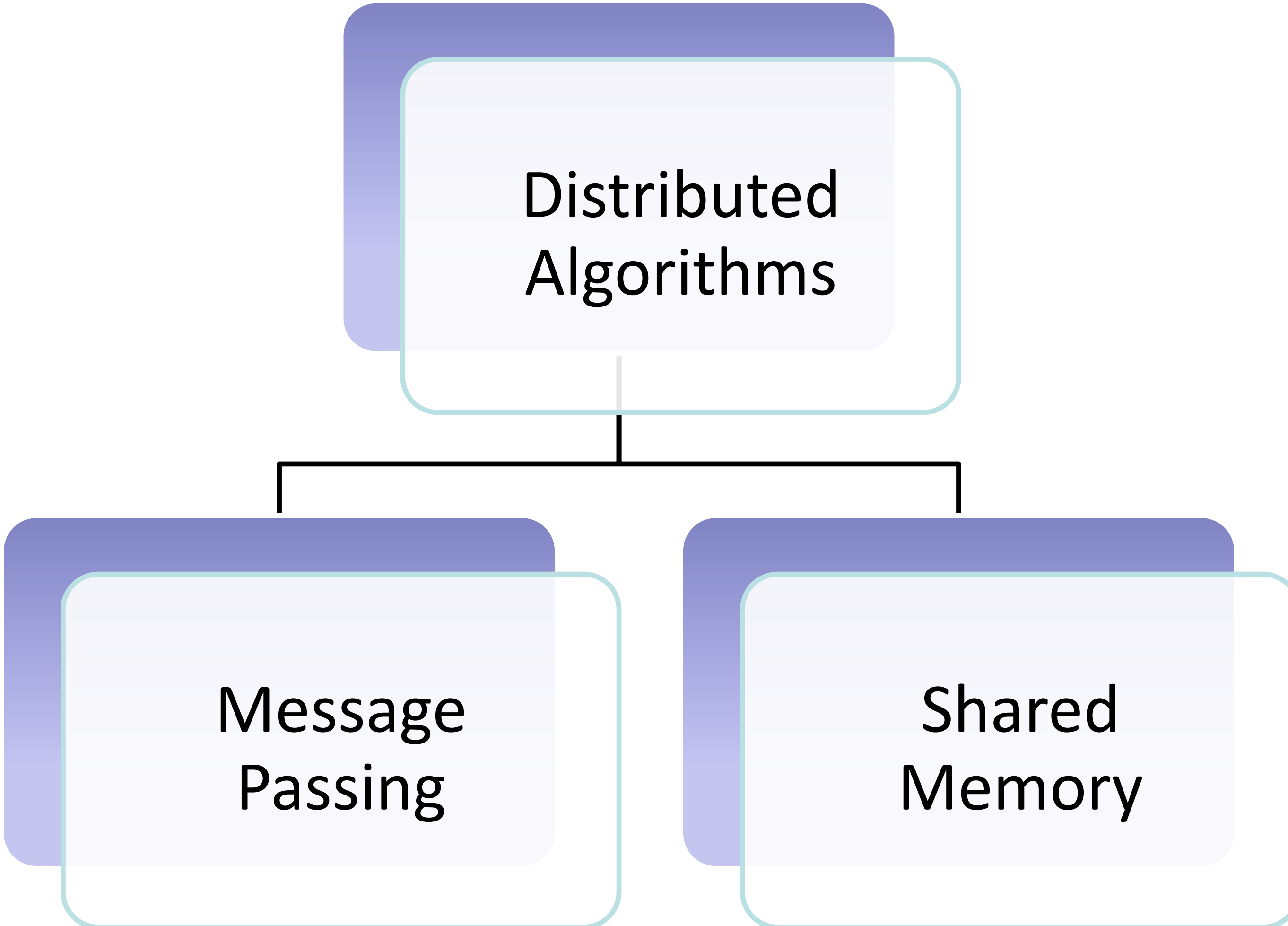
General Lower Bound

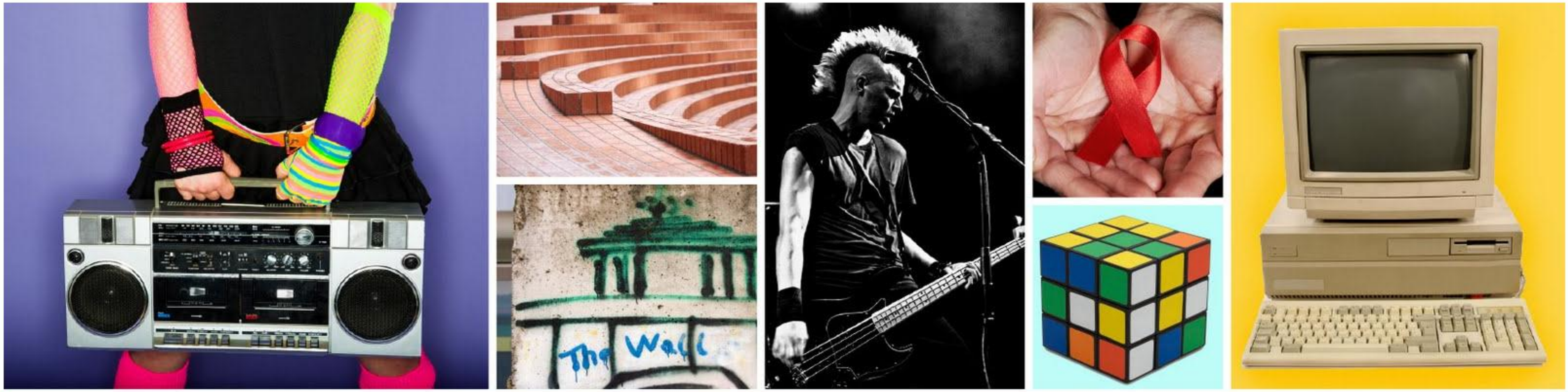
… for Wireless Multihop Networks

Connectivity Models

Interference Models

Communication Models

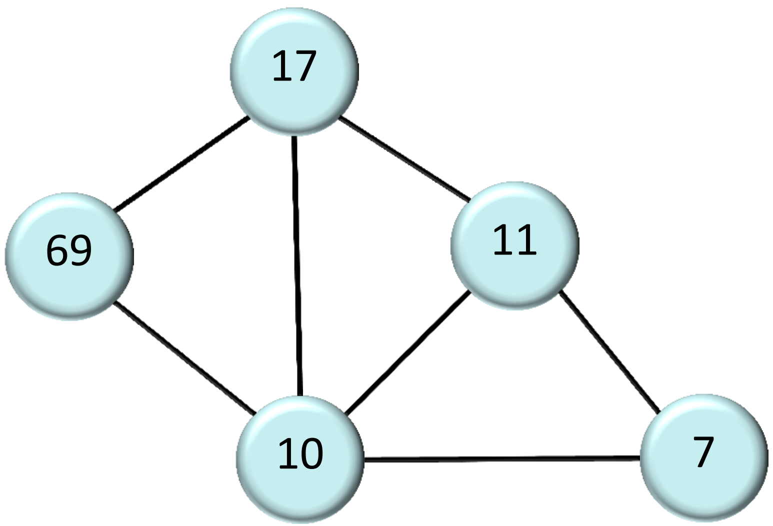

Distributed Algorithms
Message Passing
Shared Memory



THE 1980s

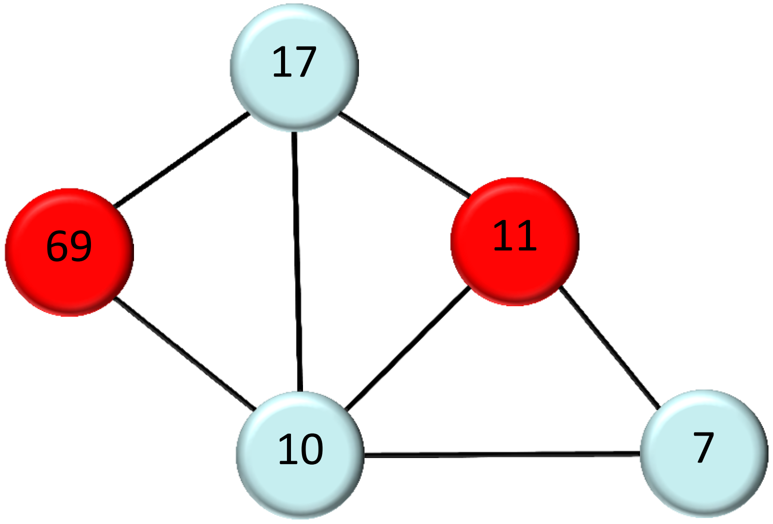
Example: Maximal Independent Set (MIS)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



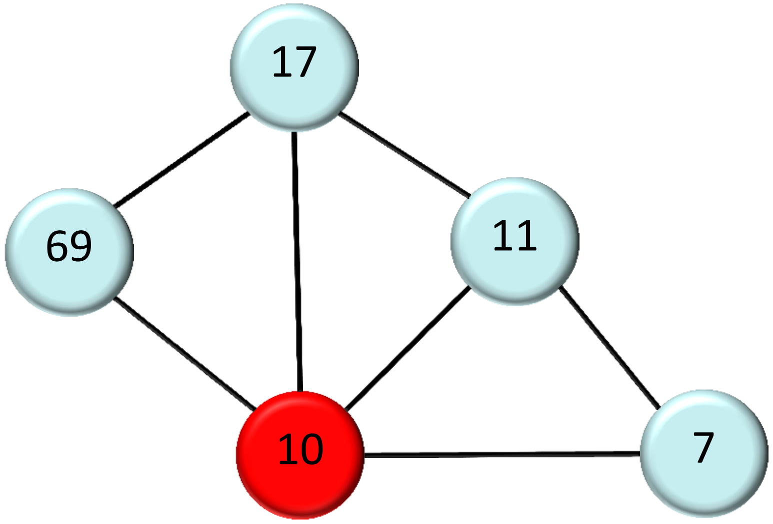
Example: Maximal Independent Set (MIS)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



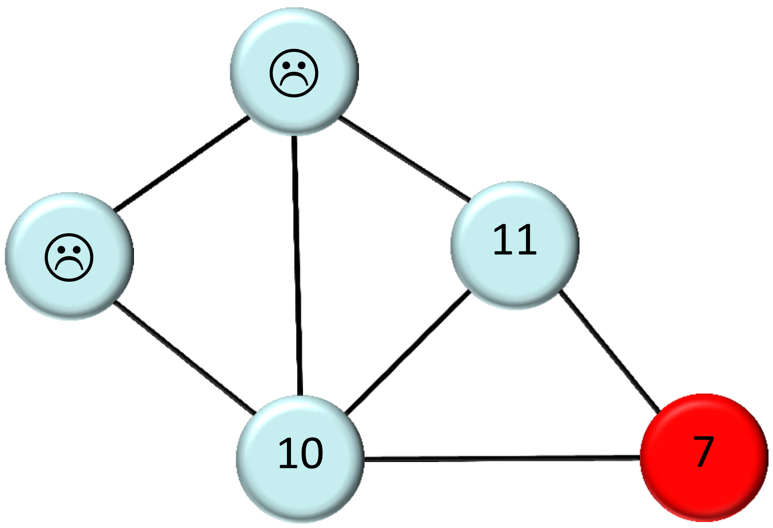
Example: Maximal Independent Set (MIS)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



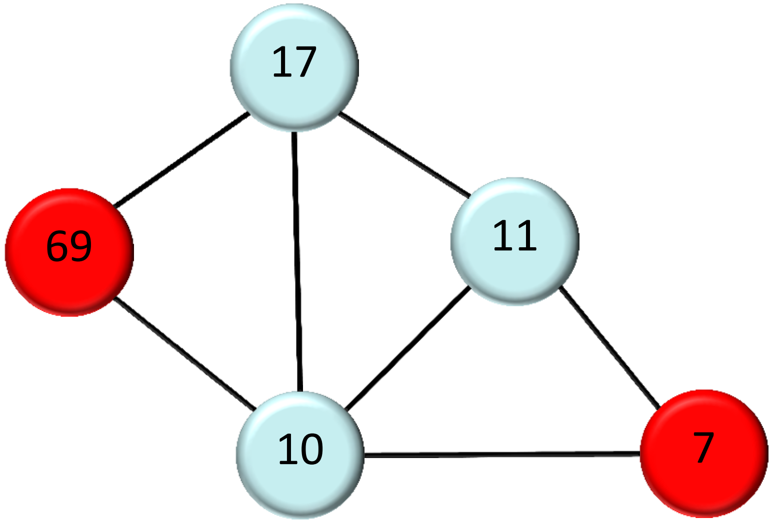
Example: Maximal Independent Set (MIS)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



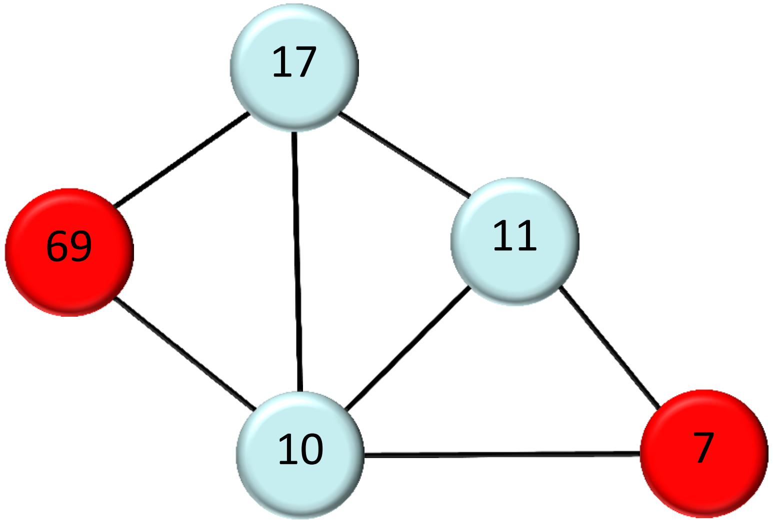
Example: Maximal Independent Set (MIS)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



Example: Maximal Independent Set (MIS)

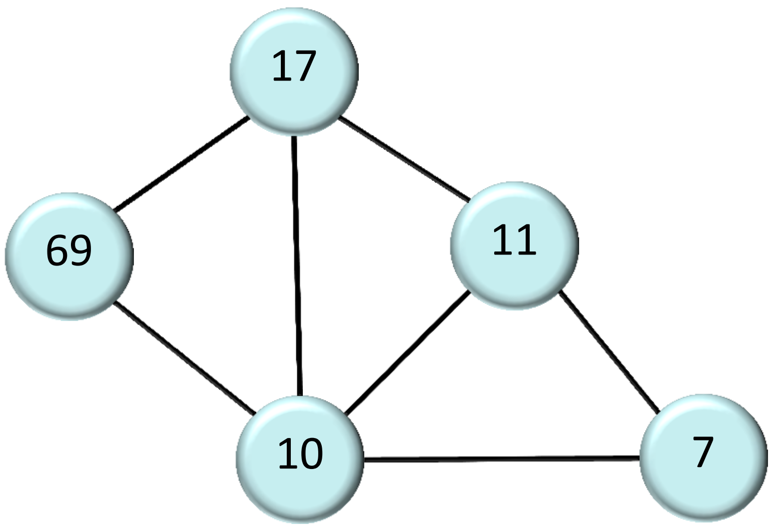
- Given a network with *n* nodes, nodes have unique IDs.
- Find a Maximal Independent Set (MIS)
  - a non-extendable set of pair-wise non-adjacent nodes



- Traditional (sequential) computation:
  The simple greedy algorithm finds MIS (in linear time)

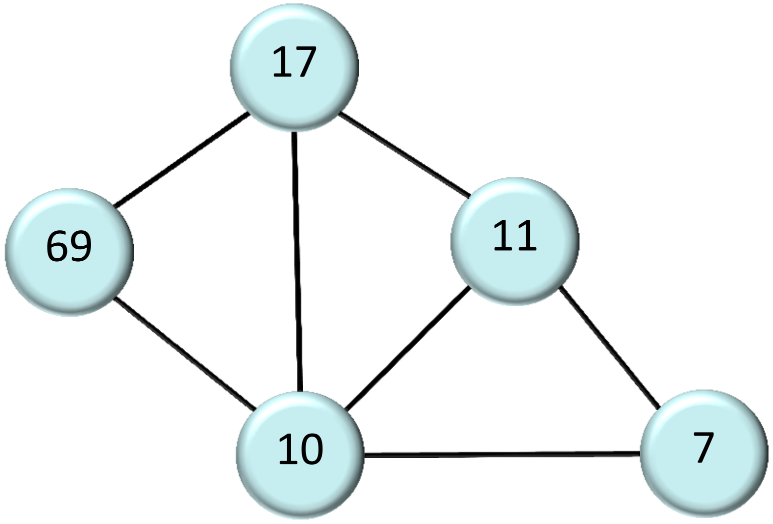
What about a Distributed Algorithm?

- Nodes are agents with unique ID’s that can communicate with neighbors by sending messages. In each synchronous round, every node can send a (different) message to each neighbor.



What about a Distributed Algorithm?

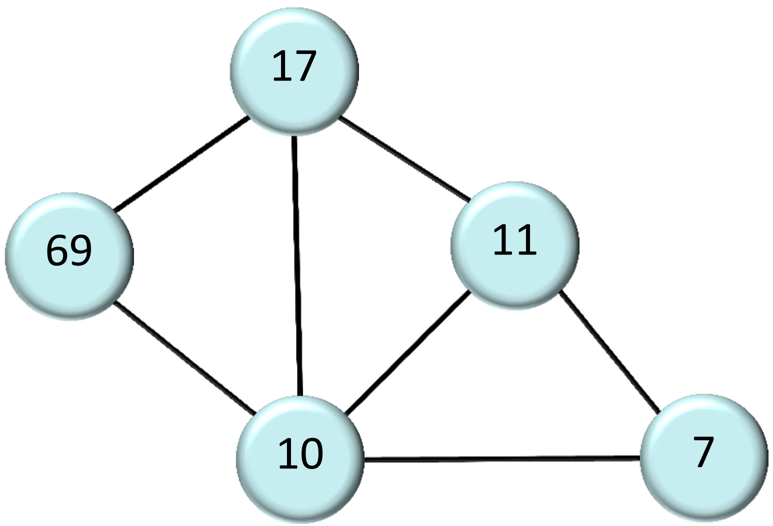
- Nodes are agents with unique ID's that can communicate with neighbors by sending messages. In each synchronous round, every node can send a (different) message to each neighbor.




```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

A Simple Distributed Algorithm

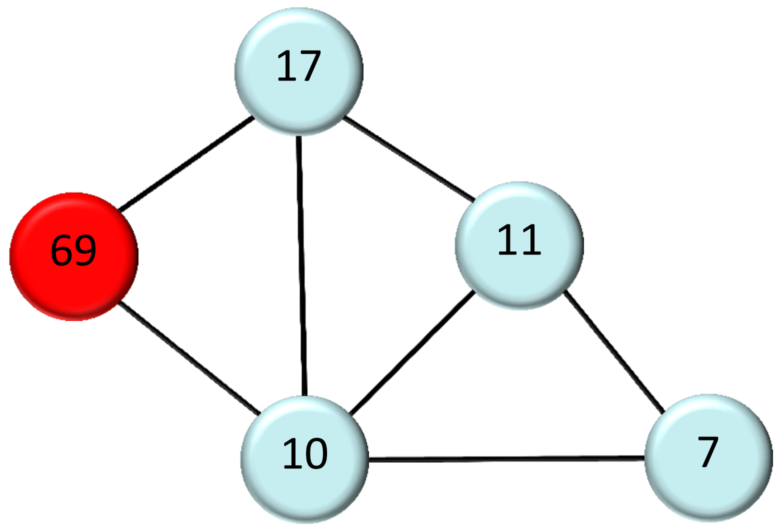
- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS




```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

A Simple Distributed Algorithm

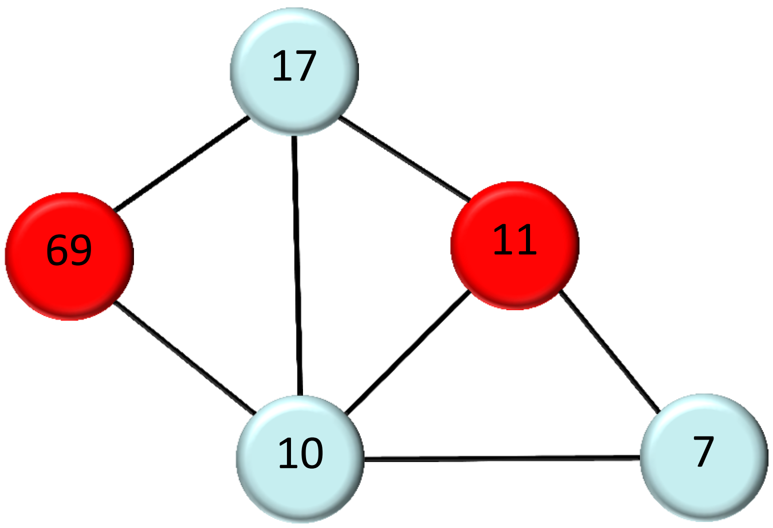
- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS




```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

A Simple Distributed Algorithm

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

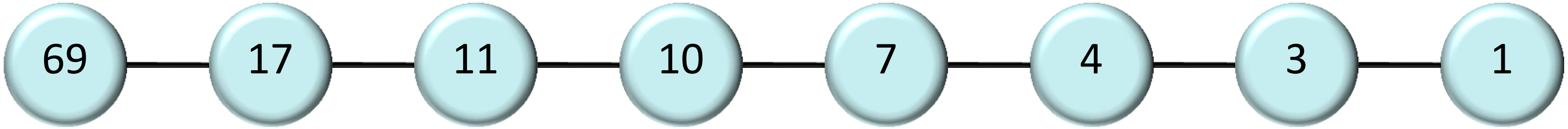



```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

- What's the problem with this distributed algorithm?

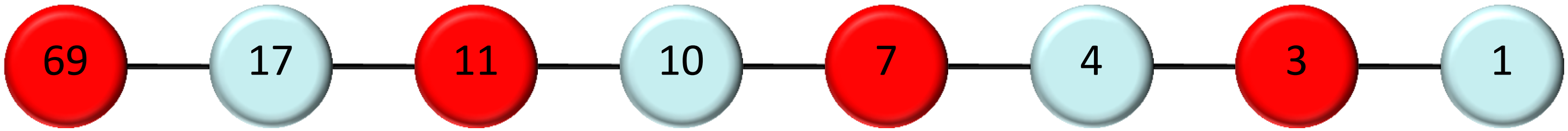
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS



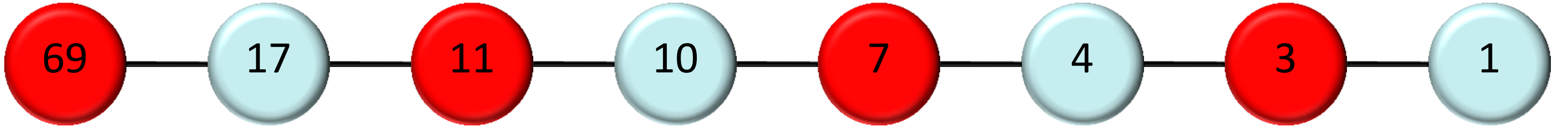
Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

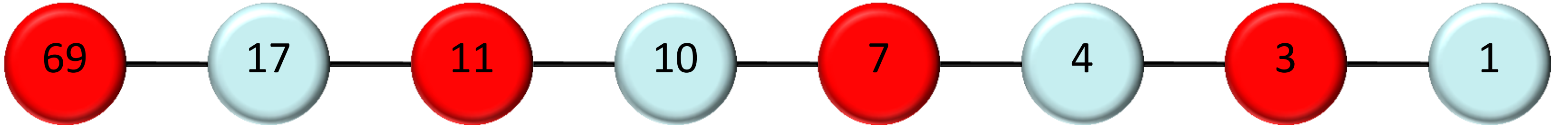


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

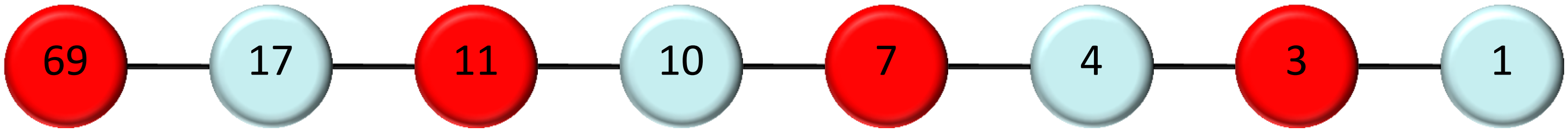


- What if we have minor changes?

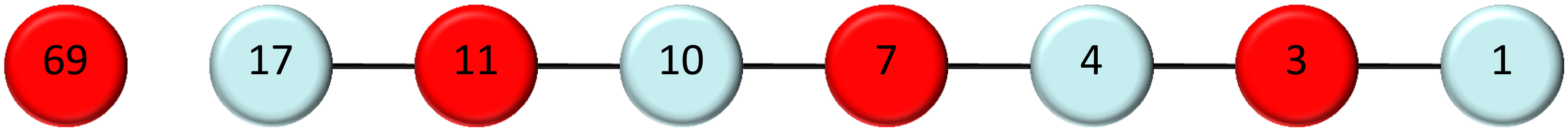


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

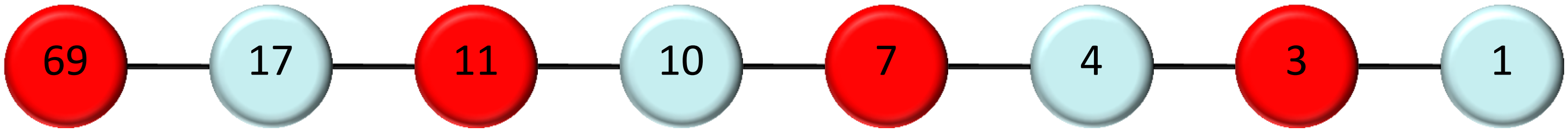


- What if we have minor changes?

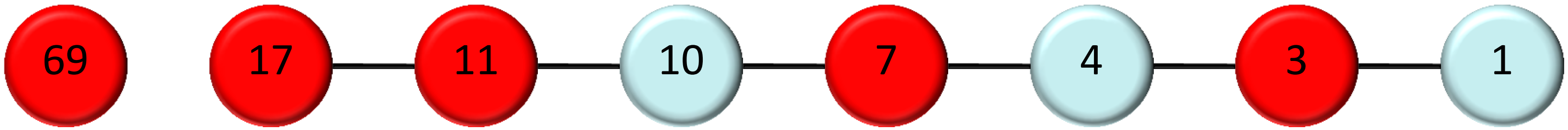


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

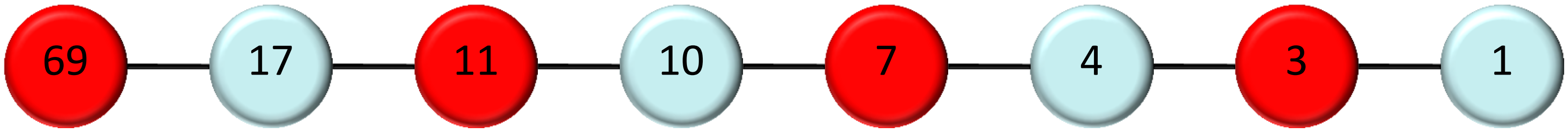


- What if we have minor changes?

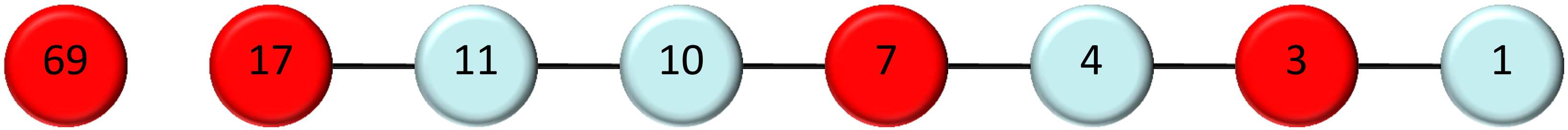


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

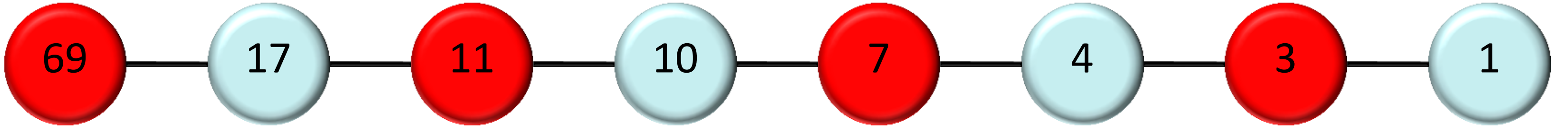


- What if we have minor changes?

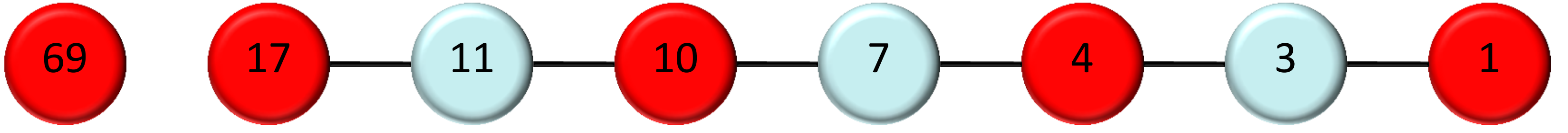


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS

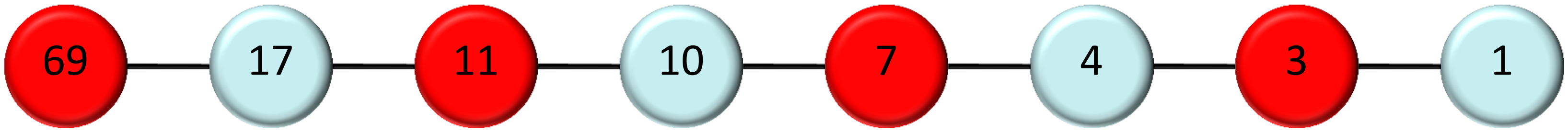


- What if we have minor changes?

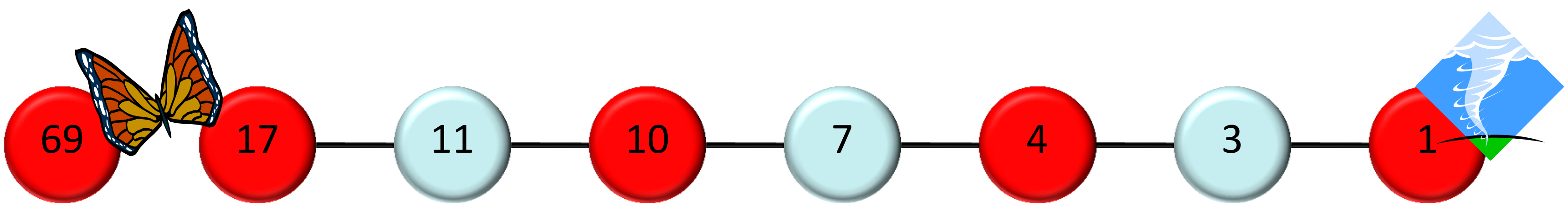


Example

- Wait until all neighbors with higher ID decided
- If no higher ID neighbor is in MIS → join MIS



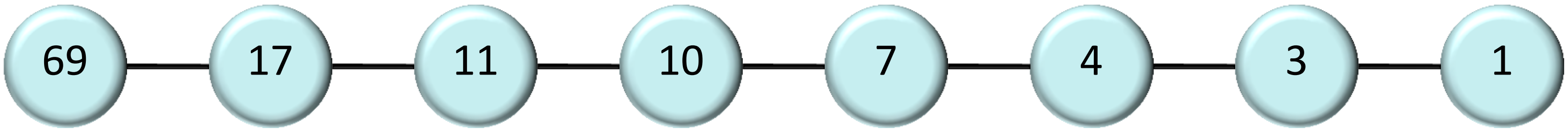
- What if we have minor changes?

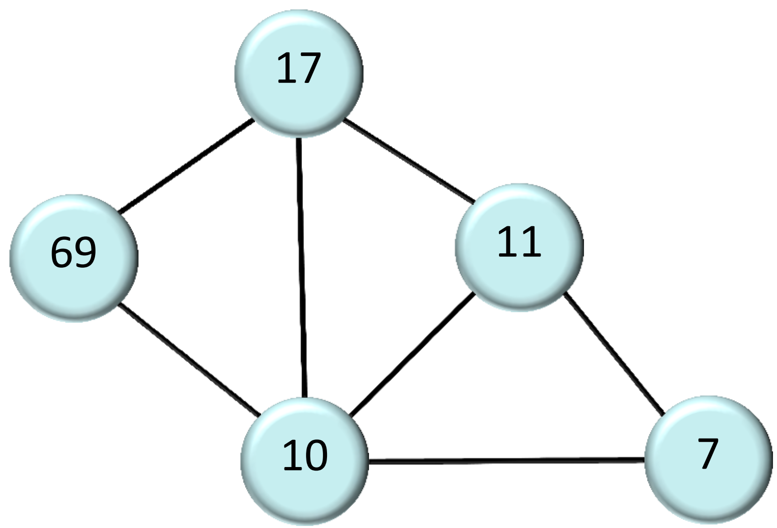


- Proof by animation: In the worst case, the algorithm is slow (linear in the number of nodes). In addition, we have a terrible „butterfly effect“.

What about a Fast Distributed Algorithm?

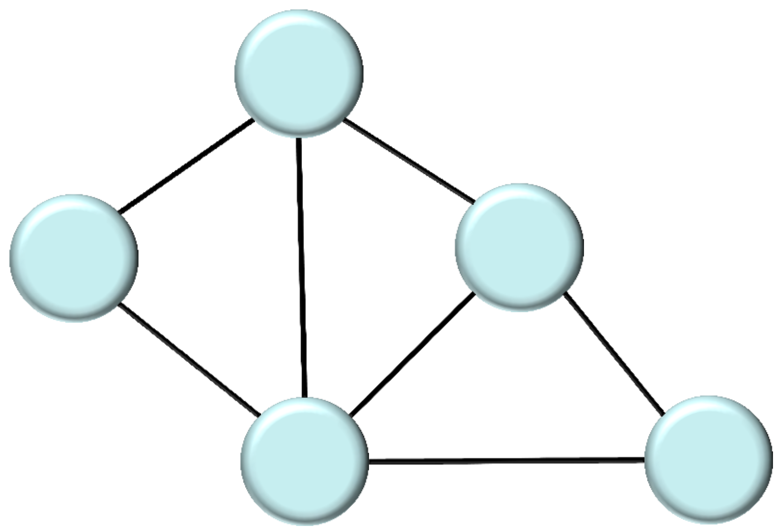
- Can you find a distributed algorithm that is polylogarithmic in the number of nodes *n*, for any graph?





What about a Fast Distributed Algorithm?

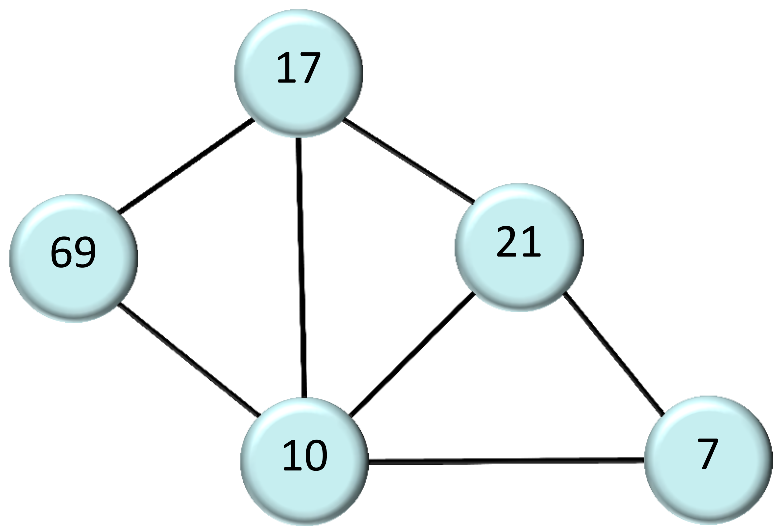
- Surprisingly, for deterministic distributed algorithms, this is an Open problem!
- However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!



What about a Fast Distributed Algorithm?

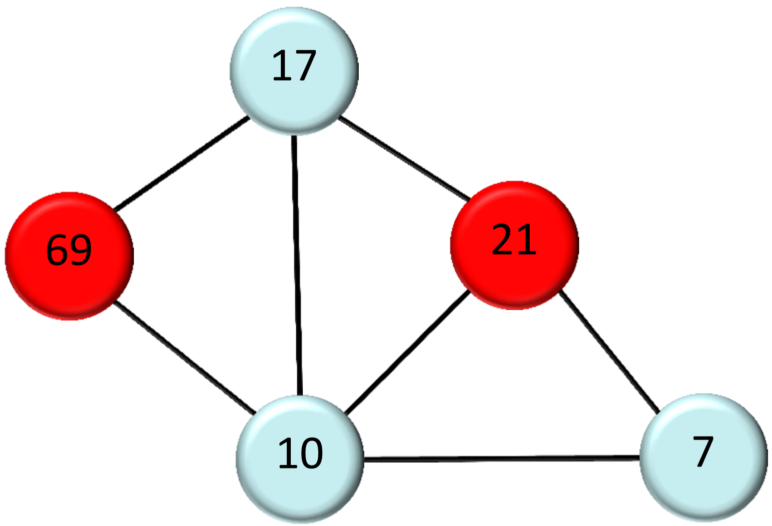
- Surprisingly, for deterministic distributed algorithms, this is an Open problem!

- However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!



What about a Fast Distributed Algorithm?

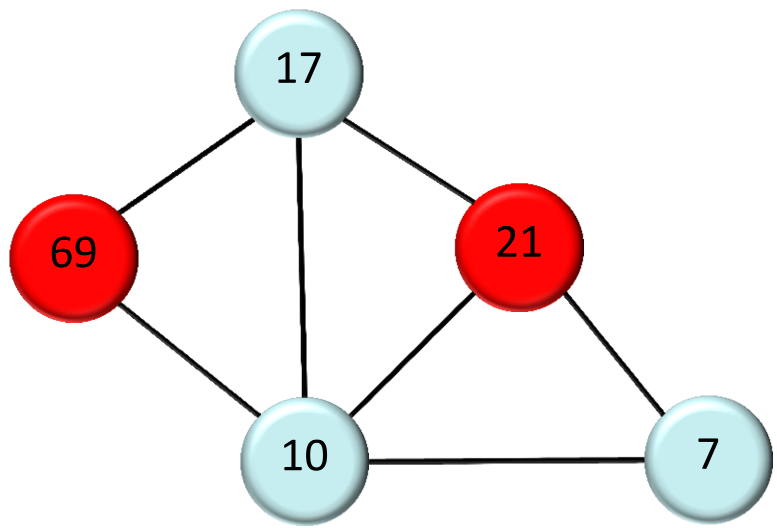
- Surprisingly, for deterministic distributed algorithms, this is an Open problem!
- However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!



What about a Fast Distributed Algorithm?

- Surprisingly, for deterministic distributed algorithms, this is an Open problem!

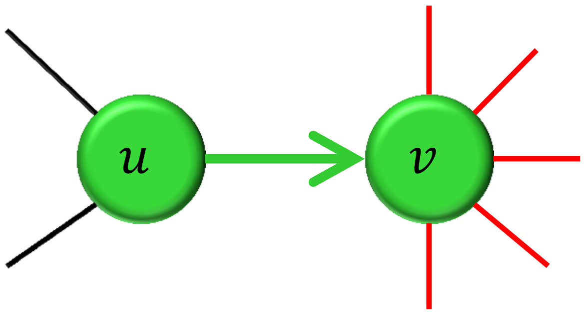
- However, randomization helps! In each synchronous round, nodes should choose a random value. If your value is larger than the value of your neighbors, join MIS!



- How many synchronous rounds does this take in expectation (or whp)?

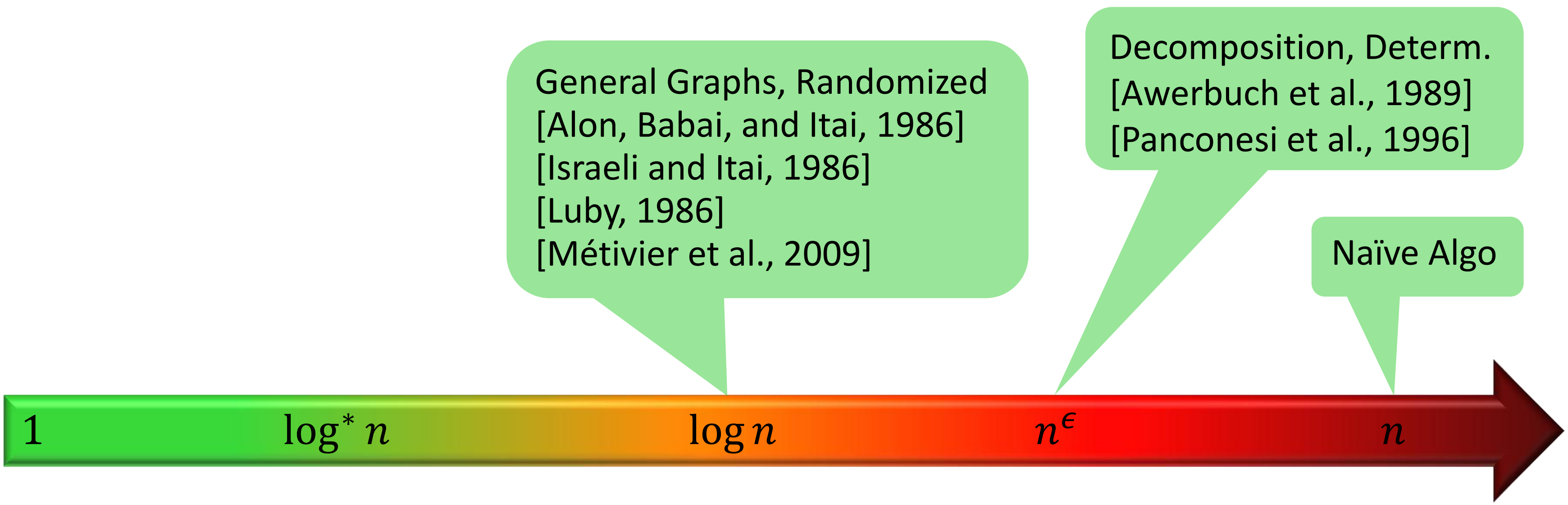
Analysis

- Event $(u \rightarrow v)$: node u got largest random value in combined neighborhood $N_u \cup N_v$.
- We only count edges of v as deleted.



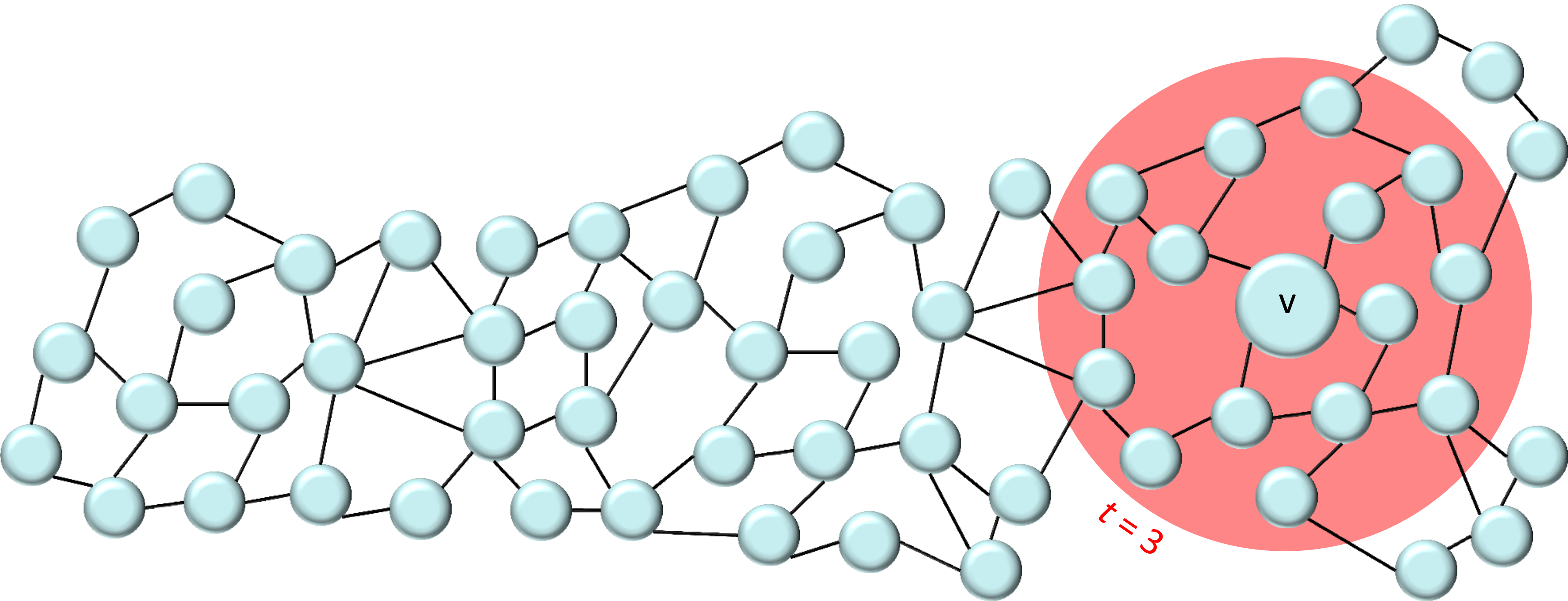
- Similarly event $(v \rightarrow u)$ deletes edges of u.
- We only double-counted edges.
- Using linearity of expectation, in expectation at least half of the edges are removed in each round.
- In other words, whp it takes $O(\log n)$ rounds to compute an MIS.

Results: MIS

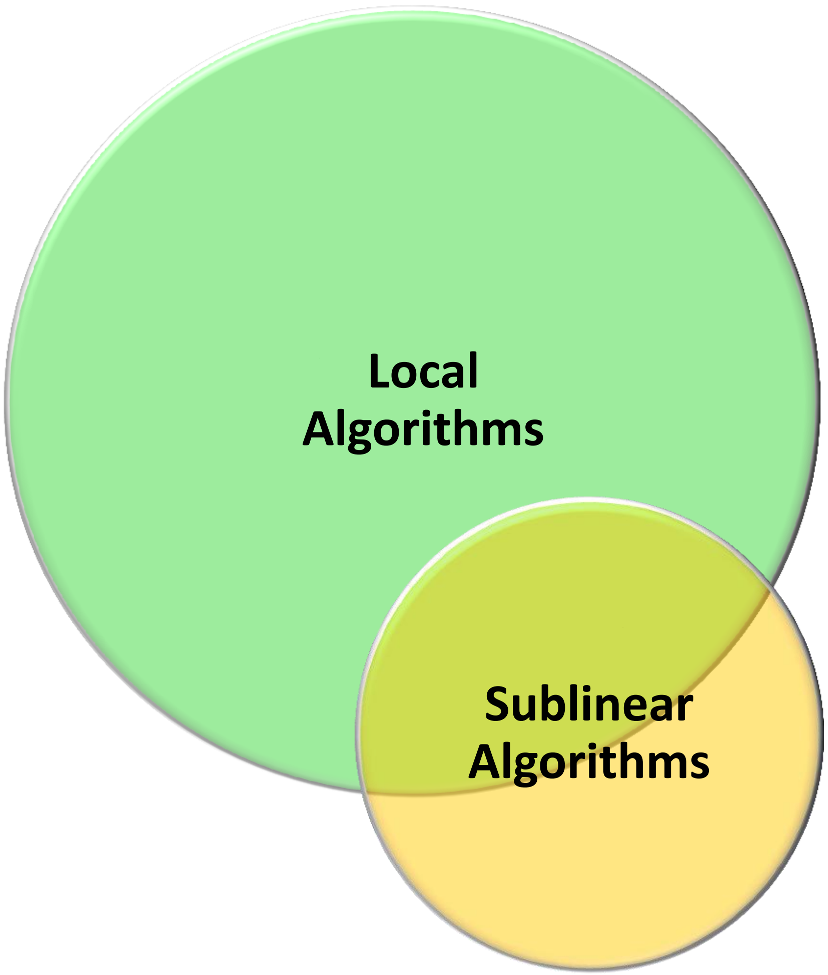

General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]
Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]
Naïve Algo
1
$\log^* n$
$\log n$
n^ϵ
n

Local Algorithms

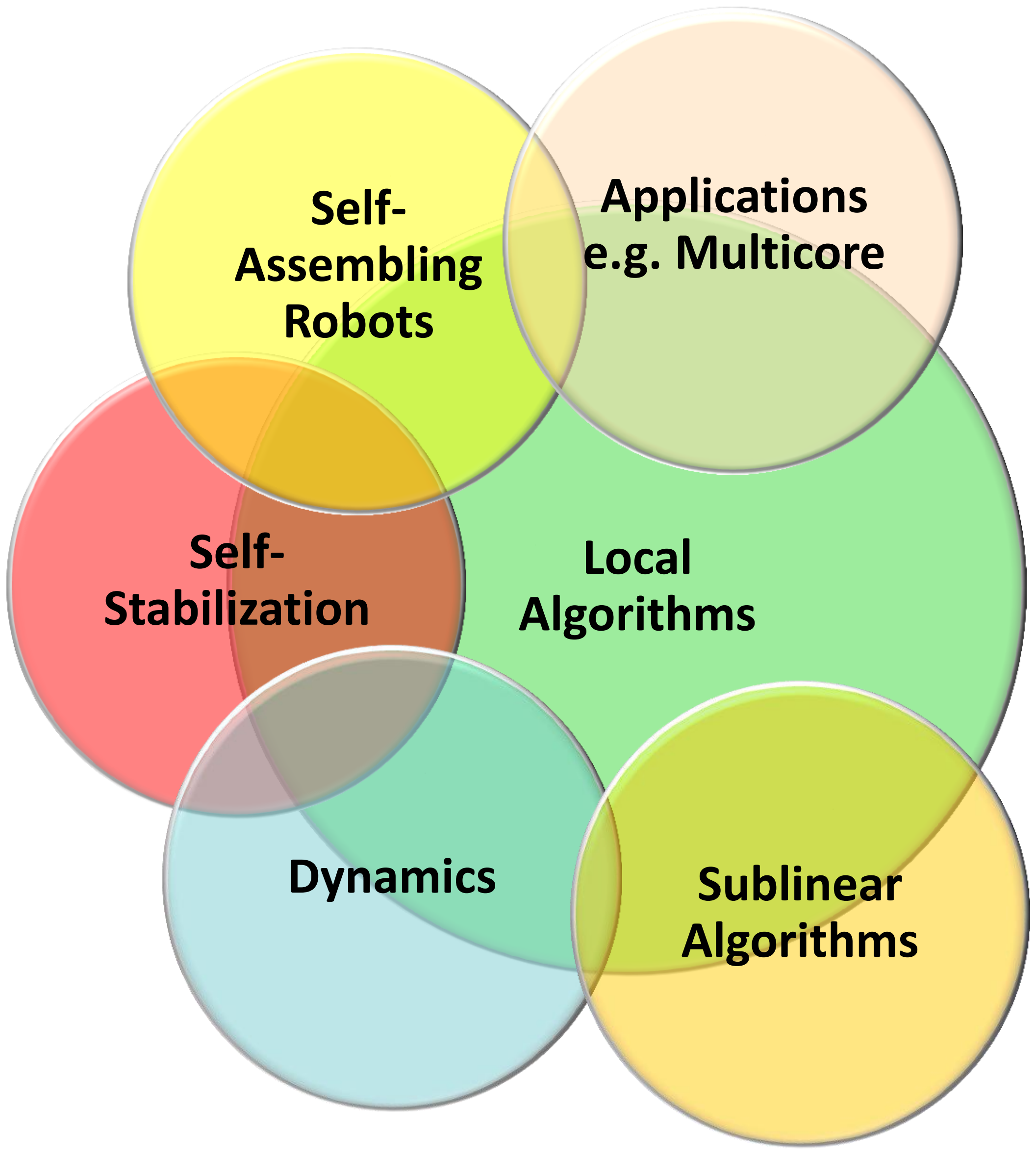
- Each node can exchange a message with all neighbors, for *t* communication rounds, and must then decide.
- Or: Given a graph, each node must determine its decision as a function of the information available within radius *t* of the node.
- Or: Change can only affect nodes up to distance *t*. 
- Or: …



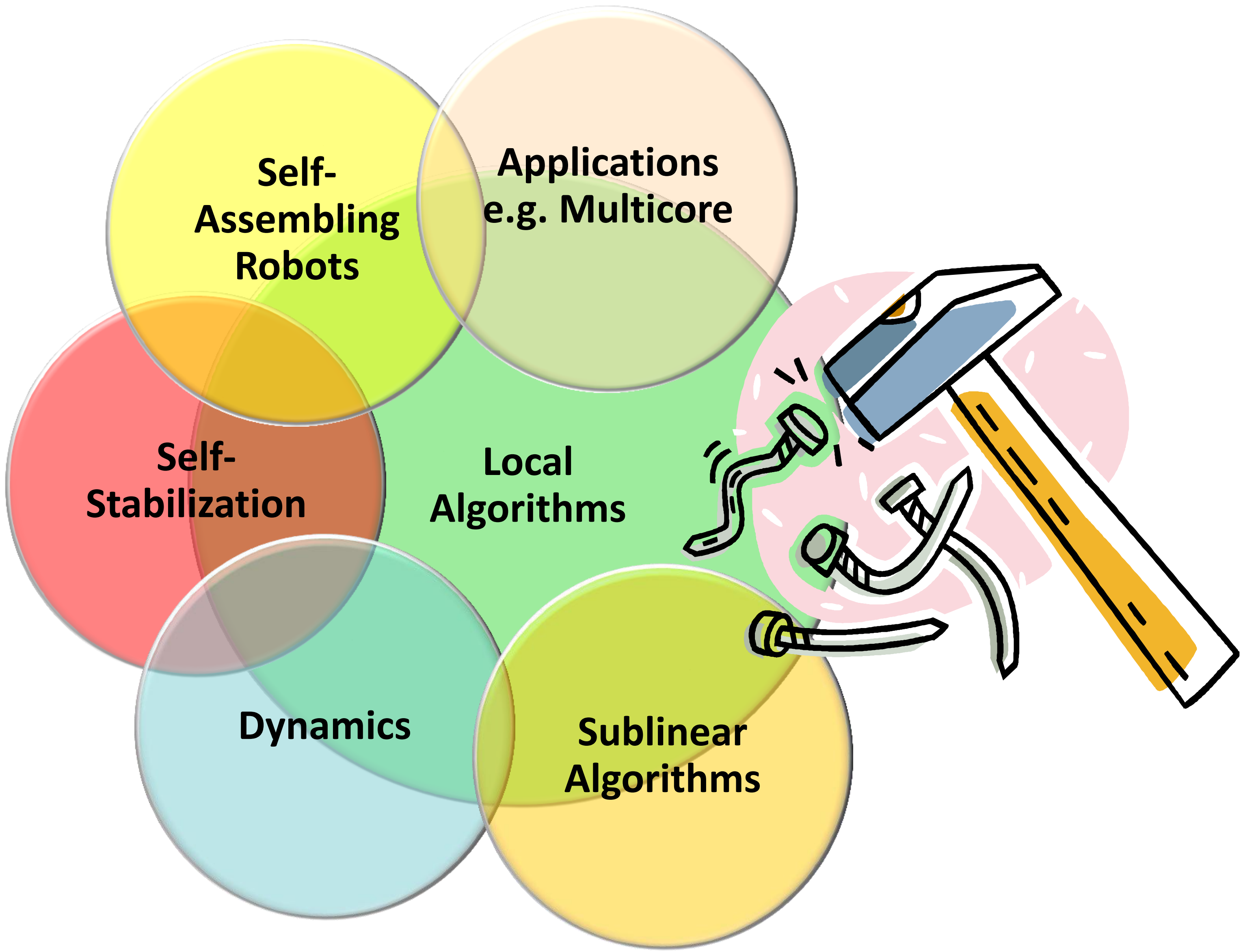
Locality

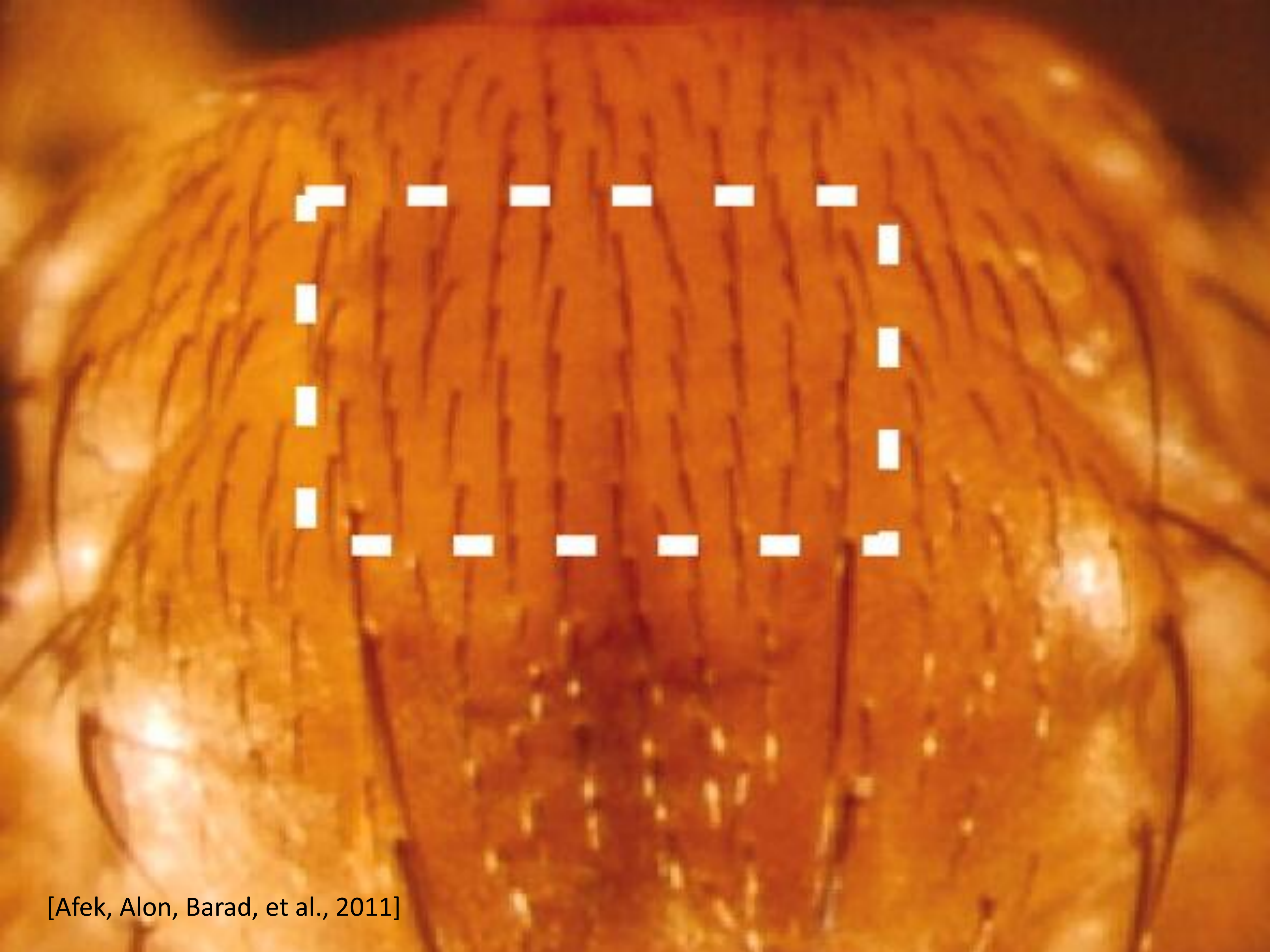


Locality is Everywhere!



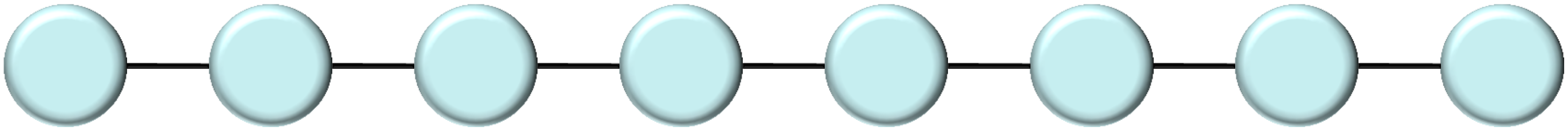
Locality is Everywhere!




[Afek, Alon, Barad, et al., 2011]

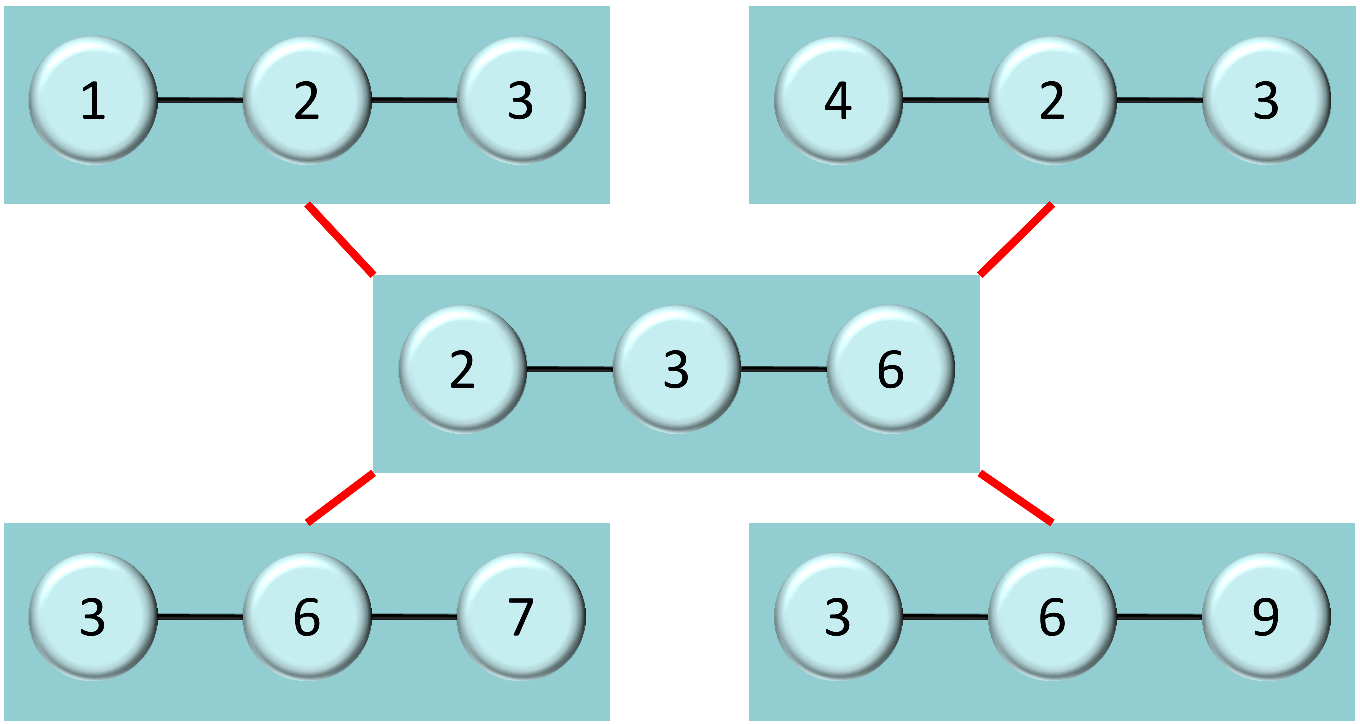
What about an Even Faster Distributed Algorithm?

- Since the 1980s, nobody was able to improve this simple algorithm.
- What about lower bounds?
- There is an interesting lower bound, essentially using a Ramsey theory argument, that proves that an MIS needs at least $\Omega(\log^* n)$ time.
  - log* is the so-called iterated logarithm – how often you need to take the logarithm until you end up with a value smaller than 1.
  - This lower bound already works on simple networks such as the linked list



Coloring Lower Bound on Oriented Ring

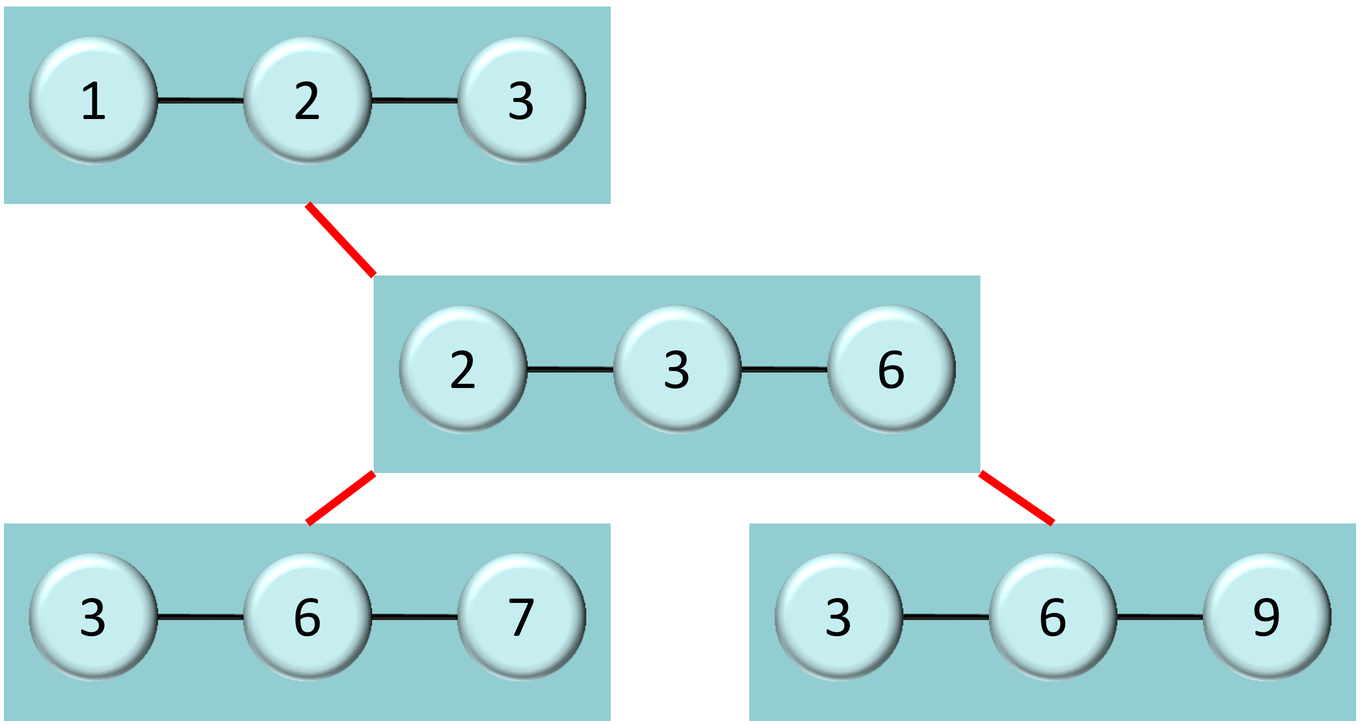
- Build graph G_t, where nodes are possible views of nodes for distributed algorithms of time t. Connect views that could be neighbors in ring.
- Here is for instance of G_1:



- Chromatic number of G_t is exactly minimum possible colors in time t.

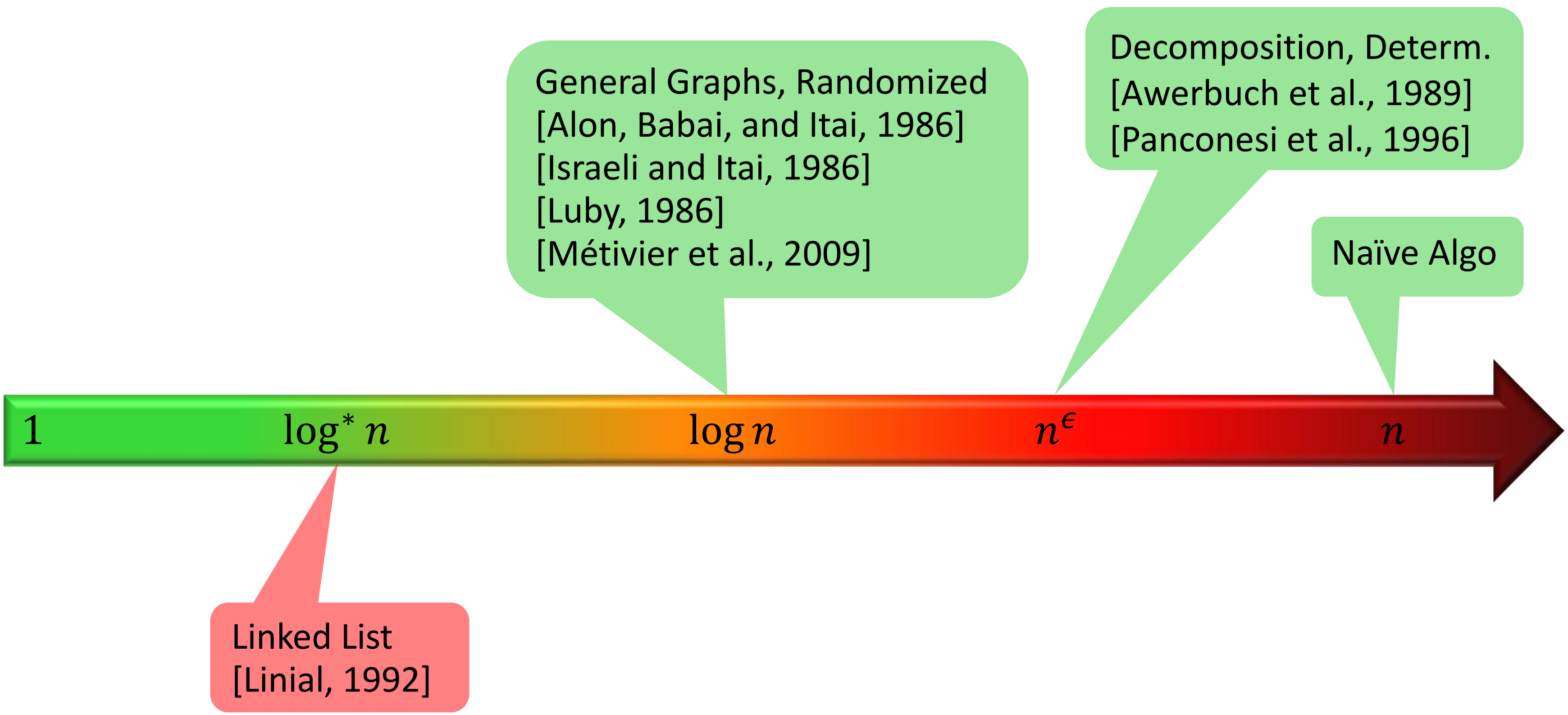
Coloring Lower Bound on Oriented Ring

- Build graph G_t, where nodes are possible views of nodes for distributed algorithms of time t. Connect views that could be neighbors in ring.
- Here is for instance of G_1:



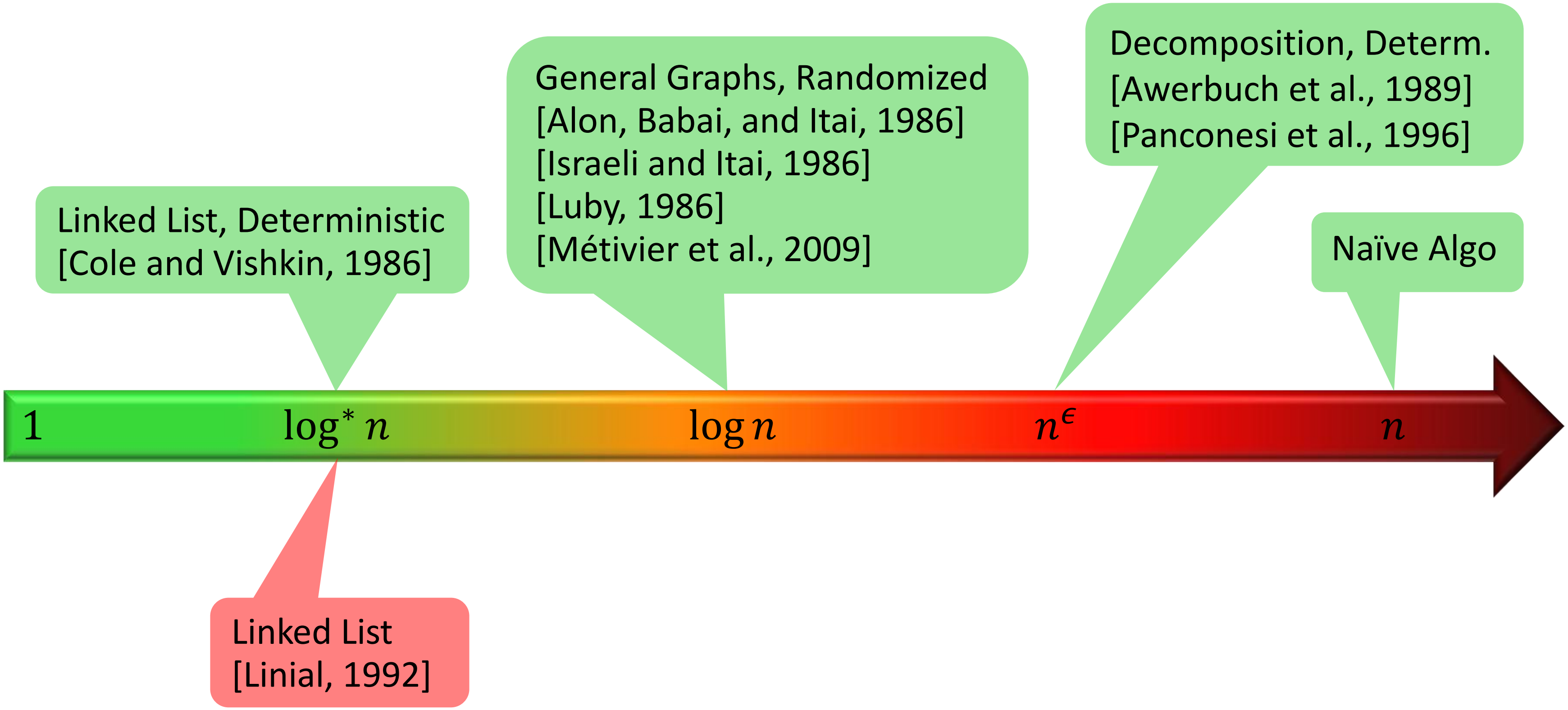
- Chromatic number of G_t is exactly minimum possible colors in time t.

Results: MIS



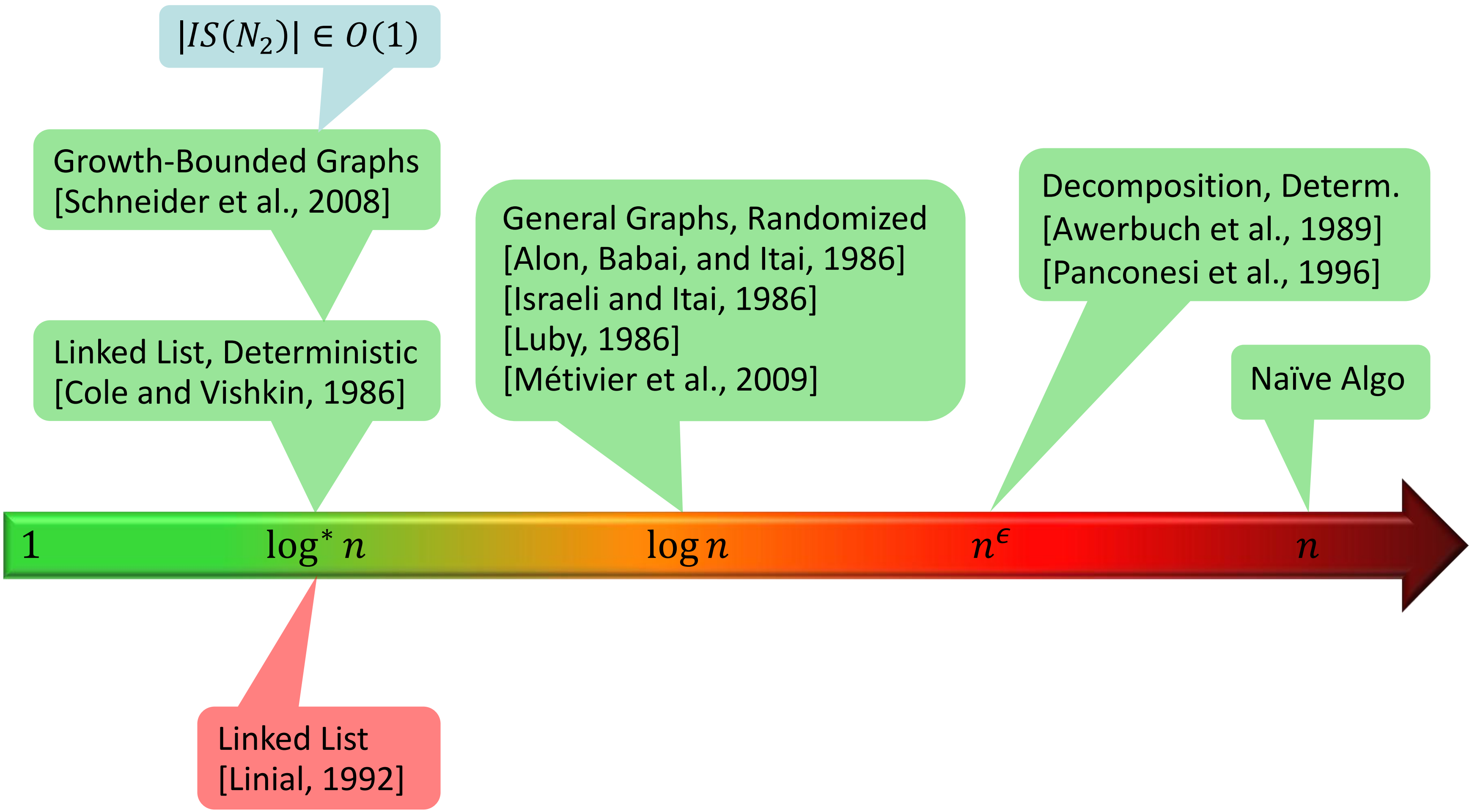
General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]
Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]
Naïve Algo
1
$\log^* n$
$\log n$
n^{ϵ}
n
Linked List
[Linial, 1992]

Results: MIS



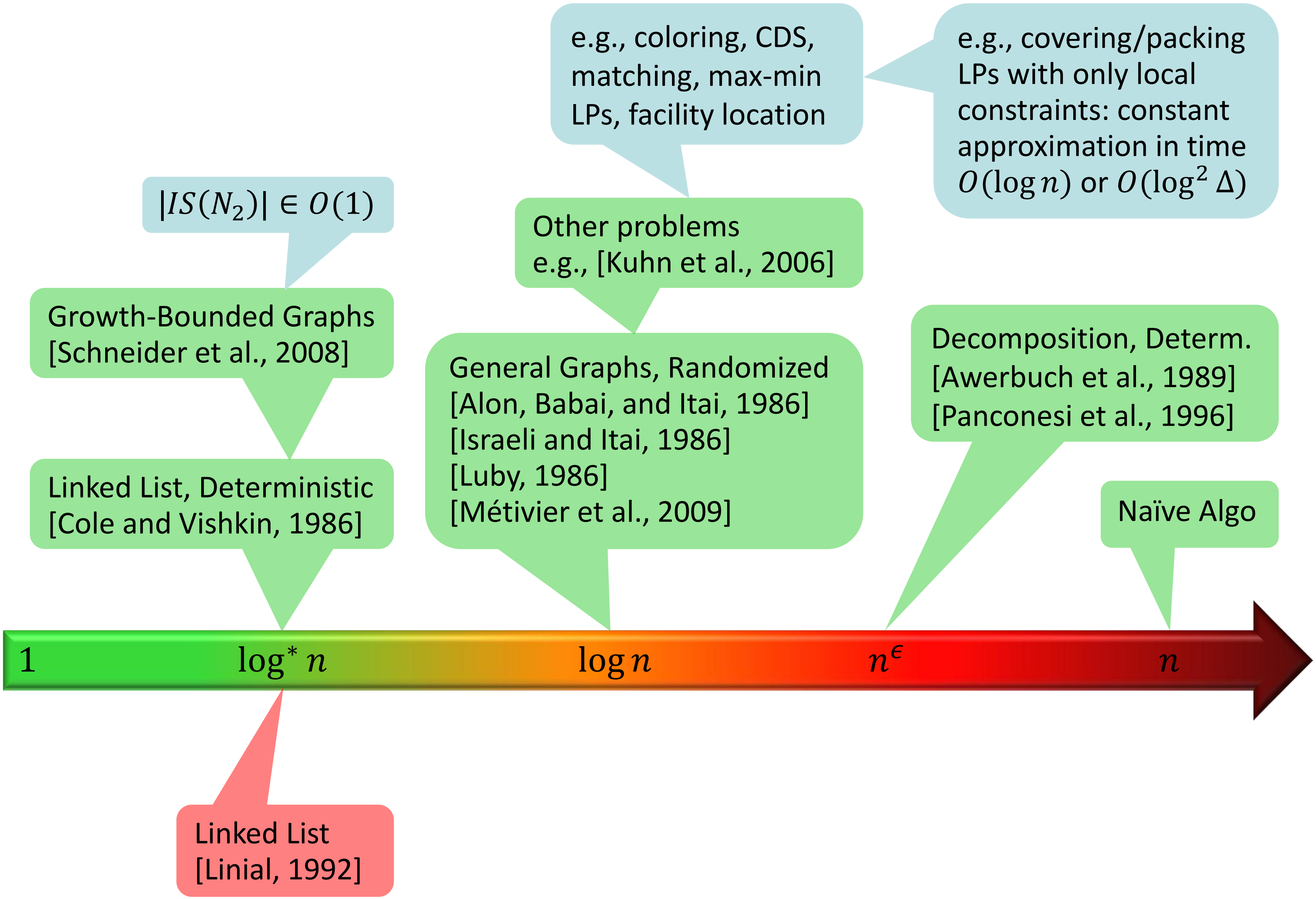
Linked List, Deterministic
[Cole and Vishkin, 1986]
General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]
Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]
Naïve Algo
1
$\log^* n$
$\log n$
n^ϵ
n
Linked List
[Linial, 1992]

Results: MIS

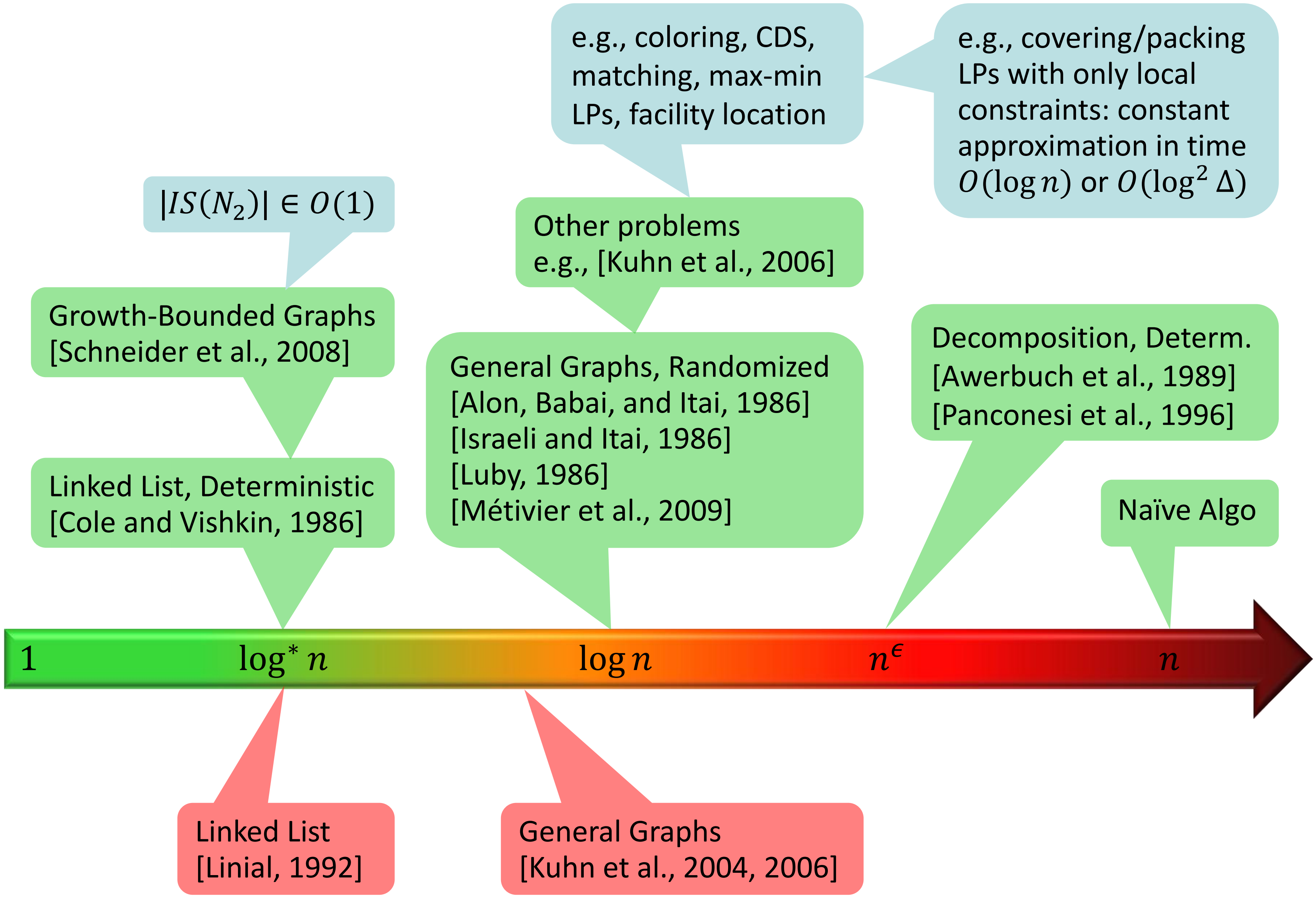


$|IS(N_2)| \in O(1)$
Growth-Bounded Graphs
[Schneider et al., 2008]
Linked List, Deterministic
[Cole and Vishkin, 1986]
General Graphs, Randomized
[Alon, Babai, and Itai, 1986]
[Israeli and Itai, 1986]
[Luby, 1986]
[Métivier et al., 2009]
Decomposition, Determ.
[Awerbuch et al., 1989]
[Panconesi et al., 1996]
Naïve Algo
1
$\log^* n$
$\log n$
n^ϵ
n
Linked List
[Linial, 1992]

Results: MIS


e.g., coloring, CDS, matching, max-min LPs, facility location
e.g., covering/packing LPs with only local constraints: constant approximation in time $O(\log n)$ or $O(\log^2 \Delta)$
$|IS(N_2)| \in O(1)$
Other problems e.g., [Kuhn et al., 2006]
Growth-Bounded Graphs [Schneider et al., 2008]
General Graphs, Randomized [Alon, Babai, and Itai, 1986] [Israeli and Itai, 1986] [Luby, 1986] [Métivier et al., 2009]
Decomposition, Determ. [Awerbuch et al., 1989] [Panconesi et al., 1996]
Linked List, Deterministic [Cole and Vishkin, 1986]
Naïve Algo
1
$\log^* n$
$\log n$
n^ϵ
n
Linked List [Linial, 1992]

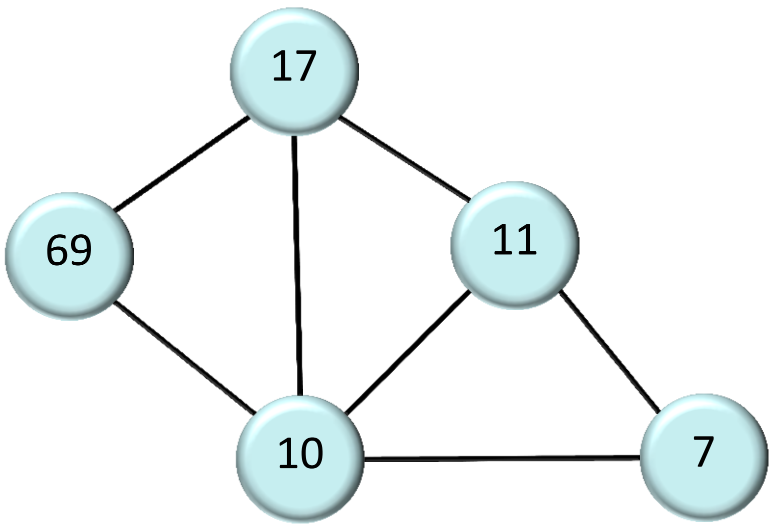
Results: MIS



e.g., coloring, CDS, matching, max-min LPs, facility location
e.g., covering/packing LPs with only local constraints: constant approximation in time $O(\log n)$ or $O(\log^2 \Delta)$
$|IS(N_2)| \in O(1)$
Other problems e.g., [Kuhn et al., 2006]
Growth-Bounded Graphs [Schneider et al., 2008]
General Graphs, Randomized [Alon, Babai, and Itai, 1986] [Israeli and Itai, 1986] [Luby, 1986] [Métivier et al., 2009]
Decomposition, Determ. [Awerbuch et al., 1989] [Panconesi et al., 1996]
Linked List, Deterministic [Cole and Vishkin, 1986]
Naïve Algo
1
$\log^* n$
$\log n$
n^{ϵ}
n
Linked List [Linial, 1992]
General Graphs [Kuhn et al., 2004, 2006]

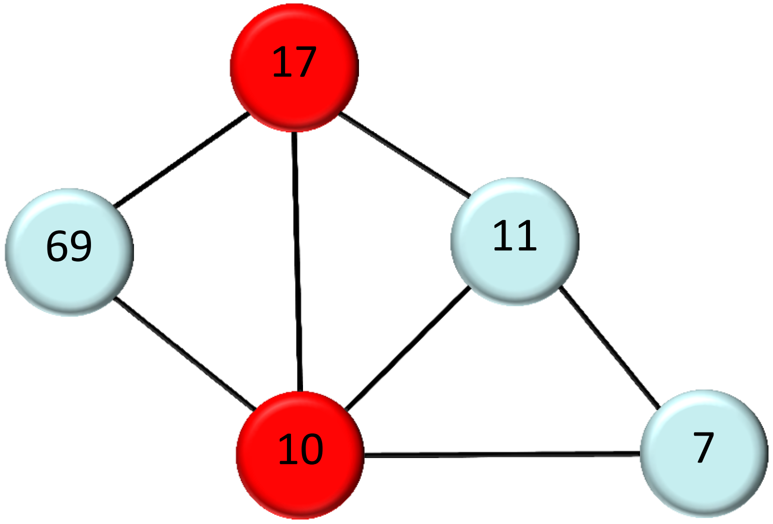
Example: Minimum Vertex Cover (MVC)

- Given a network with *n* nodes, nodes have unique IDs.
- Find a Minimum Vertex Cover (MVC)
  - a minimum set of nodes such that all edges are adjacent to node in MVC



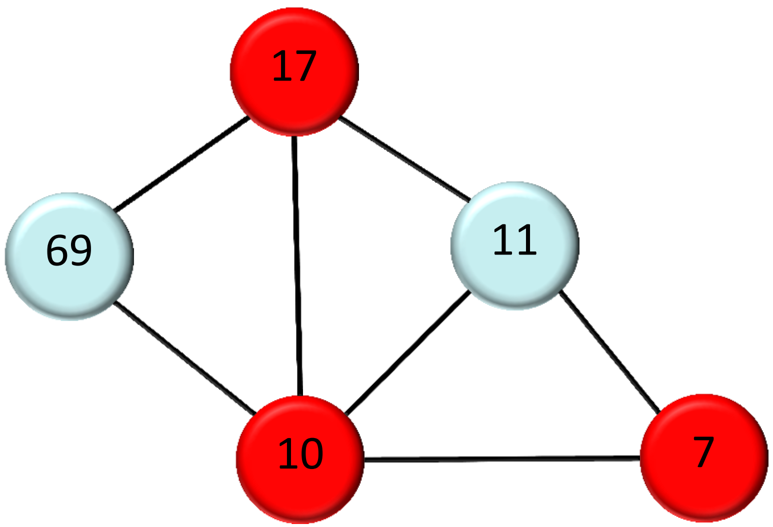
Example: Minimum Vertex Cover (MVC)

- Given a network with n nodes, nodes have unique IDs.
- Find a Minimum Vertex Cover (MVC)
  - a minimum set of nodes such that all edges are adjacent to node in MVC



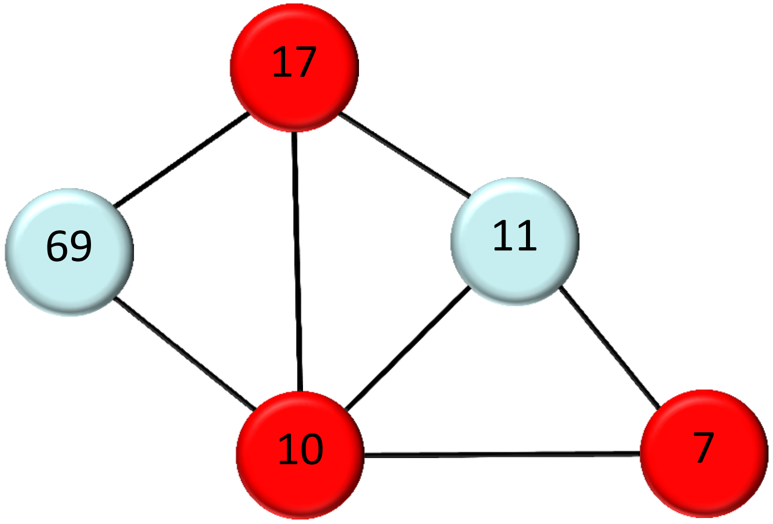
Example: Minimum Vertex Cover (MVC)

- Given a network with n nodes, nodes have unique IDs.
- Find a Minimum Vertex Cover (MVC)
  - a minimum set of nodes such that all edges are adjacent to node in MVC



Differences between MIS and MVC

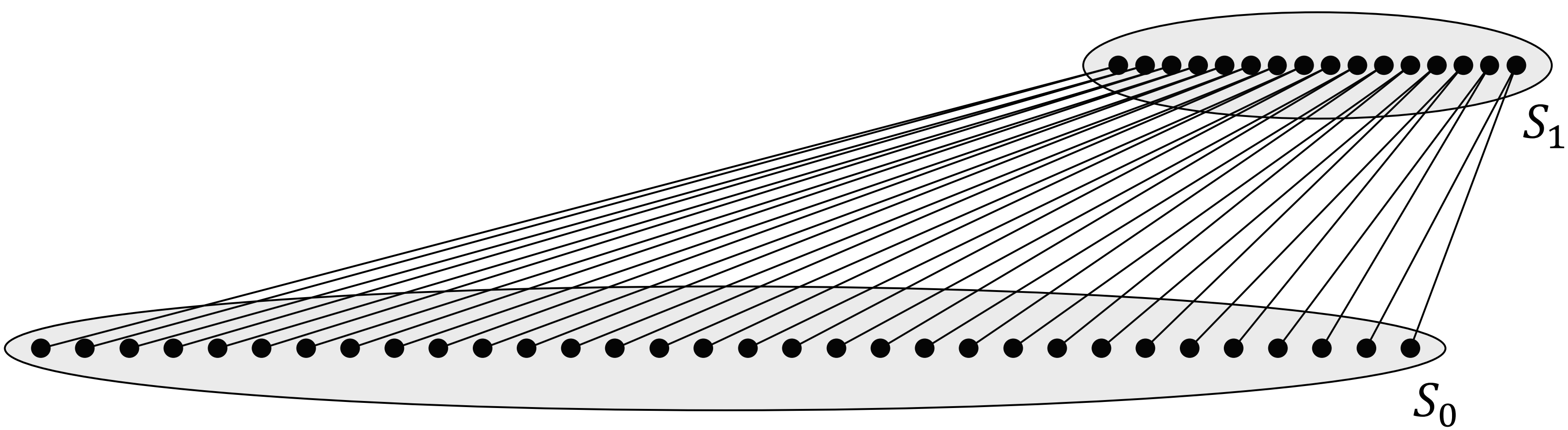
- Central (non-local) algorithms: MIS is trivial, whereas MVC is NP-hard
- Instead: Find an MVC that is “close” to minimum (approximation)
- Trade-off between time complexity and approximation ratio



- MVC: Various simple (non-distributed) 2-approximations exist!
- What about distributed algorithms?!?

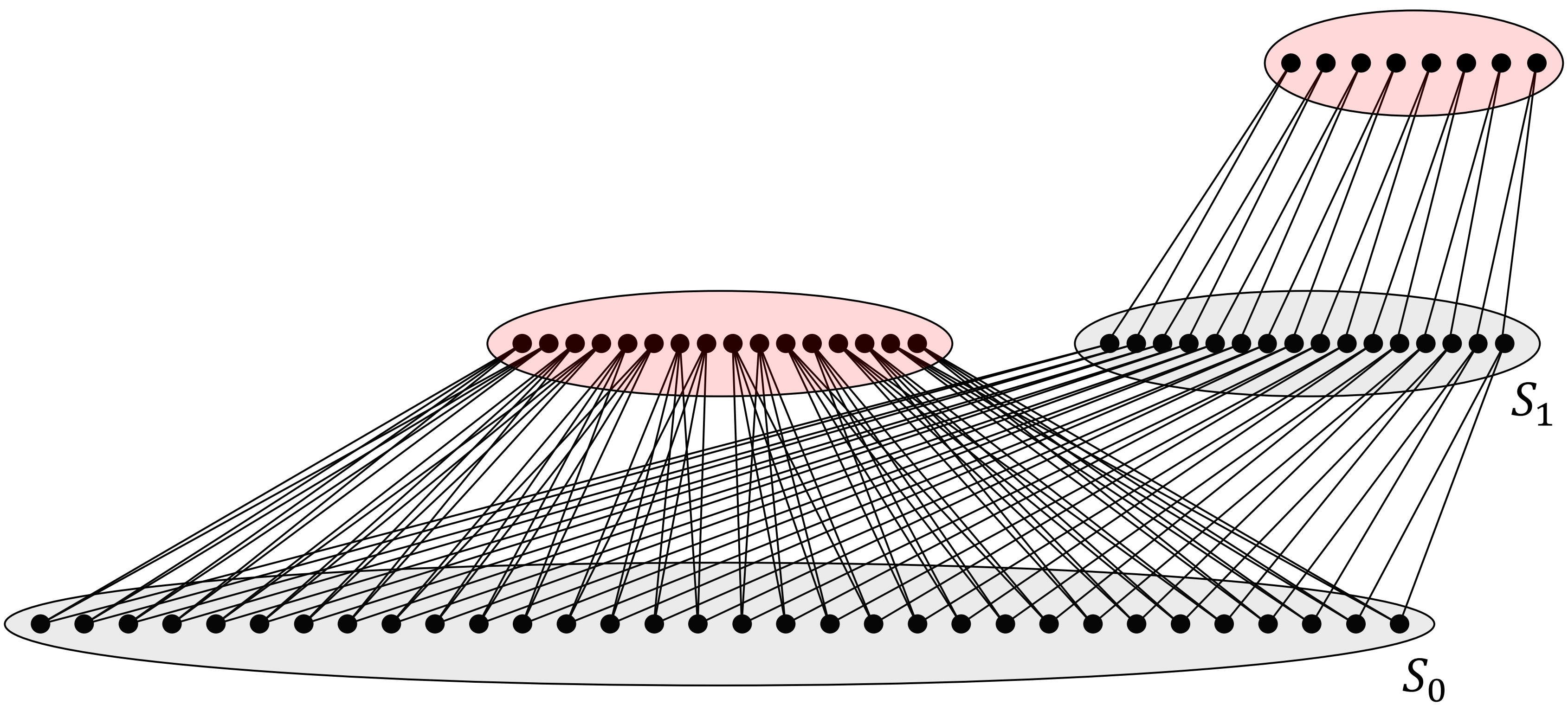
Finding the MVC (by Distributed Algorithm)

- Given the following bipartite graph with $|S_0| = \delta\ |S_1|$
- The MVC is just all the nodes in S_1
- Distributed Algorithm…



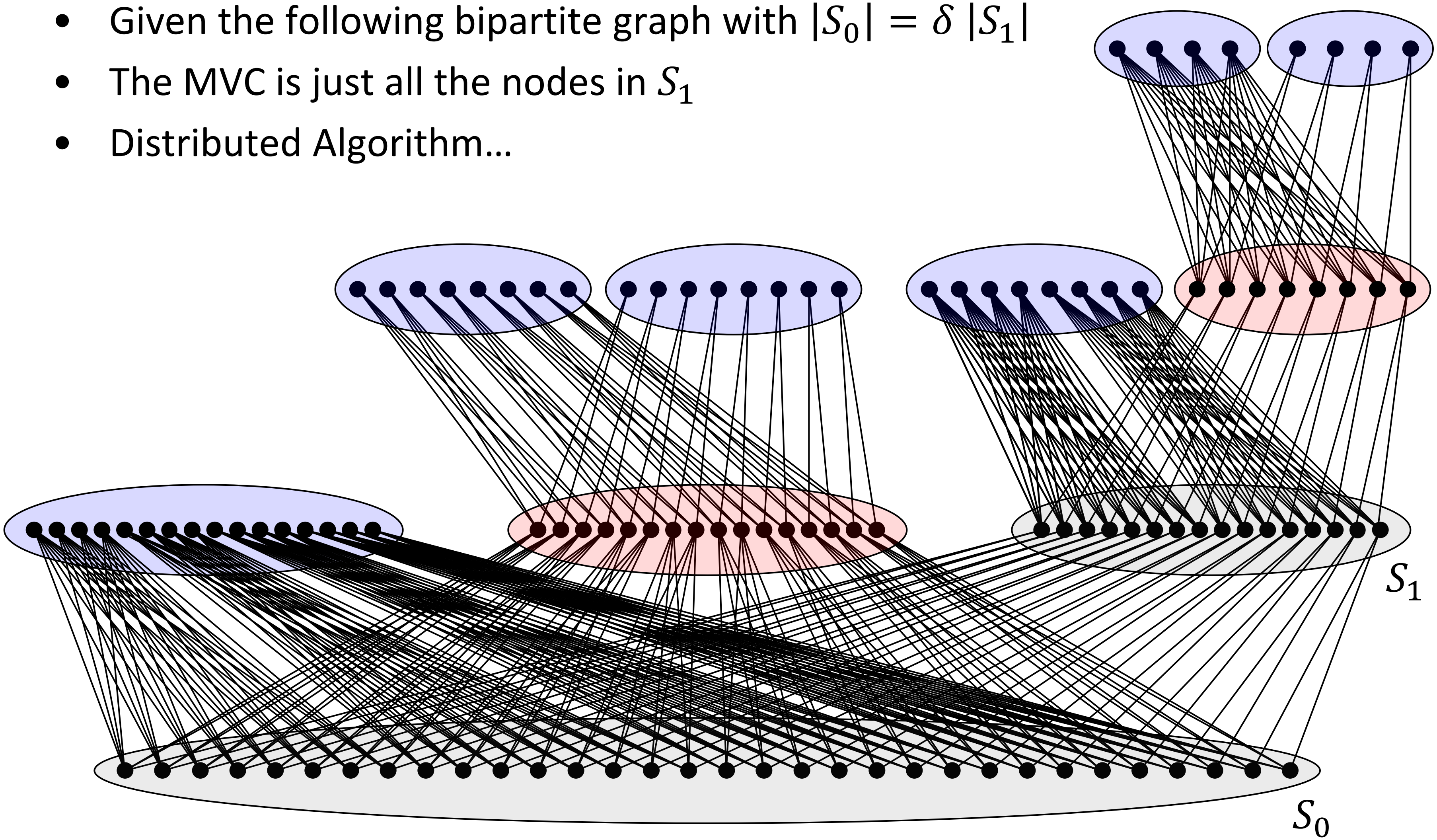
Finding the MVC (by Distributed Algorithm)

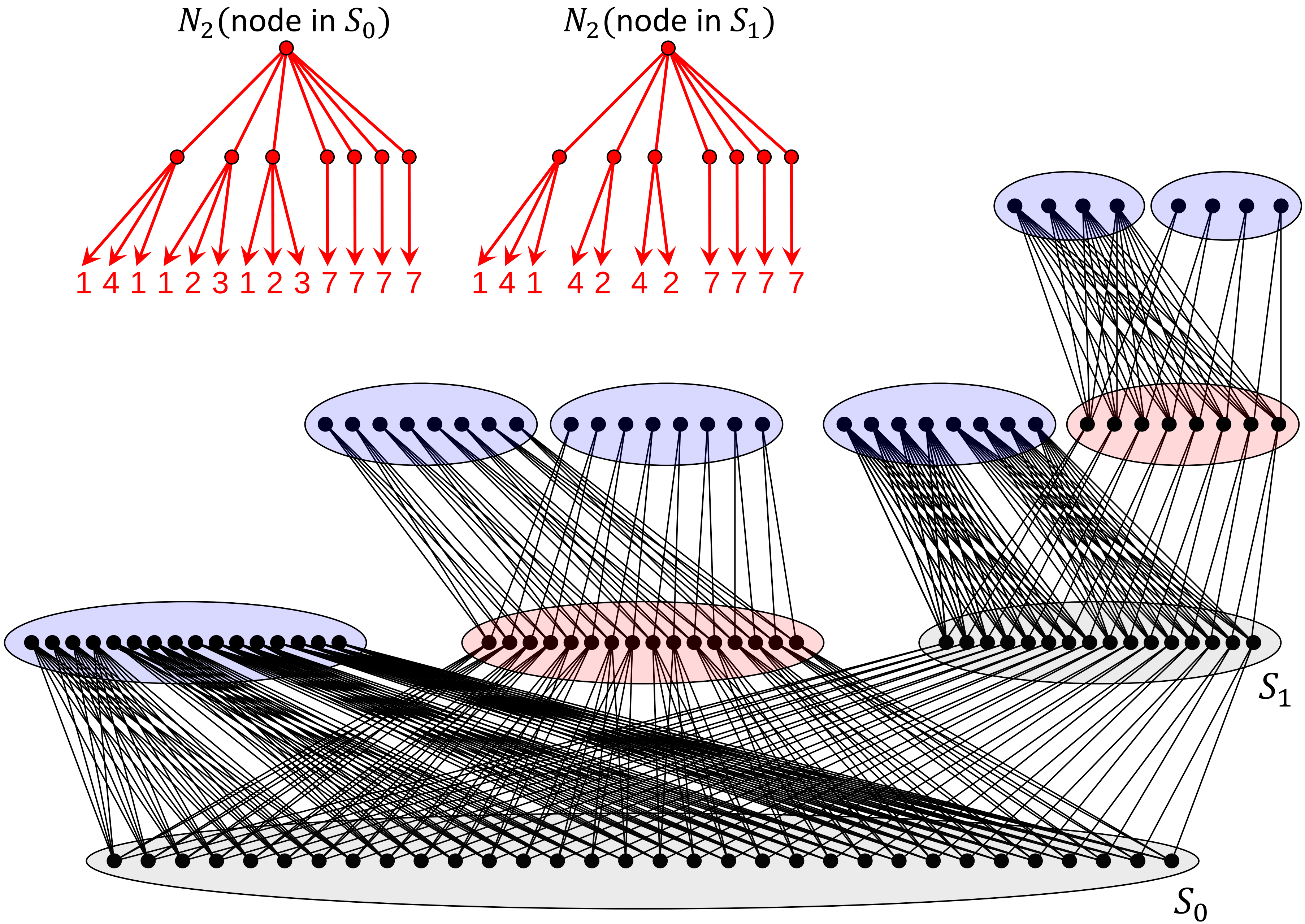
- Given the following bipartite graph with $|S_0| = \delta\ |S_1|$
- The MVC is just all the nodes in S_1
- Distributed Algorithm…



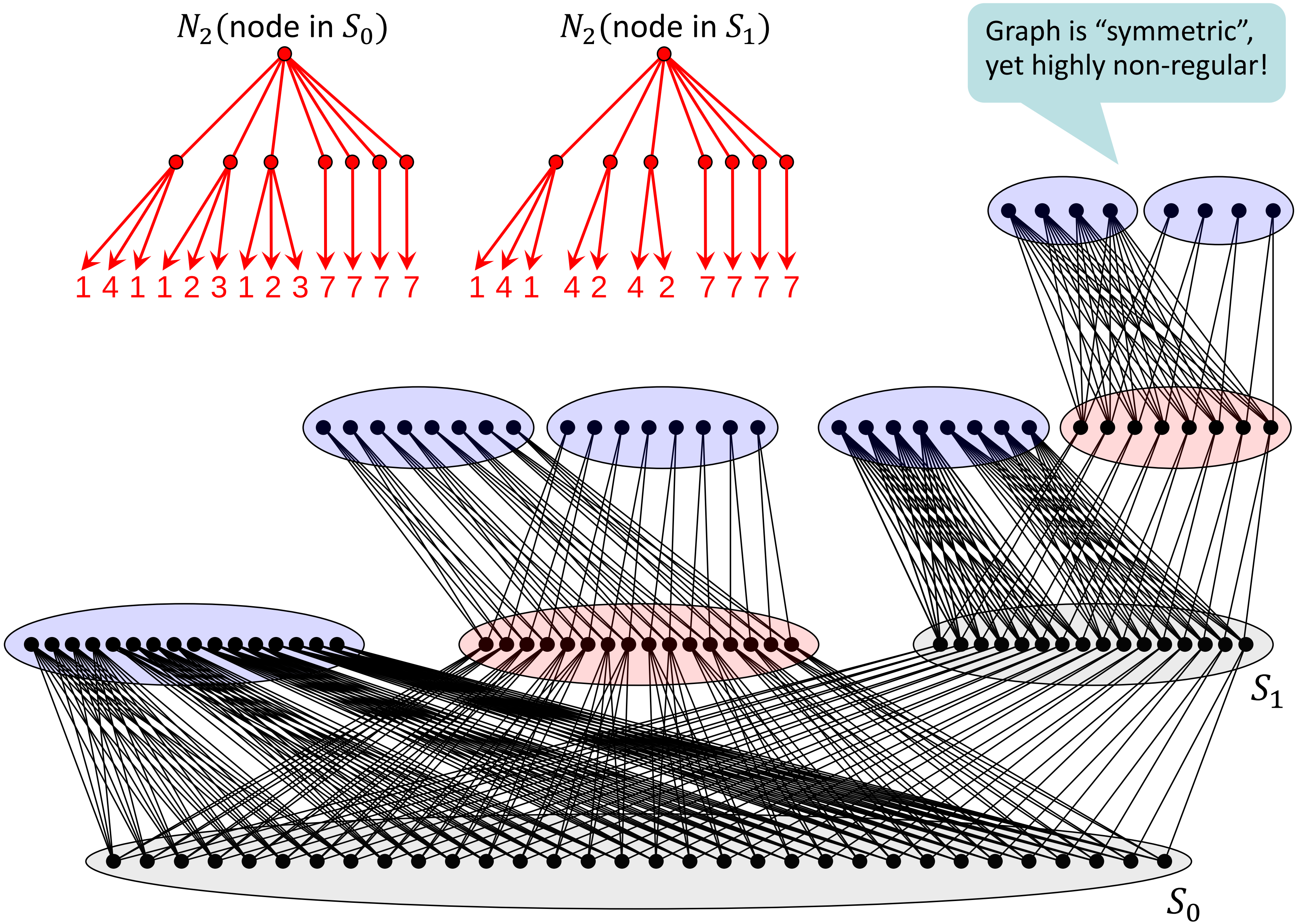
Finding the MVC (by Distributed Algorithm)

- Given the following bipartite graph with $|S_0| = \delta\ |S_1|$
- The MVC is just all the nodes in S_1
- Distributed Algorithm…



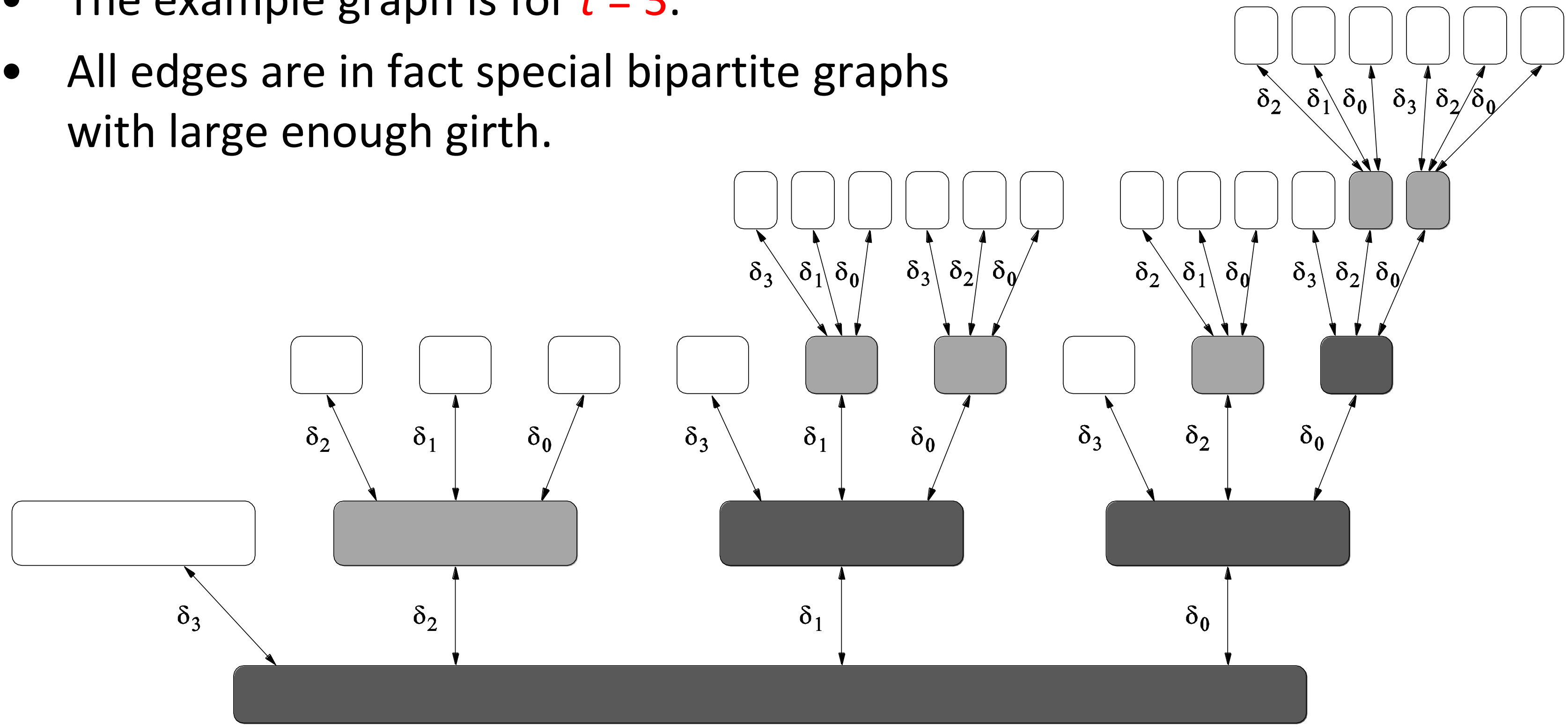


N_2(node in S_0)
1 4 1 1 2 3 1 2 3 7 7 7 7
N_2(node in S_1)
1 4 1 4 2 4 2 7 7 7 7
S_1
S_0


N_2(node in S_0)
1 4 1 1 2 3 1 2 3 7 7 7 7
N_2(node in S_1)
1 4 1 4 2 4 2 7 7 7 7
Graph is "symmetric", yet highly non-regular!
S_1
S_0

Lower Bound: The Argument

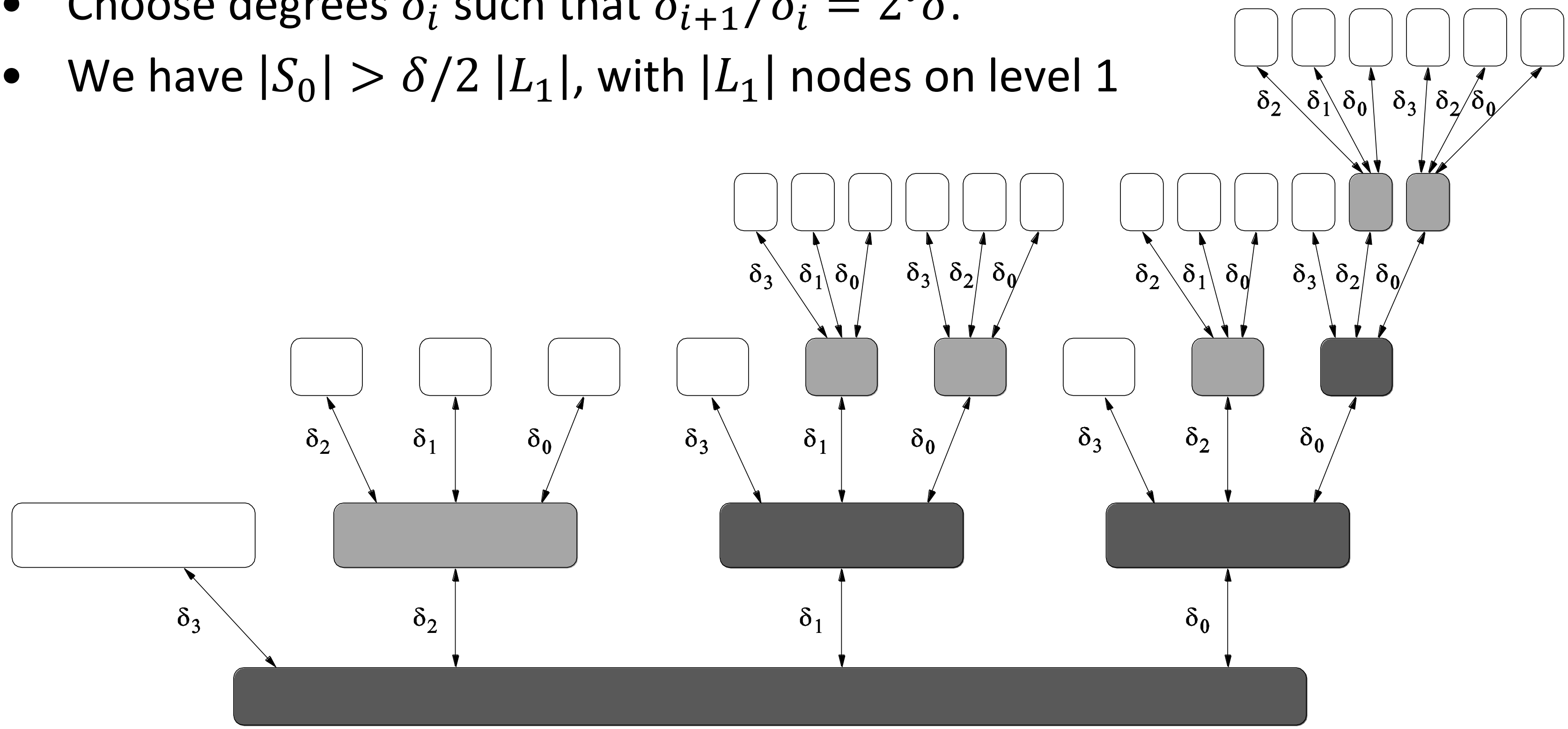
- The example graph is for t = 3.
- All edges are in fact special bipartite graphs with large enough girth.



- If you use the graph of recursion level t, then a distributed algorithm cannot find a good MVC approximation in time t.

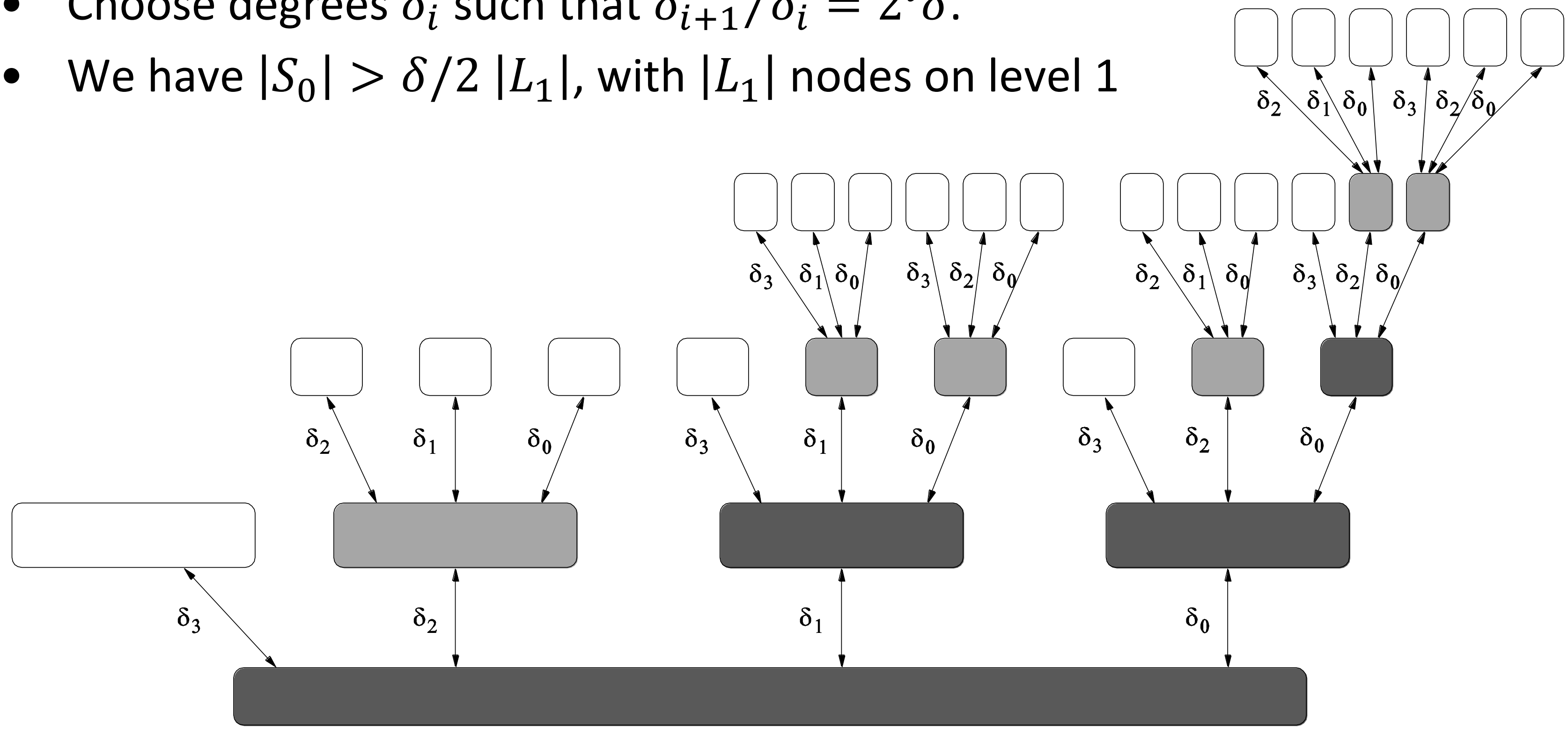
Lower Bound: The Math

- Choose degrees δ_i such that $\delta_{i+1}/\delta_i = 2^i\delta$.
- We have $|S_0| > \delta/2\;|L_1|$, with $|L_1|$ nodes on level 1



Lower Bound: The Math

- Choose degrees δ_i such that $\delta_{i+1}/\delta_i = 2^i\delta$.
- We have $|S_0| > \delta/2\ |L_1|$, with $|L_1|$ nodes on level 1



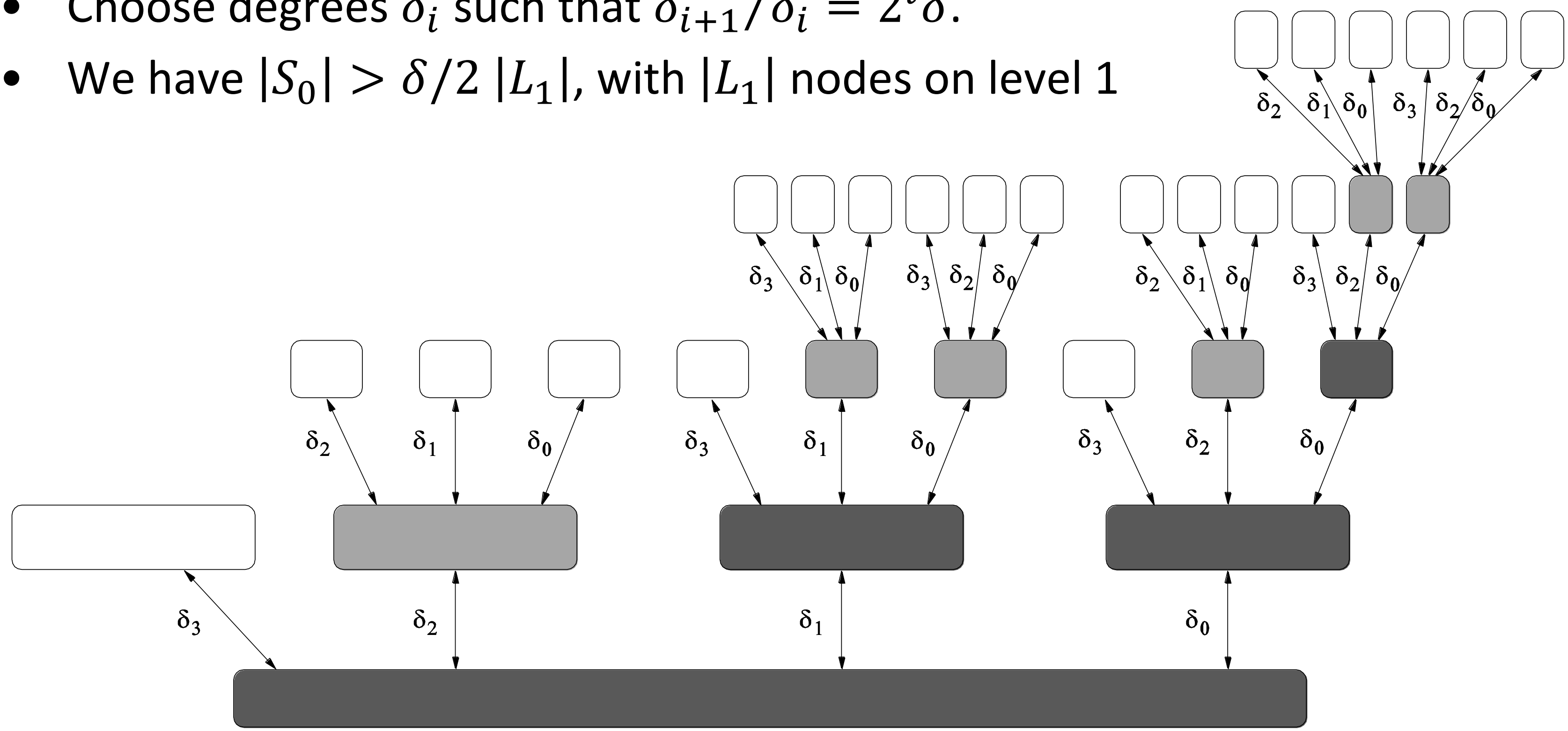
- By induction we have a $(1 - \Theta(1/\delta))$ fraction of the nodes is in S_0.
- Now δ, n, Δ are depending on the recursion level t.

Lower Bound: The Math

Graph useful for proving lower bounds in sublinear algos?

Open

- Choose degrees δ_i such that $\delta_{i+1}/\delta_i = 2^i\delta$.
- We have $|S_0| > \delta/2\ |L_1|$, with $|L_1|$ nodes on level 1



- By induction we have a $(1 - \Theta(1/\delta))$ fraction of the nodes is in S_0.
- Now δ, n, Δ are depending on the recursion level t.

Lower Bound: Results

- We can show that for $\epsilon > 0$, in t time, the approximation ratio is at least

$$\Omega\left(n^{\frac{1/4-\varepsilon}{t^2}}\right) \quad and \quad \Omega\left(\Delta^{\frac{1-\varepsilon}{t+1}}\right)$$

- Constant approximation needs at least $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ time.
- Polylog approximation $\Omega(\log \Delta / \log \log \Delta)$ and $\Omega(\sqrt{\log n / \log \log n})$.

Lower Bound: Results

- We can show that for $\epsilon > 0$, in t time, the approximation ratio is at least

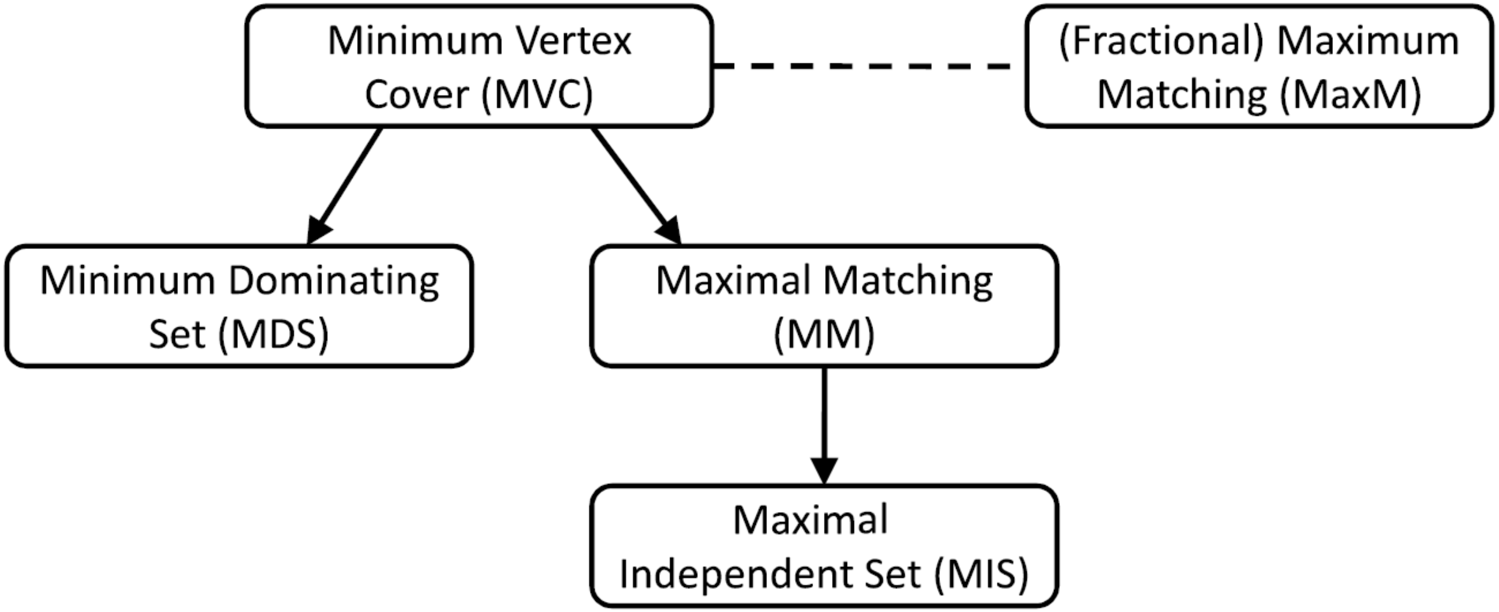
tight for MVC

$$\Omega\left(n^{\frac{1/4-\varepsilon}{t^2}}\right) \quad and \quad \Omega\left(\Delta^{\frac{1-\varepsilon}{t+1}}\right)$$

- Constant approximation needs at least $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ time.
- Polylog approximation $\Omega(\log \Delta / \log \log \Delta)$ and $\Omega(\sqrt{\log n / \log \log n})$.

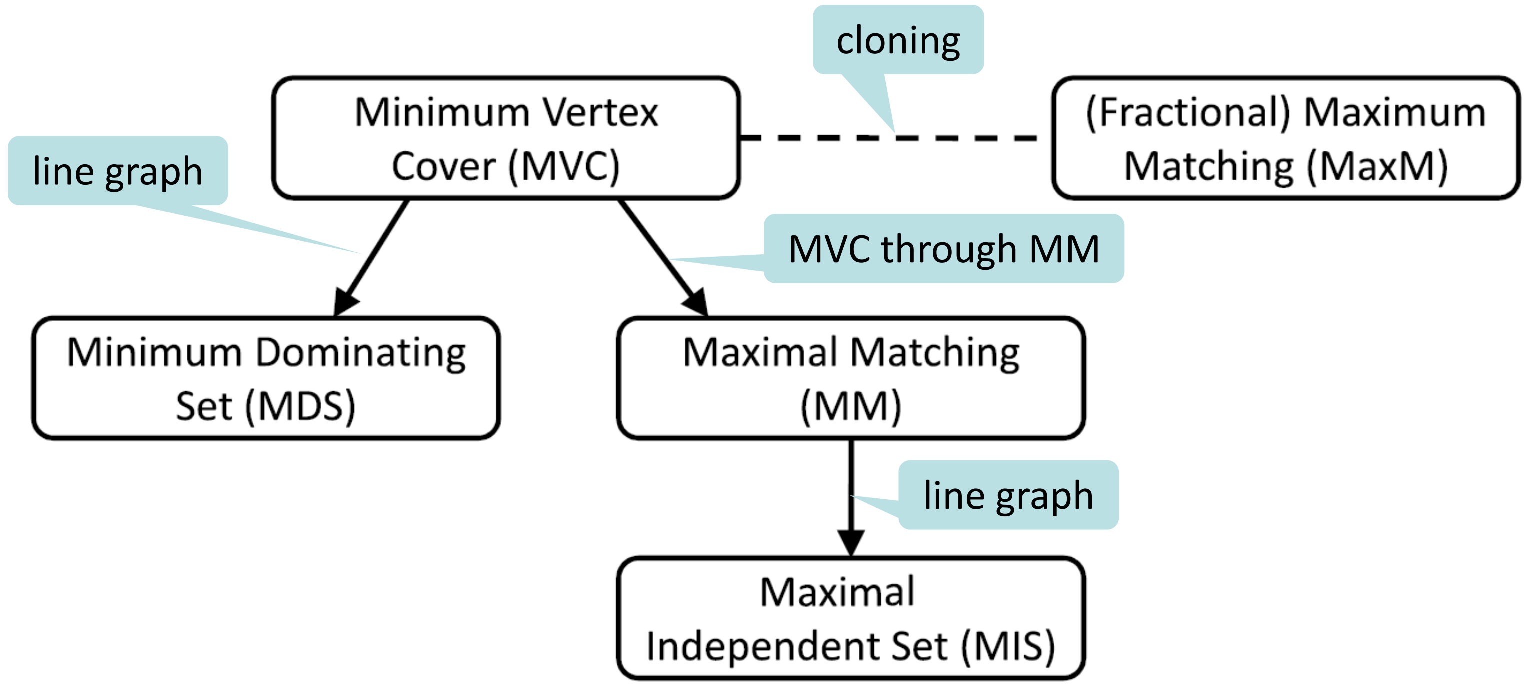
Lower Bound: Reductions

- Many "local looking" problems need non-trivial t, in other words, the bounds $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ hold for a variety of classic problems.

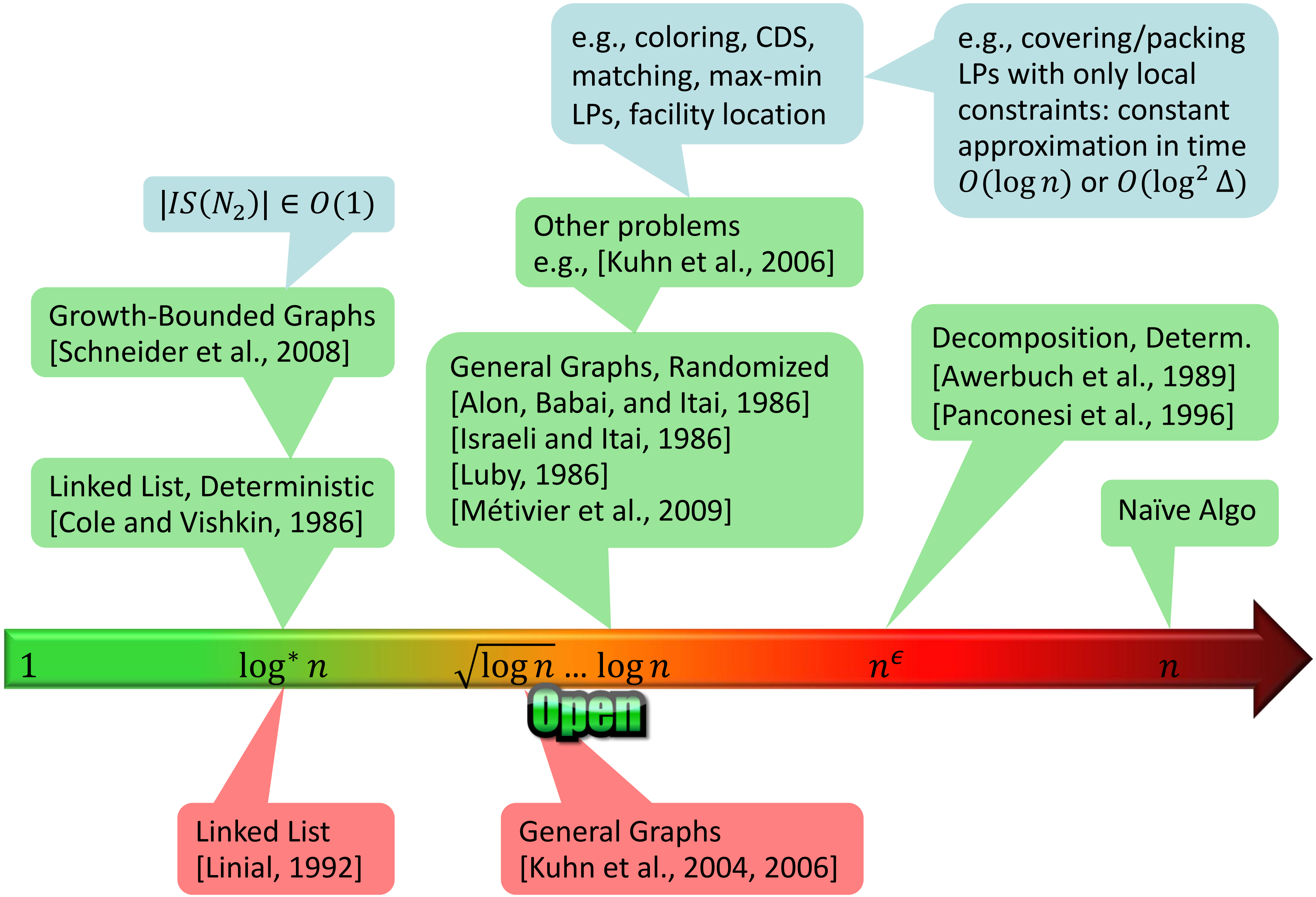


Lower Bound: Reductions

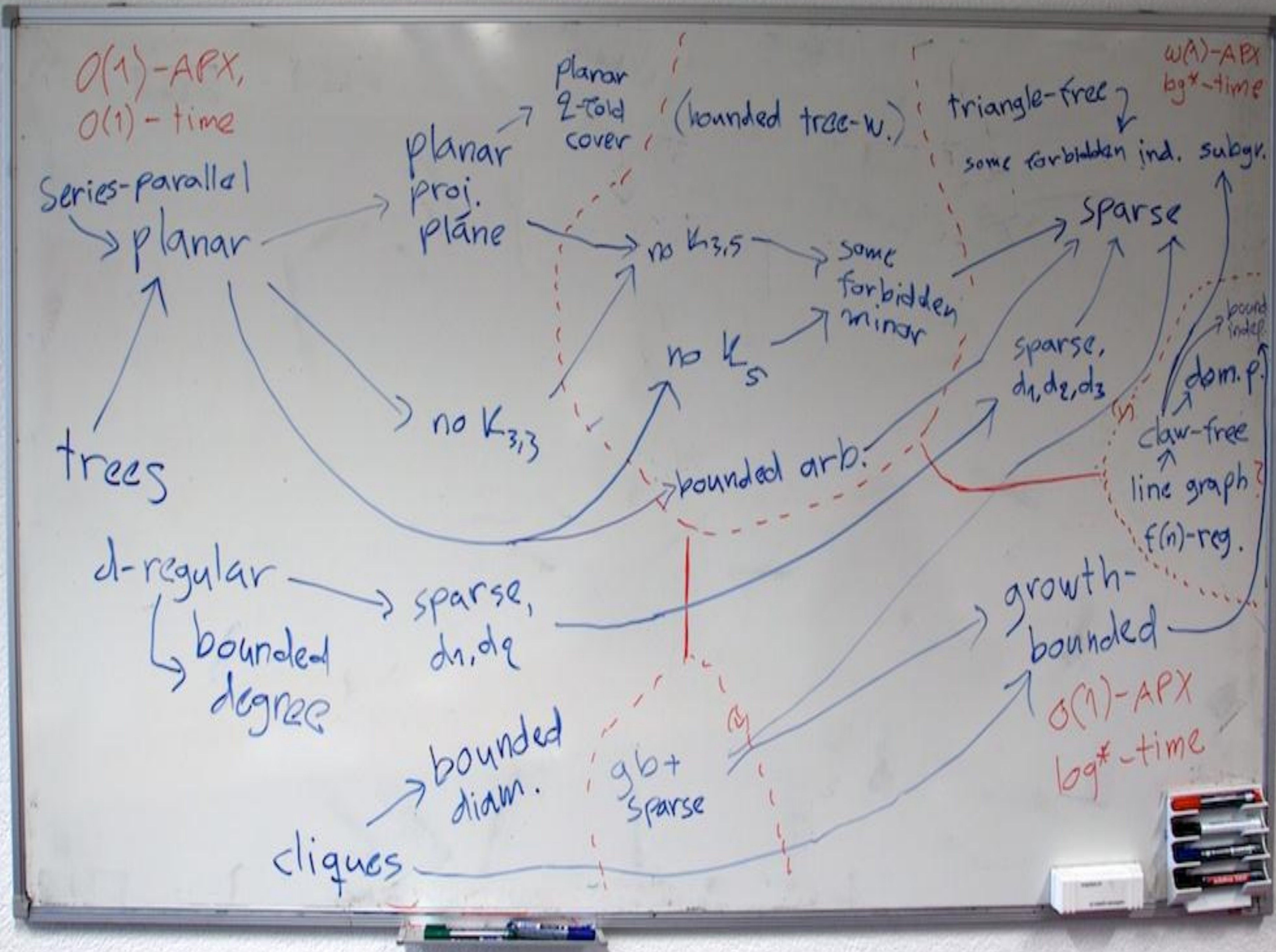
- Many "local looking" problems need non-trivial *t*, in other words, the bounds $\Omega(\log \Delta)$ and $\Omega(\sqrt{\log n})$ hold for a variety of classic problems.



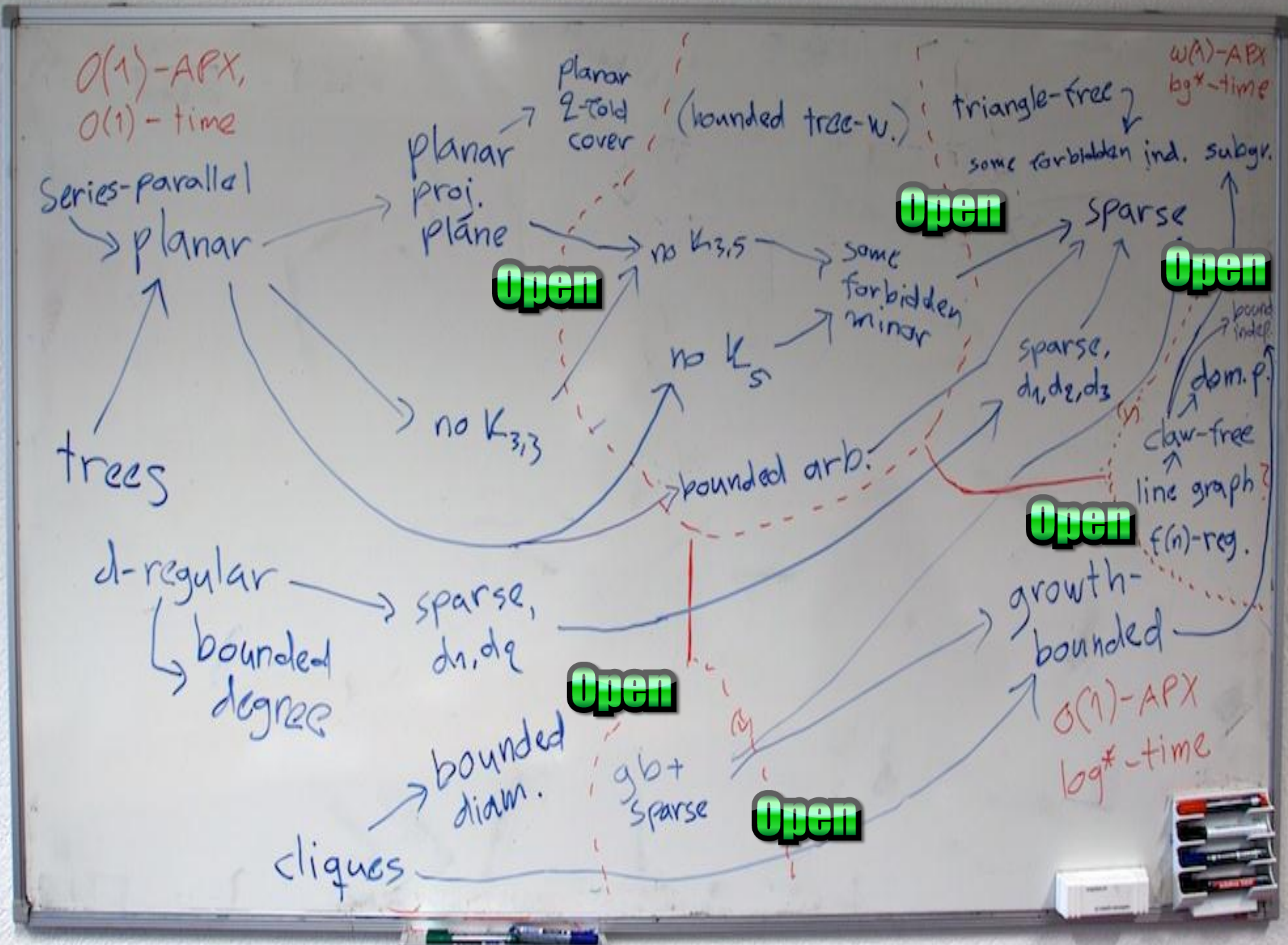
Results: MIS



e.g., coloring, CDS, matching, max-min LPs, facility location
e.g., covering/packing LPs with only local constraints: constant approximation in time $O(\log n)$ or $O(\log^2 \Delta)$
$|IS(N_2)| \in O(1)$
Other problems e.g., [Kuhn et al., 2006]
Growth-Bounded Graphs [Schneider et al., 2008]
General Graphs, Randomized [Alon, Babai, and Itai, 1986] [Israeli and Itai, 1986] [Luby, 1986] [Métivier et al., 2009]
Decomposition, Determ. [Awerbuch et al., 1989] [Panconesi et al., 1996]
Linked List, Deterministic [Cole and Vishkin, 1986]
Naïve Algo
1
$\log^* n$
$\sqrt{\log n}$ ... $\log n$
n^ϵ
n
Open
Linked List [Linial, 1992]
General Graphs [Kuhn et al., 2004, 2006]

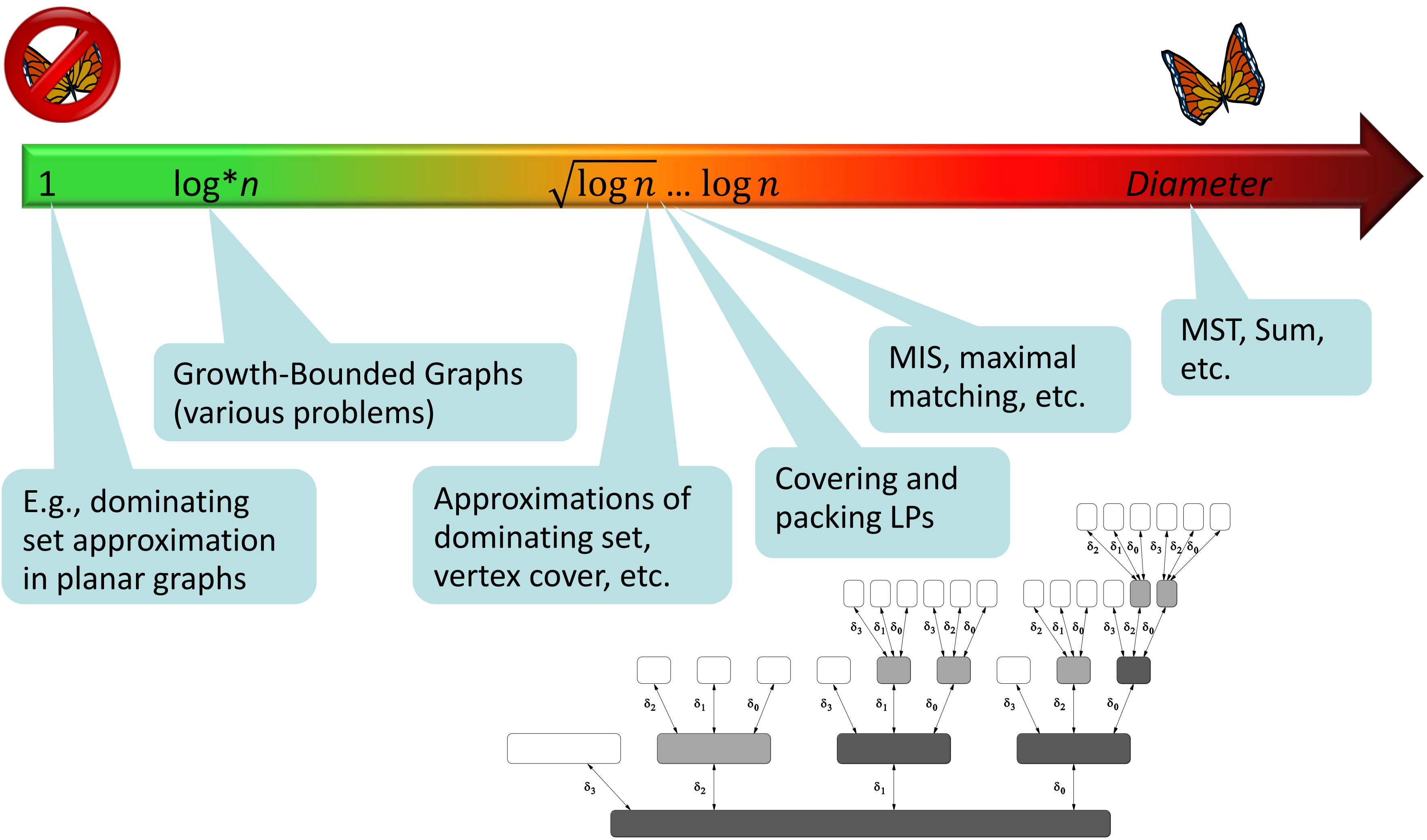


O(1)-APX,
O(1)-time
planar 2-fold cover
(bounded tree-w.)
triangle-free
some forbidden ind. subgr.
ω(1)-APX
log*-time
series-parallel
planar
planar proj. plane
sparse
no K3,5
some forbidden minor
no K5
no K3,3
sparse, d1,d2,d3
bound indep.
dom. p.
claw-free
line graph?
f(n)-reg.
trees
bounded arb.
d-regular
sparse, d1,d2
bounded degree
growth-bounded
O(1)-APX
log*-time
bounded diam.
gb+sparse
cliques



O(1)-APX,
O(1)-time
planar 2-fold cover
(bounded tree-w.)
triangle-free
ω(1)-APX
log*-time
some forbidden ind. subgr.
Open
planar
proj.
plane
Series-parallel
planar
sparse
Open
no K3,5
some forbidden minor
Open
no K5
sparse, d1,d2,d3
bound indep.
dom. p.
trees
no K3,3
claw-free
bounded arb.
line graph?
Open
f(n)-reg.
d-regular
sparse, d1,d2
growth-bounded
bounded degree
Open
O(1)-APX
log*-time
bounded diam.
gb+ sparse
Open
cliques

Summary so far…

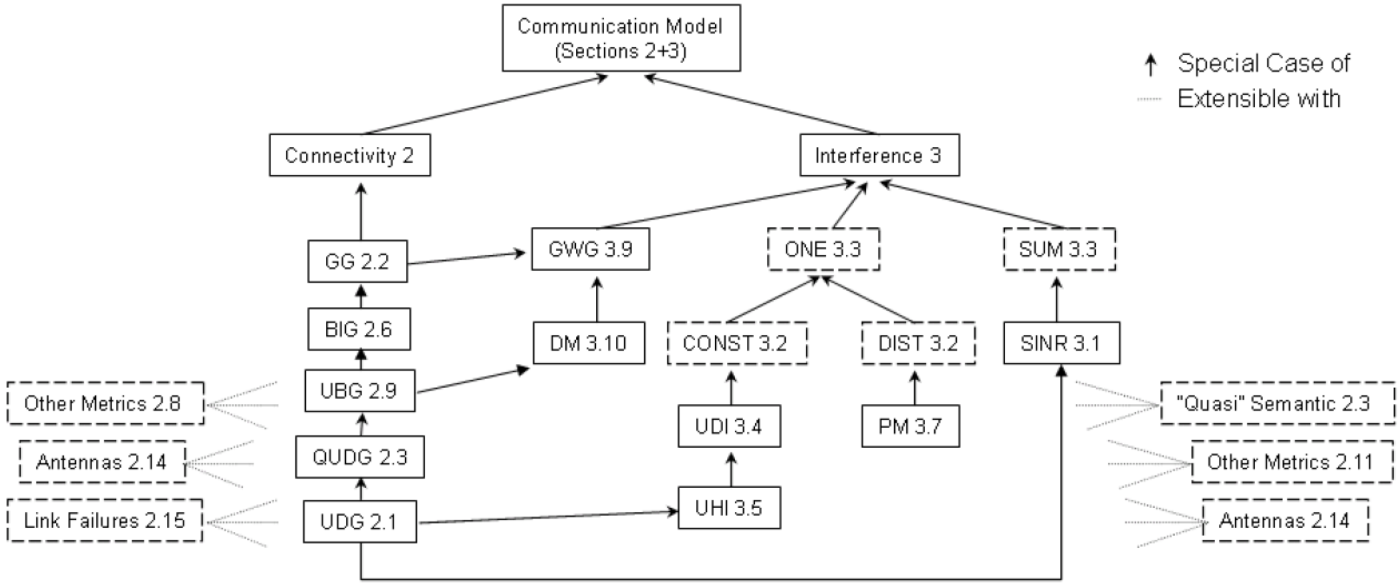


Ad Hoc & Sensor Networks?

Ad hoc Network Connectivity Models

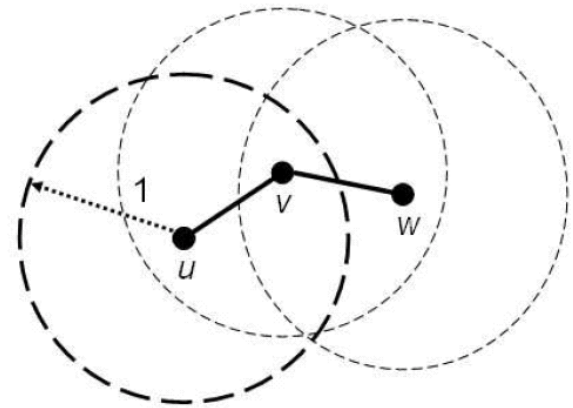
- Formal models help us understanding a problem

- Formal proofs of correctness and efficiency
- Common basis to compare results
- Unfortunately, for ad hoc and sensor networks, a myriad of models exist, most of them make sense in some way or another. On the next few slides we look at a few selected models



Unit Disk Graph (UDG)

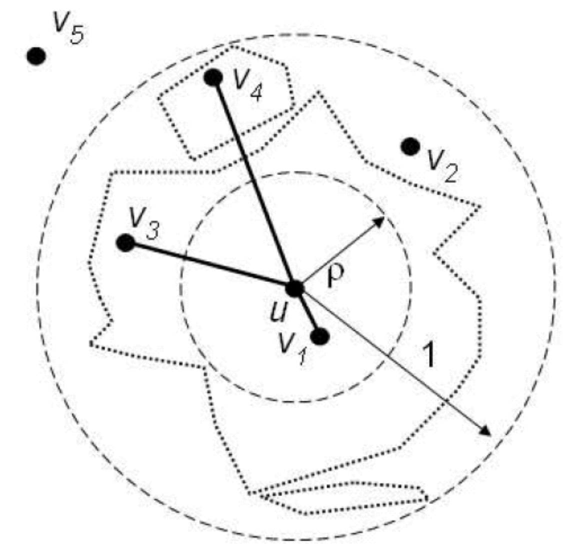
- Classic computational geometry model, special case of disk graphs
- All nodes are points in the plane, two nodes are connected iff (if and only if) their distance is at most 1, that is $\{u,v\} \in E \Leftrightarrow |u,v|$ 1

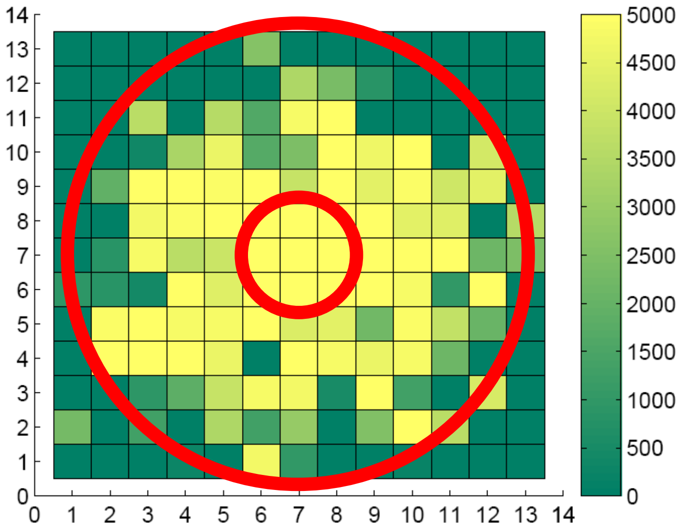


+ Very simple, allows for strong analysis
– Not realistic: “If you gave me $100 for each paper written with the unit disk assumption, I still could not buy a radio that is unit disk!”
– Particularly bad in obstructed environments (walls, hills, etc.)
- Natural extension: 3D UDG

Quasi Unit Disk Graph (QUDG)

- Two radii, 1 and , with 1
  - |u,v| ⇔ {u,v} ∈ E
  - 1 < |u,v| ⇔ {u,v} ∉ E
  - < |u,v| 1 ⇔ it depends!
    - … on an adversary
    - … on probabilistic model
    - …





\+ Simple, analyzable

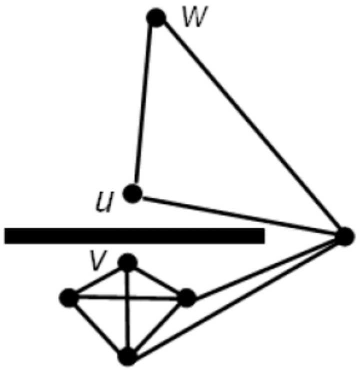
\+ More realistic than UDG

– Still bad in obstructed environments (walls, hills, etc.)

- Natural extension: 3D QUDG

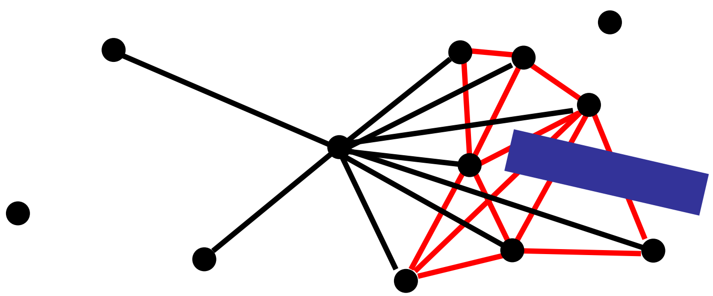
Bounded Independence Graph (BIG)

- How realistic is a QUDG?
  - u and v can be close but not adjacent
  - model requires very small in obstructed environments (walls)



- However: in practice, neighbors are often also neighboring

- Solution: BIG Model
  - Bounded independence graph
  - Size of any independent set grows polynomially with hop distance *r*
  - e.g., $f(r) = O(r^2)$ or $O(r^3)$
  - A set S of nodes is an independent set, if there is no edge between any two nodes in S.
  - BIG model also known as bounded-growth
    - Unfortunately, the term bounded-growth is ambiguous



Unit Ball Graph (UBG)

- $\exists$ metric (V,d) with constant doubling dimension.

- Metric: Each edge has a distance d, with

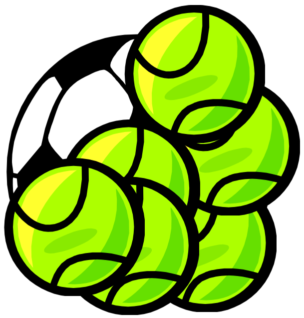
  1. $d(u,v) \geq 0$ (non-negativity)
  2. $d(u,v) = 0$ iff $u = v$ (identity of indiscernibles)
  3. $d(u,v) = d(v,u)$ (symmetry)
  4. $d(u,w)\ \ d(u,v) + d(v,w)$ (triangle inequality)

- Doubling dimension: log(#balls of radius r/2 to cover ball of radius r)
  - Constant: you only need a constant number of balls of half the radius

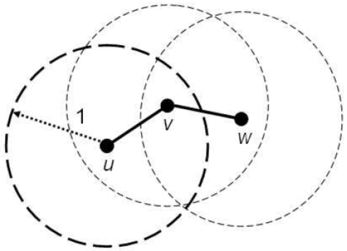
- Connectivity graph is same as UDG:

  such that: $d(u,v)\ \ 1 : (u,v) \in E$

  $d(u,v) > 1\ : (u,v) \in E/$



Connectivity Models: Overview



General Graph

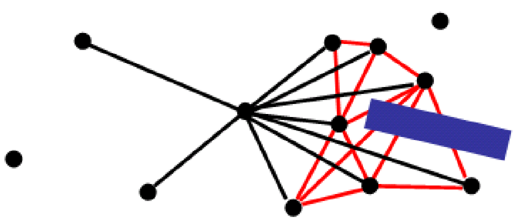
UDG

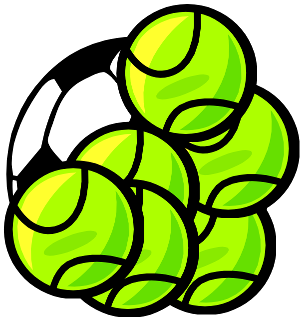
too pessimistic **too optimistic**

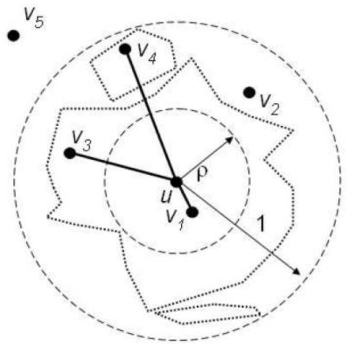
Bounded Independence

Unit Ball Graph

Quasi UDG

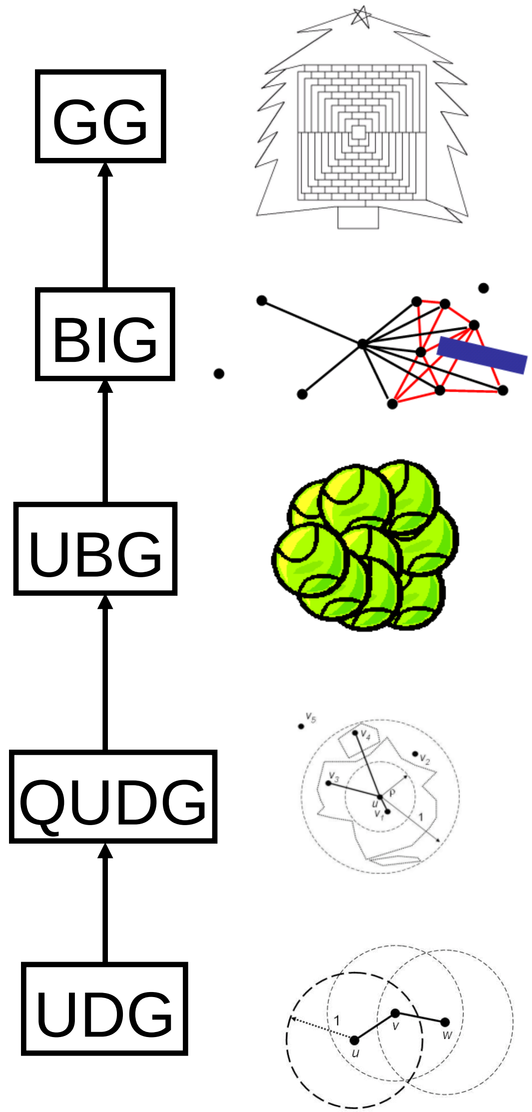






Models are related

- BIG is special case of general graph, BIG $\subseteq$ GG
- UBG $\subseteq$ BIG because the size of the independent sets of any UBG is polynomially bounded
- QUDG(constant) $\subseteq$ UBG
- QUDG(=1) = UDG

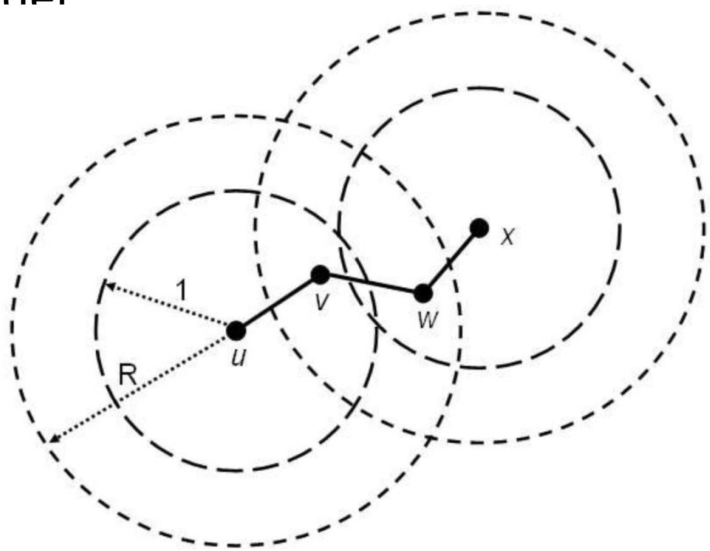


Wireless Interference Models: Protocol Model

- For lower layer protocols, a model needs to be specific about interference. A simplest interference model is an extension of the UDG. In the protocol model, a transmission by a node in at most distance 1 is received iff there is no conflicting transmission by a node in distance at most R, with R $\geq$ 1, sometimes just R = 2.

+ Easy to explain
– Inherits all major drawbacks from the UDG model
– Does not easily allow for designing distributed algorithms/protocols
– Lots of interfering transmissions just outside the interference radius R do not sum up

- Can be extended with the same extensions as UDG, e.g. QUDG



Hop Interference (HI)

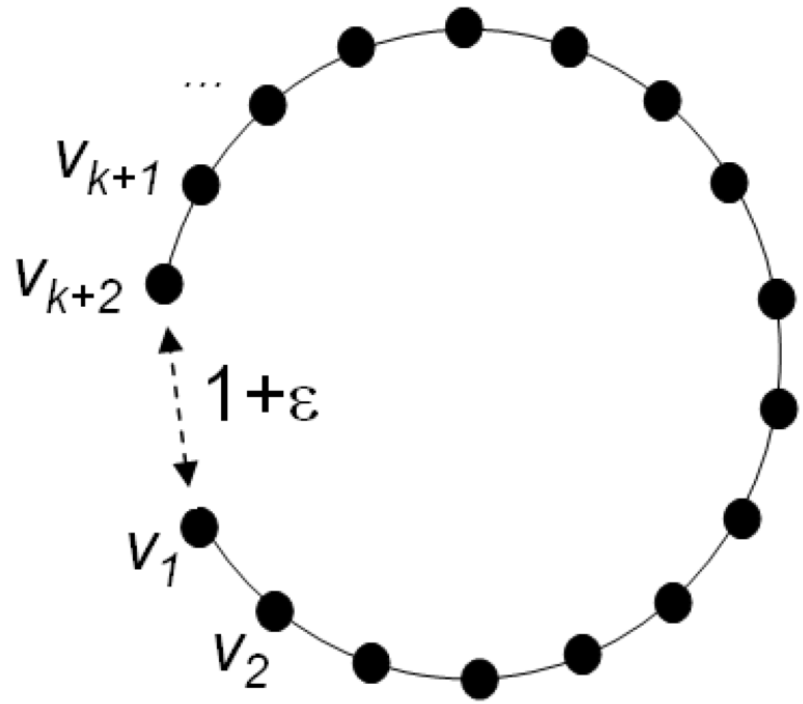
- An often-used interference model is hop-interference. Here a UDG is given. Two nodes can communicate directly iff they are adjacent, and if there is no concurrent sender in the *k*-hop neighborhood of the receiver (in the UDG). Sometimes $k = 2$.

- Special case of the protocol model, inheriting all its drawbacks

+ Simple
+ Allows for distributed algorithms

– A node can be close but not produce any interference (see picture)

- Can be extended with the same extensions as UDG, e.g. QUDG



Physical (SINR) Model

- We look at the signal-to-noise-plus-interference (SINR) ratio.
- Message arrives if SINR is larger than β at receiver

Power level of sender u

Path-loss exponent, = 2,...,6

$$\frac{\frac{P_u}{d(u,v)^{\alpha}}}{N + \sum_{w \in V \setminus \{u\}} \frac{P_w}{d(w,v)^{\alpha}}} \geq \beta$$

Noise

Distance between transmitter *w* and receiver *v*

Minimum signal-to-interference ratio, depending on quality of hardware, etc.

- Mind that the SINR model is far from perfect as well.

SINR Discussion

+ In contrast to other low-layer models such as PM the SINR model allows for interference that does sum up. This is certainly closer to reality. However, SINR is not reality. In reality, e.g., competing transmissions may even cancel themselves, and produce less interference. In that sense the SINR model is pessimistic (interference summing up) and optimistic (if we remove the "I" from the SINR model, we have a UDG, which we know is not correct) at the same time.

– SINR is "complicated", hard to analyze
– Similarly as PM, SINR does not really allow for distributed algorithms
– Also, in reality, e.g. the signal fluctuates over time. Some of these issues are captured by more complicated fading channel models.

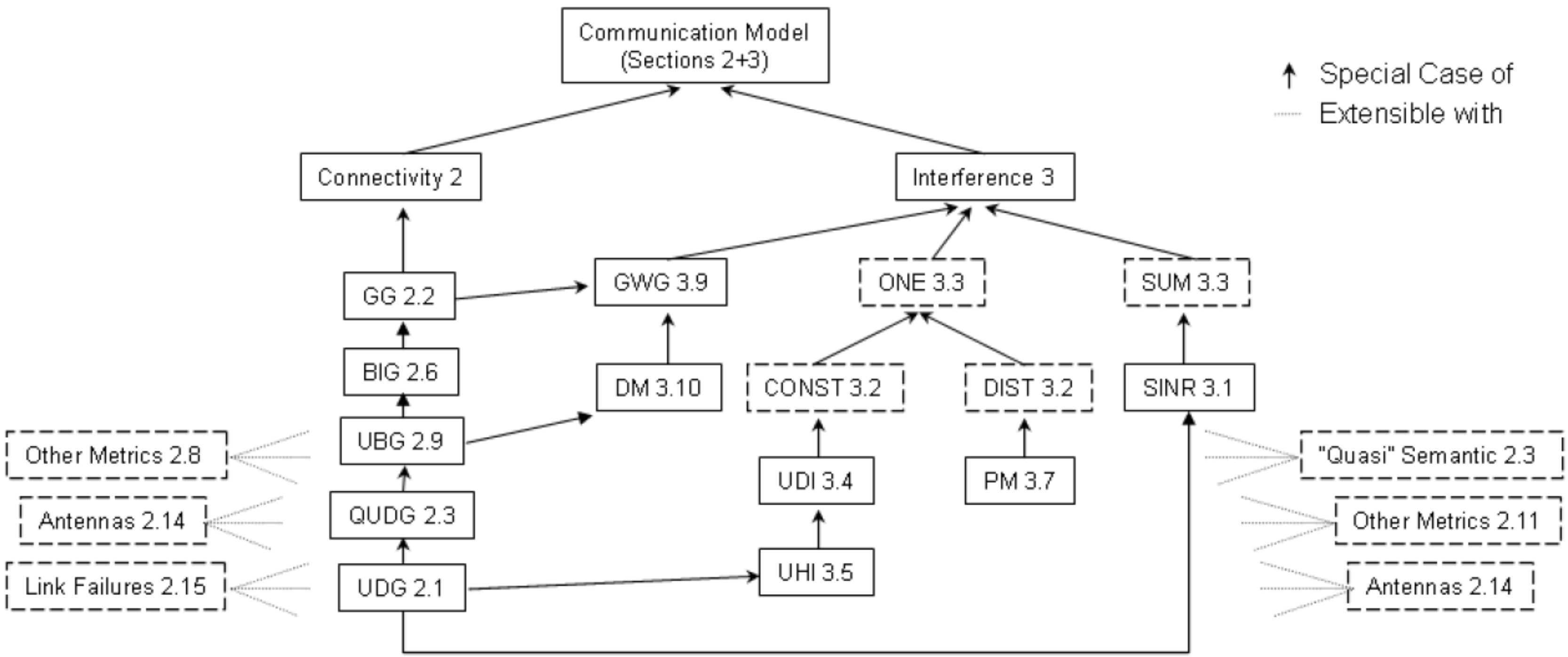
More on SINR

- Often there is more than a single threshold , that decides whether reception is possible or not. In many networks, a higher S/N ratio allows for more advanced modulation and coding techniques, allowing for higher throughput (e.g. Wireless LAN 802.11)

- However, even more is possible: For example, assume that a receiver is receiving two signals, signal S_1 being much stronger than signal S_2. Then S_2 has a terrible S/N ratio. However, we might be able to "subtract" the strong S_1 from the total signal, and with "$S - S_1 = S_2$" also get S_2.

- These are just two examples of how to get more than you expect.

Model Overview



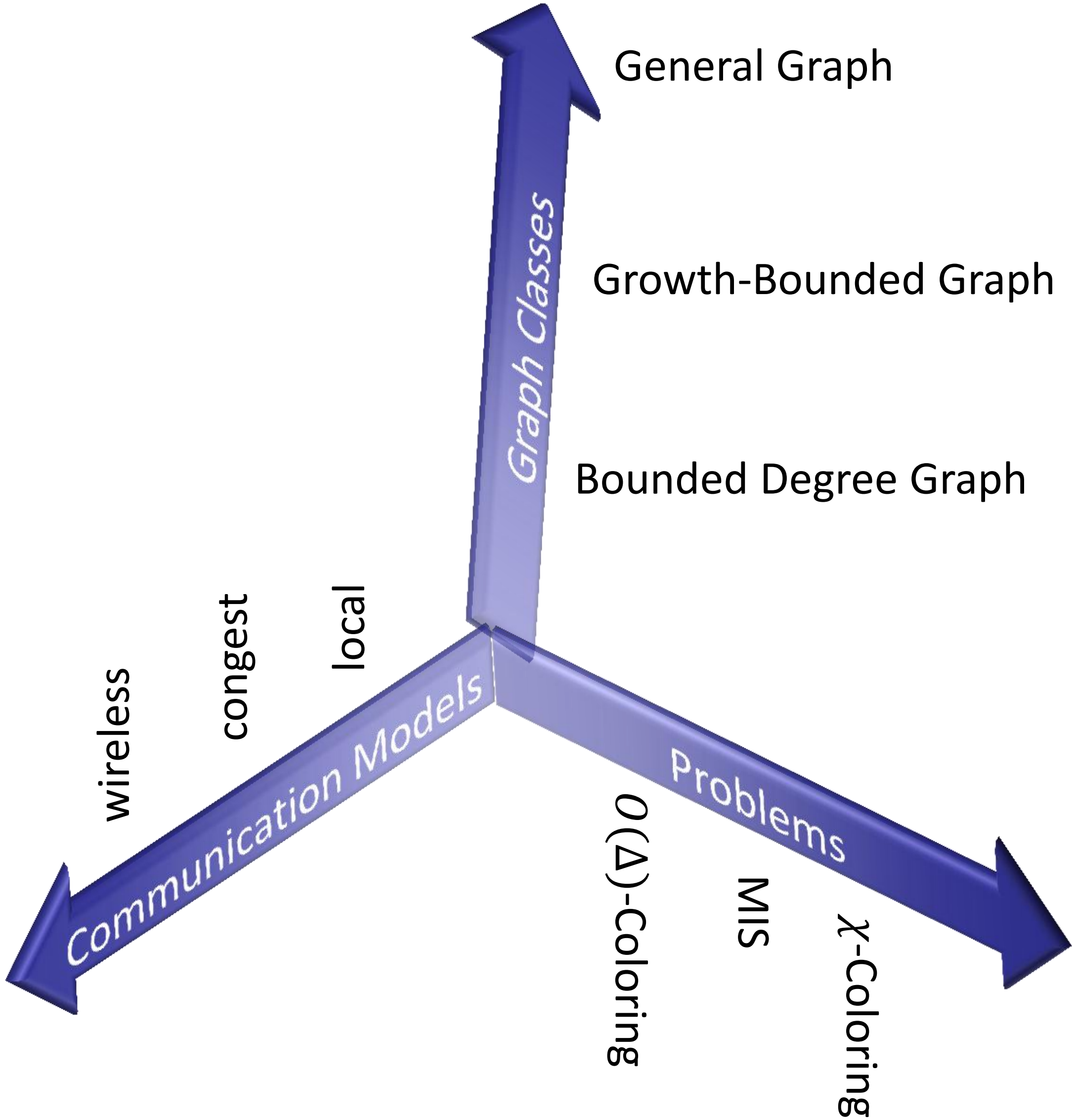
[Schmid et al., 2006]

- Try to proof correctness in an as "high" as possible model
- For efficiency, a more optimistic ("lower") model is fine
- Lower bounds should be proved in "low" models.

Wireless Media Access?

- Radio Network Model
  - Slotted time (unslotted time only costs factor 2)
  - In each slot, each node can either transmit, receive, or sleep
  - Nodes receive transmissions depending on connectivity & interference models
  - With or without collision detection
  - With or without synchronous start
  - With or without …
- Beeper Model
  - Nodes can just beep
  - If at least one neighbor beeps, a node will receive that (no interference)
  - Yes, this can be done in reality, e.g. slotted programming

Summary



General Graph
Graph Classes
Growth-Bounded Graph
Bounded Degree Graph
wireless
congest
local
Communication Models
Problems
$O(\Delta)$-Coloring
MIS
χ-Coloring

Thank You!

Questions & Comments?

Open Problems

- Close the gap between $\sqrt{\log n}$ and $\log n$ (for randomized algorithms)!
- Find a fast deterministic MIS algorithm (or strong det. lower bound)!
- Where are the boundaries between constant, log*, log, and diameter?
- What about algorithms that cannot even exchange messages?
- Can the lower bound graph be used in the context of sublinear algorithms?