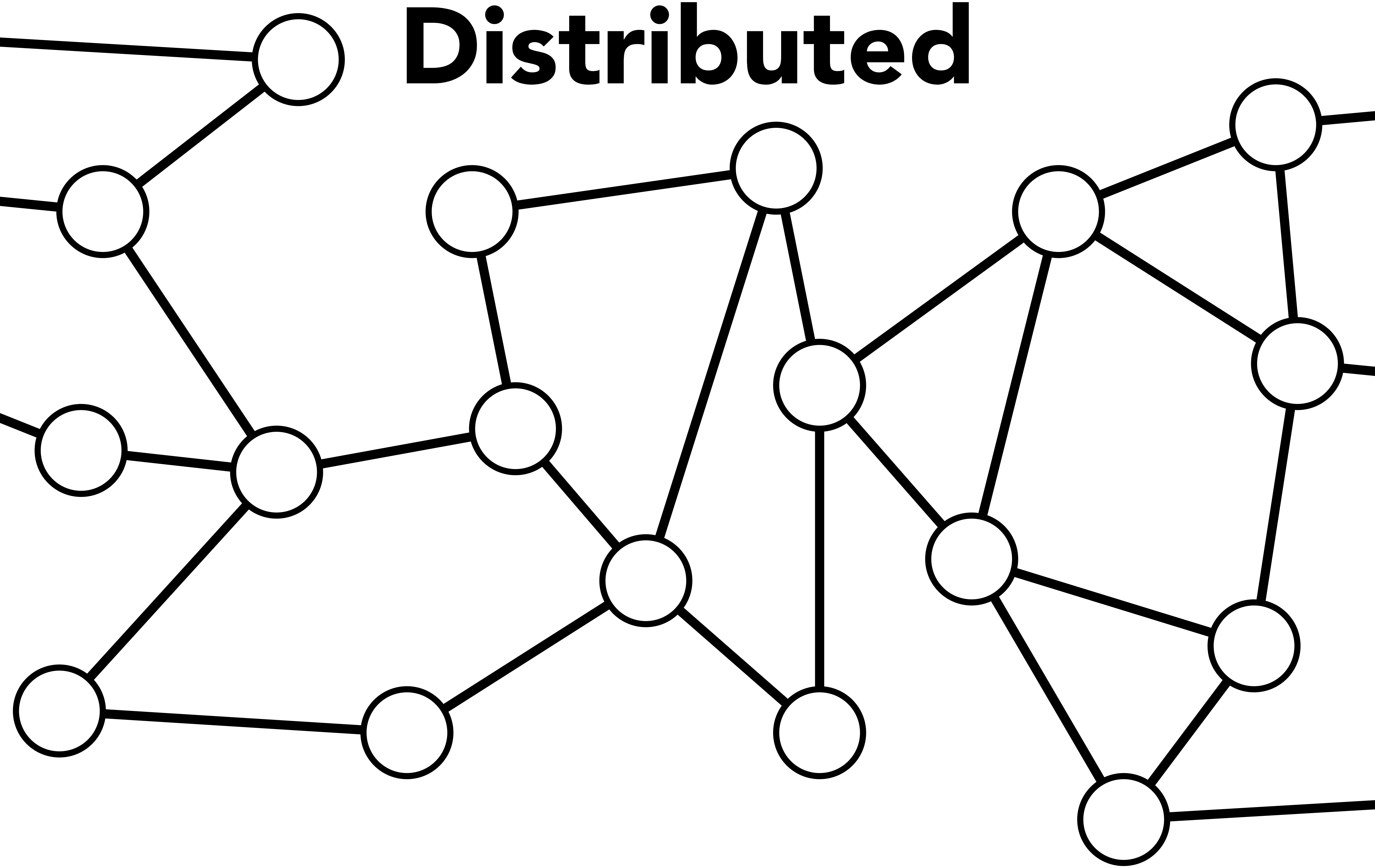
Fast distributed graph algorithms

Juho Hirvonen
Aalto University & HIIT

Distributed



Distributed

Algorithms

Fast

- the number of nodes ***n***
- the maximum degree **Δ**

Fast

- the number of nodes n
- the maximum degree Δ

Imagine sparse graphs: Δ = constant, $n \to \infty$

Fast

Define:

fast algorithms depend only mildly on $\boldsymbol{n}$

where mildly = $\mathbf{O(\log^* n)}$

Fast

Define:

fast algorithms depend only mildly on ***n***

where mildly = **O(log*n)**

Recall that $\log^* n$ = smallest k s.t. $\log^{(k)} n \le 1$

Fast

Define:

constant-time algorithms are independent of ***n***

Q1: What can and cannot be computed by constant-time algorithms?

A: Computing a 3-coloring on an *n*-cycle requires **$\Omega(\log^* n)$** time [Linial ’92]

Implies the same lower bound for maximal matching and maximal independent set

A: Computing a 3-coloring on an *n*-cycle *takes* **O(log****n***)** time [Cole & Vishkin ’86]

Implies the same *upper bound* for maximal matching and maximal independent set

A: Computing any constant-approximation of an independent set or a maximum matching on an n-cycle requires **$\Omega(\log^* n)$** time

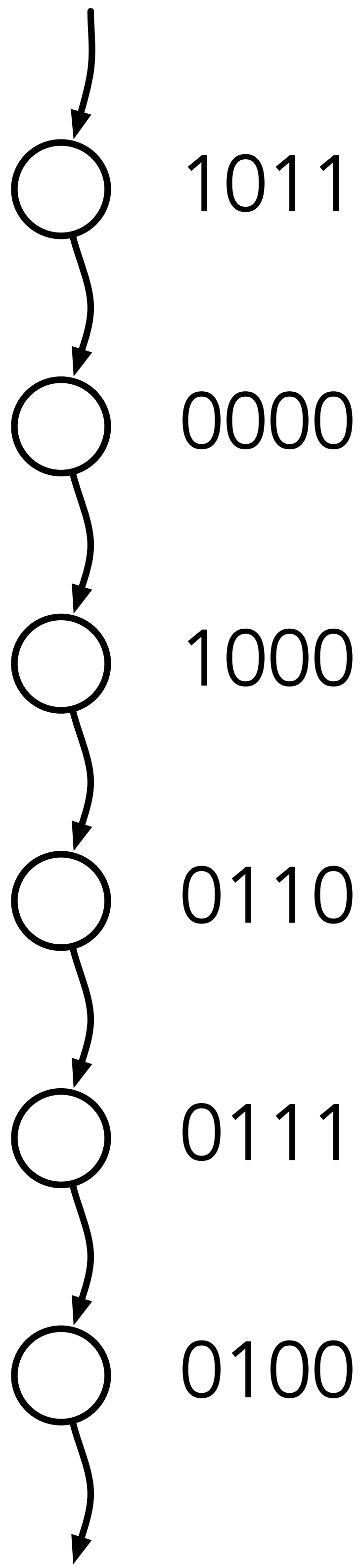
A: Computing any constant-approximation of an independent set or a maximum matching on an *n*-cycle takes **$O(\log^* n)$** time

There is a fundamental barrier at $\boldsymbol{\Omega(\log^* n)}$ time

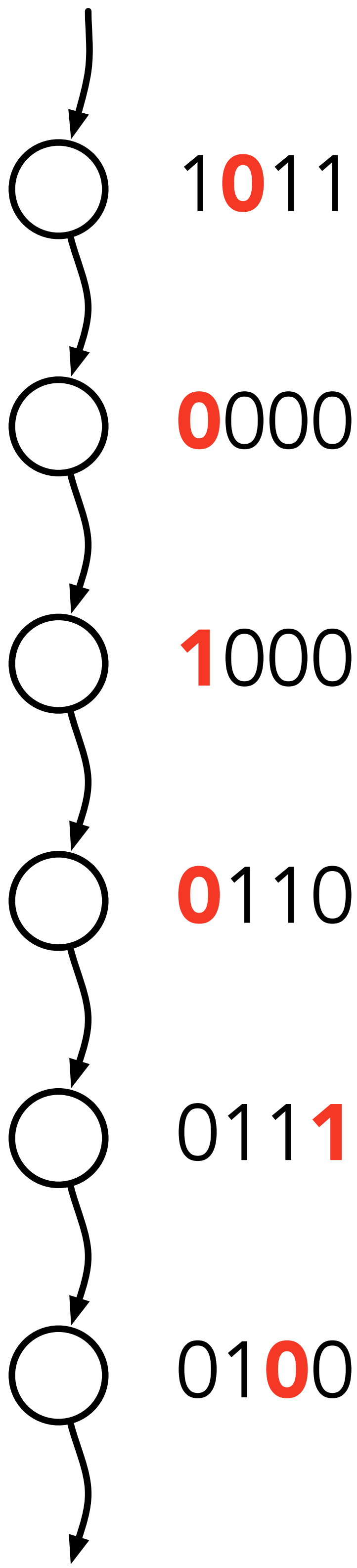
We will call this *symmetry breaking*

This is roughly the same as saying that adjacent nodes are not allowed *to produce the same output*

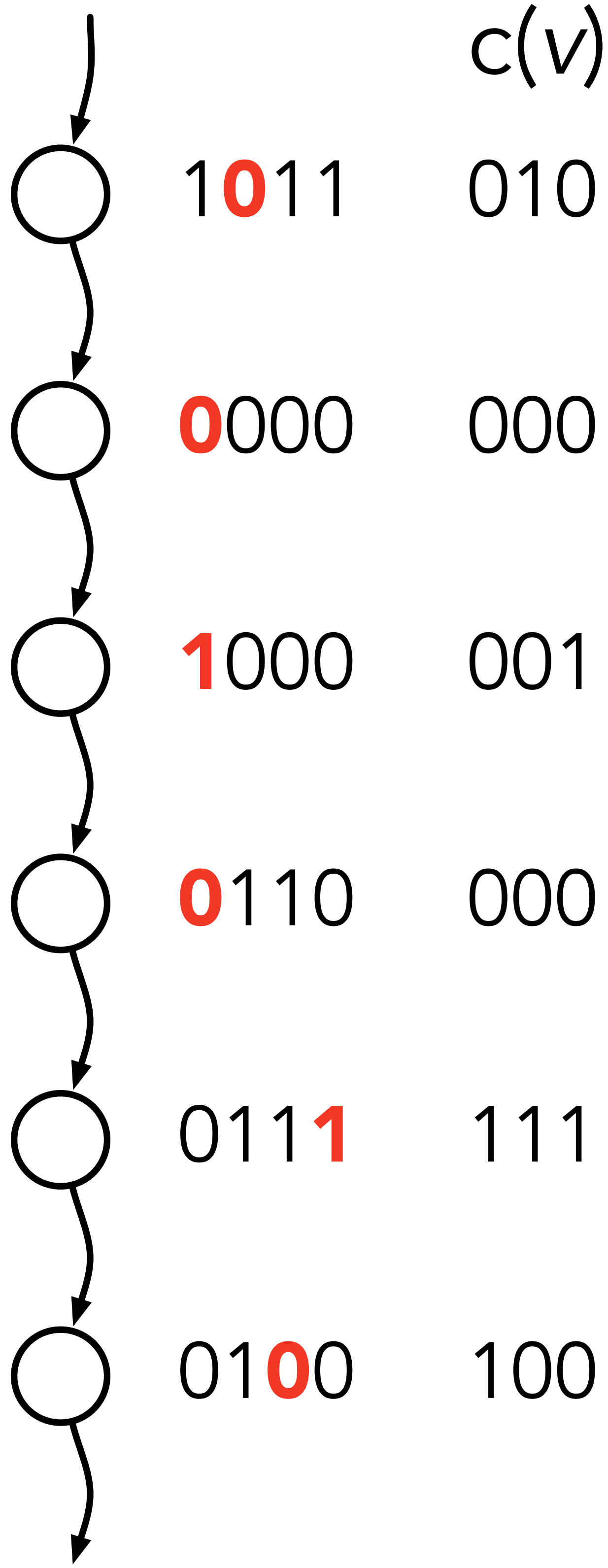
What does symmetry breaking look like?



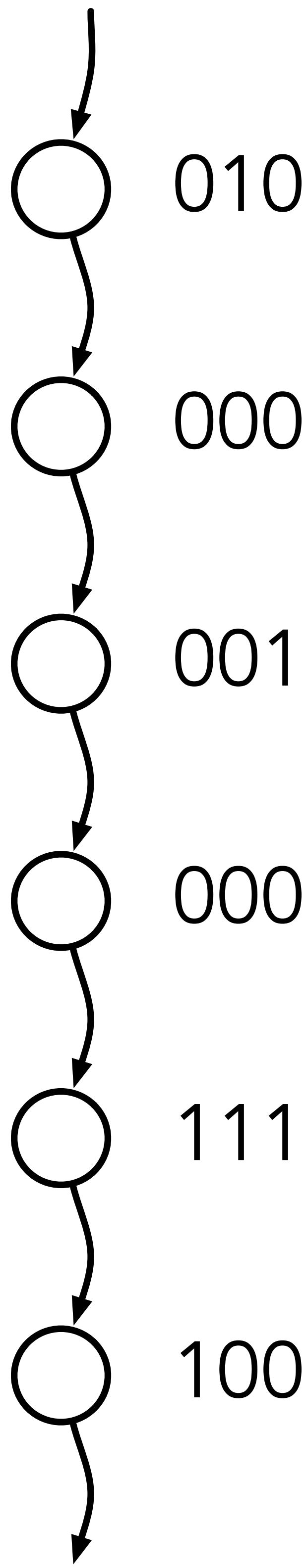
Start with *some* coloring



Learn parents color and the first bit (from the start) that differs



Compute c(*v*), which is the concatenation of the index of the differing bit and the bit itself



Repeat

Each round colors are reduced

$n \longrightarrow \log n + 1$

$n \longrightarrow$ 6 colors in $O(\log^* n)$ iterations

Q1: So what can be computed by constant-time algorithms?

A: 2-approximation of vertex cover [Åstrand et al. ’09]

A: Constant approximation of covering and packing LPs [Kuhn ’05]

A: Maximal fractional matchings [Åstrand and Suomela ’10]

Common theme: no need to break symmetry!

- Tähän kuvia ratkaisuista: syklissä triviaaleja…

The complexity landscape

O(1)

FMM
Packing & Covering
2APX-VC
…

Θ(log*n)

3-COL
MIS
MM
…

n

Another common theme:
constant-time algorithms do not use unique identifiers!

Conversely:
Known $O(\log^* n)$-time algorithms depend on the use of identifiers

Q2: How do unique identifiers help?

A: For constant-time algorithms solving *LCL-problems*

unique identifiers ≈ total order

[Naor & Stockmeyer '95]

- LCL problems

A: For constant-time algorithms solving *LCL-problems*

unique identifiers ≈ total order

[Naor & Stockmeyer '95]

A: For constant-time approximation of *PO-checkable problems*

unique identifiers

≈

port numbering and orientation

[Göös et al. ’13]

- PO-checkable problems

A: For constant-time approximation of *PO-checkable problems*

unique identifiers

≈

port numbering and orientation

[Göös et al. ’13]

No symmetry breaking

Symmetry breaking

Anonymous

O(1)

FMM
Packing & Covering
2APX-VC
…

ID

$\Theta(\log^* n)$

3-COL
MIS
MM
…

n

No symmetry breaking*

Anonymous
O(1)

FMM
Packing & Covering
2APX-VC
…

ID O(1)
Scheduling

Symmetry breaking

ID
Θ(log*n)

3-COL
MIS
MM
…

n

The picture is *fairly well understood* as a function of ***n***

Q3: What happens when $\Delta \geq 2$?

A: The running times of most fast algorithms depend on **Δ**!

- (Δ+1)-coloring, maximal independent set and maximal matching in time **O(Δ + log*n)**
- 2-approximation of vertex cover, fractional maximal matching in time **O(Δ)**
- Constant-approximation of various packing and covering problems in time **O(log Δ)**

In contrast usually the best known lower bound is $\Omega(\log \Delta)$!

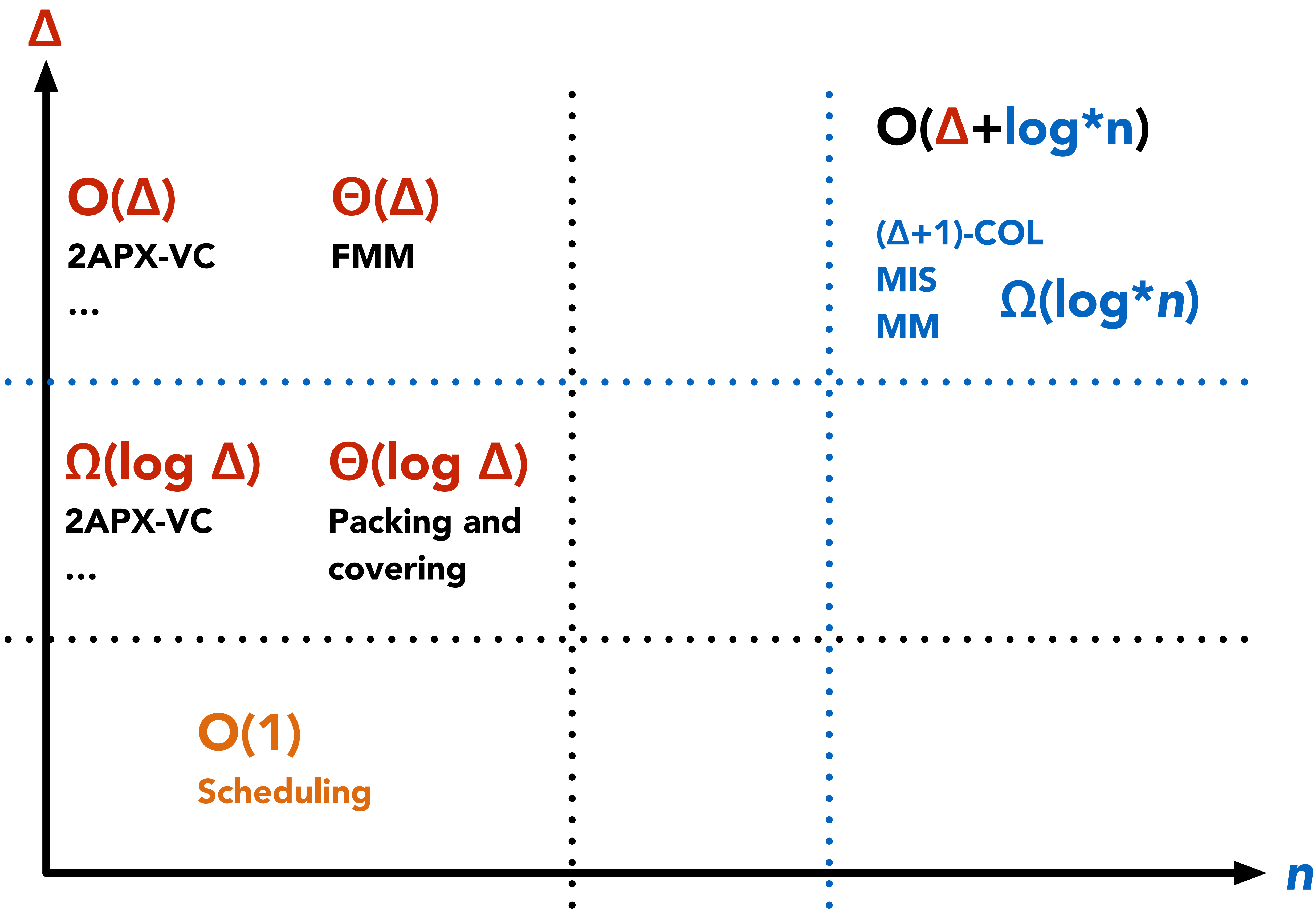
[Kuhn et al. ’05]

This bound is for constant approximation of packing and covering problems and it is tight

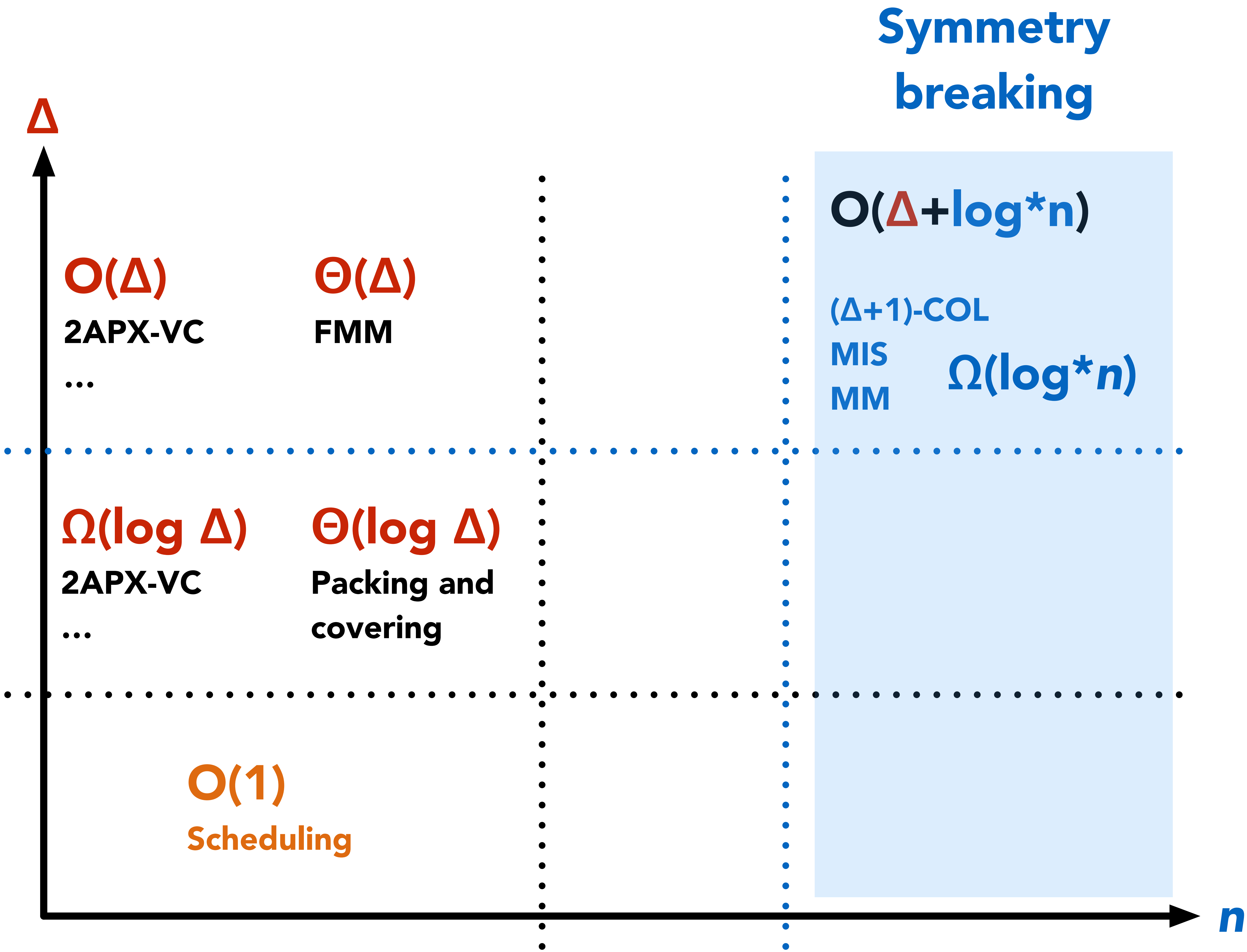
Recently we showed that fractional maximal matching requires time **$\Omega(\Delta)$** independent of ***n***

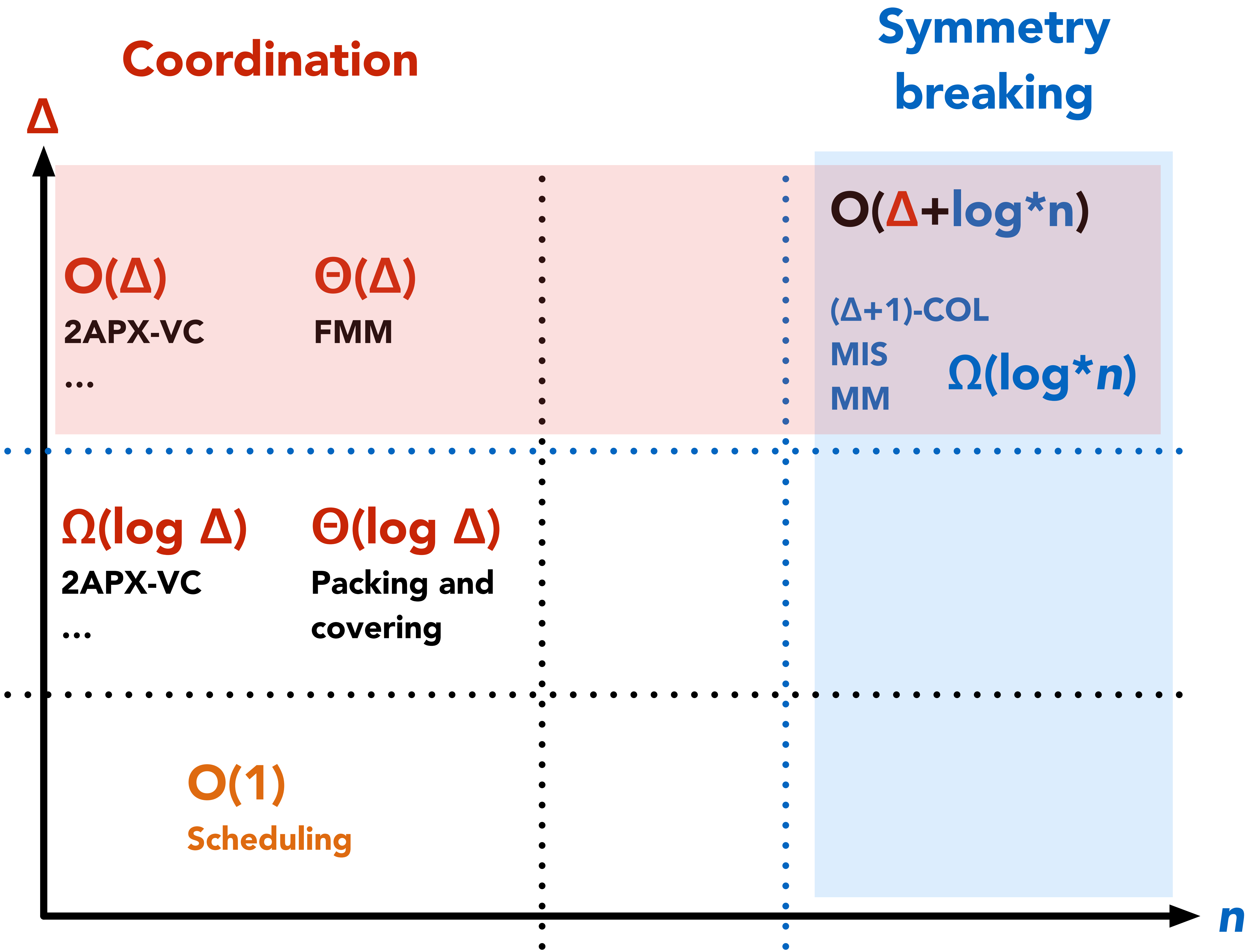
[Göös et al. '14]

Is this the case for all the other problems as well?



Δ
O(Δ+log*n)
O(Δ)
2APX-VC
...
Θ(Δ)
FMM
(Δ+1)-COL
MIS
MM
Ω(log*n)
Ω(log Δ)
2APX-VC
...
Θ(log Δ)
Packing and covering
O(1)
Scheduling
n


Symmetry breaking
Δ
O(Δ+log*n)
(Δ+1)-COL
MIS
MM
Ω(log*n)
O(Δ)
2APX-VC
...
Θ(Δ)
FMM
Ω(log Δ)
2APX-VC
...
Θ(log Δ)
Packing and covering
O(1)
Scheduling
n


Coordination
Symmetry breaking
Δ
O(Δ)
2APX-VC
...
Θ(Δ)
FMM
O(Δ+log*n)
(Δ+1)-COL
MIS
MM
Ω(log*n)
Ω(log Δ)
2APX-VC
...
Θ(log Δ)
Packing and covering
O(1)
Scheduling
n

Q4: Is there a separate symmetry breaking requirement and a coordination requirement?

Algorithms certainly seem to work this way!

Maximal matching [Panconesi & Rizzi ’95]:

1. Decompose graph into Δ forests in **O(1)** time
2. 3-Color each forest in time **O(log*n)**
3. Sequentially for each forest compute a maximal matching in time **O(Δ)**

Algorithms certainly seem to work this way!

(Δ+1)-coloring [Barenboim & Elkin '09]:

1. Compute an O(Δ^2)-coloring in time **O(log*n)** [Linial '92]
2. Improve *defective colorings* iteratively to get a (Δ+1)-coloring in time **O($\Delta \cdot \log\log \Delta$)****

On the other hand, this is certainly not true in general:

There is an $O(\log^4 n)$ time algorithm for maximal matching

[Hanckowiak et al. '01]

An interesting open question:

What is the complexity of maximal matching in 2-colored graphs?

Known to be **$O(\Delta)$** independent of ***n***

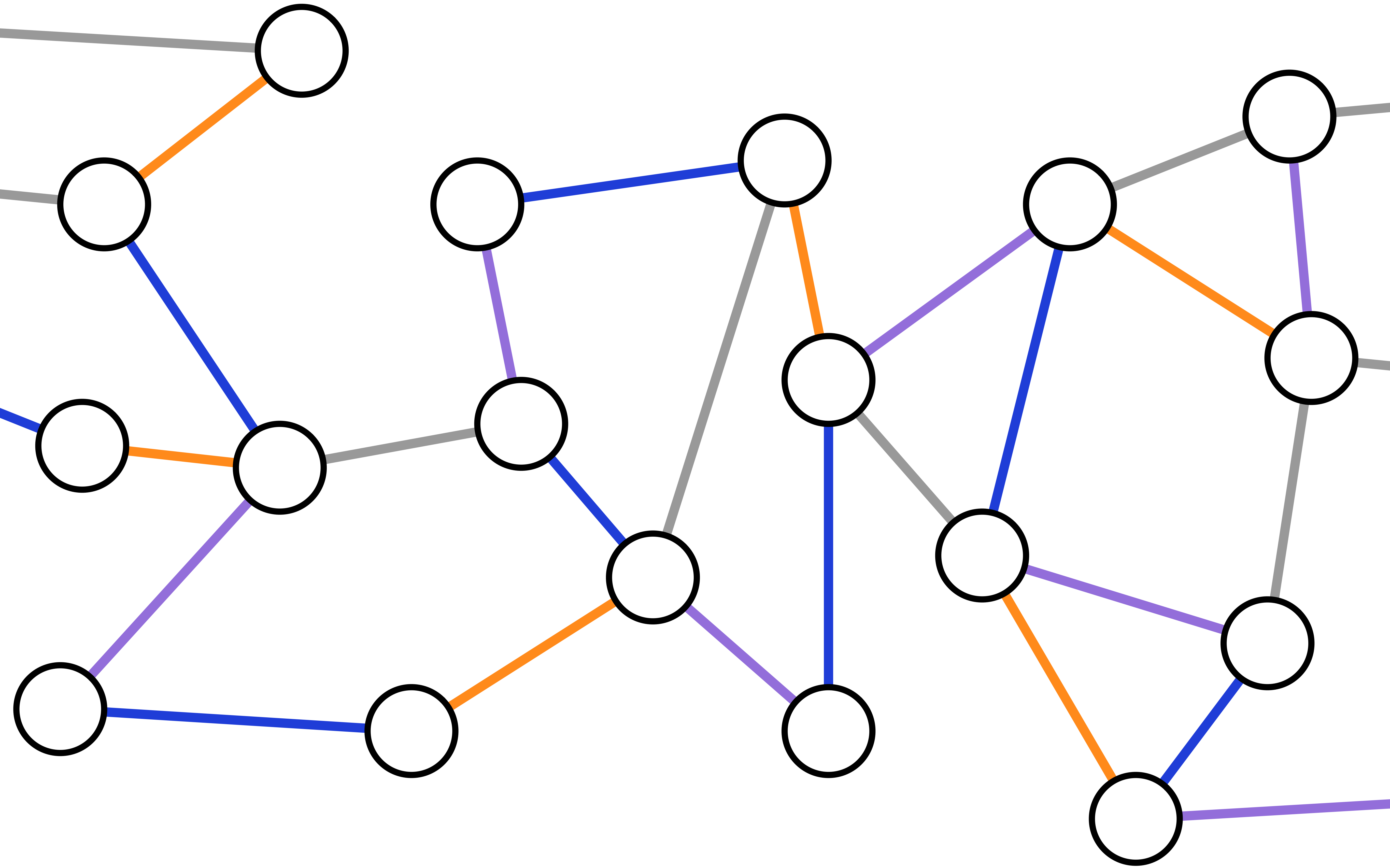
(A simple proposal algorithm)

Sometimes the simple algorithm is known to be optimal:

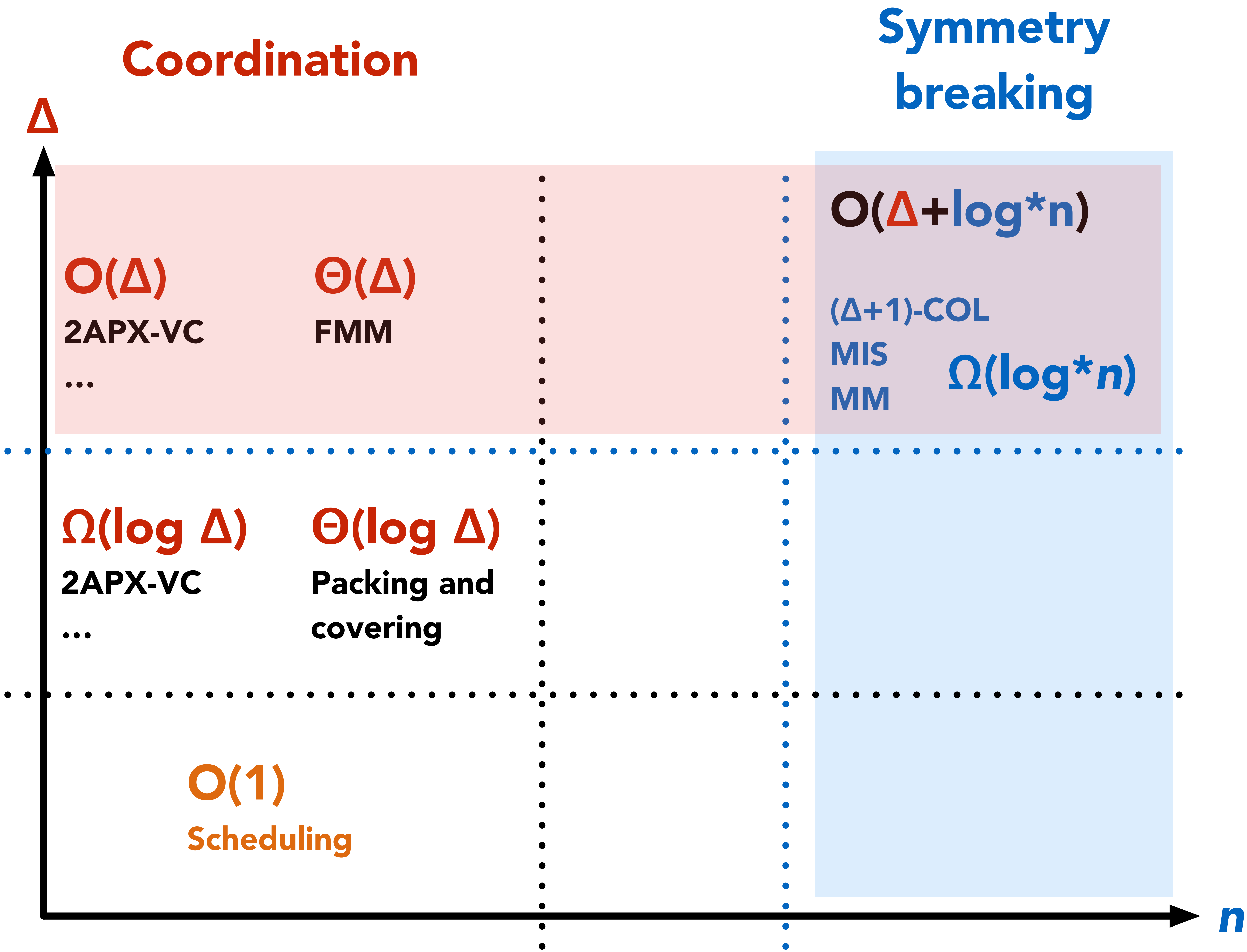
Maximal matching in (Δ+1)-edge colored Δ-regular graphs requires time $\Omega(\Delta)$

[H. and Suomela ’12]

This is tight as there is a trivial greedy algorithm



This is essentially what coordination looks like?


Coordination
Symmetry breaking
Δ
O(Δ+log*n)
O(Δ)
2APX-VC
...
Θ(Δ)
FMM
(Δ+1)-COL
MIS
MM
Ω(log*n)
Ω(log Δ)
2APX-VC
...
Θ(log Δ)
Packing and covering
O(1)
Scheduling
n

Distributed complexity of fast algorithms well understood as function of *n*

Distributed complexity as function of Δ not as well understood

Distributed complexity as function of both *n* and Δ not understood at all