Aalto University

Distributed Algorithms 2024

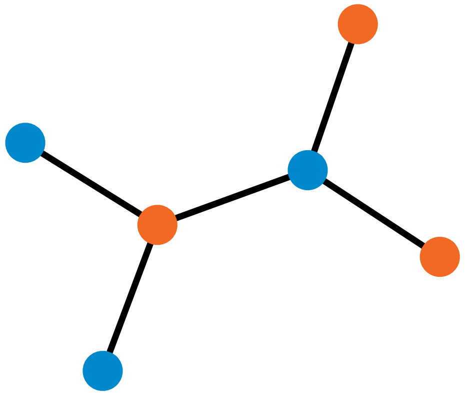
3 Port-numbering model

Port-numbered network
$N = (V, P, p)$

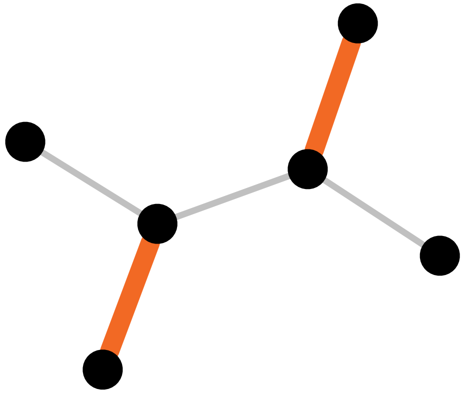
Distributed algorithm
$A = (\text{init}, \text{send}, \text{receive})$

Output of algorithm A in network N

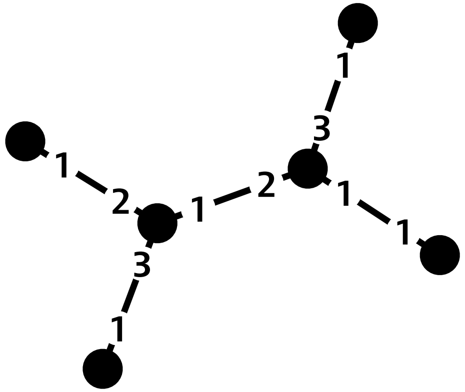
Bipartite maximal matching



Input:
proper 2-coloring



Output:
maximal matching

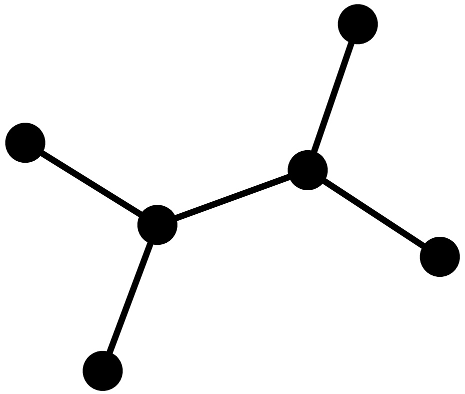


Model of computing:
PN model

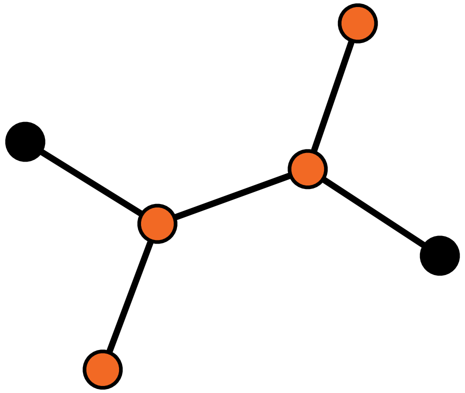
Algorithm

- *Orange* nodes send ***proposals*** to their neighbors, one by one
  - order by port numbers
- *Blue* nodes ***accept*** the first proposal they get, reject everything else
  - break ties by port numbers

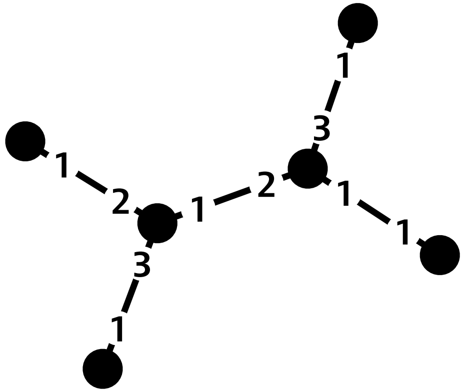
Vertex cover



Input:
nothing



Output: 3-approximation
of minimum vertex cover

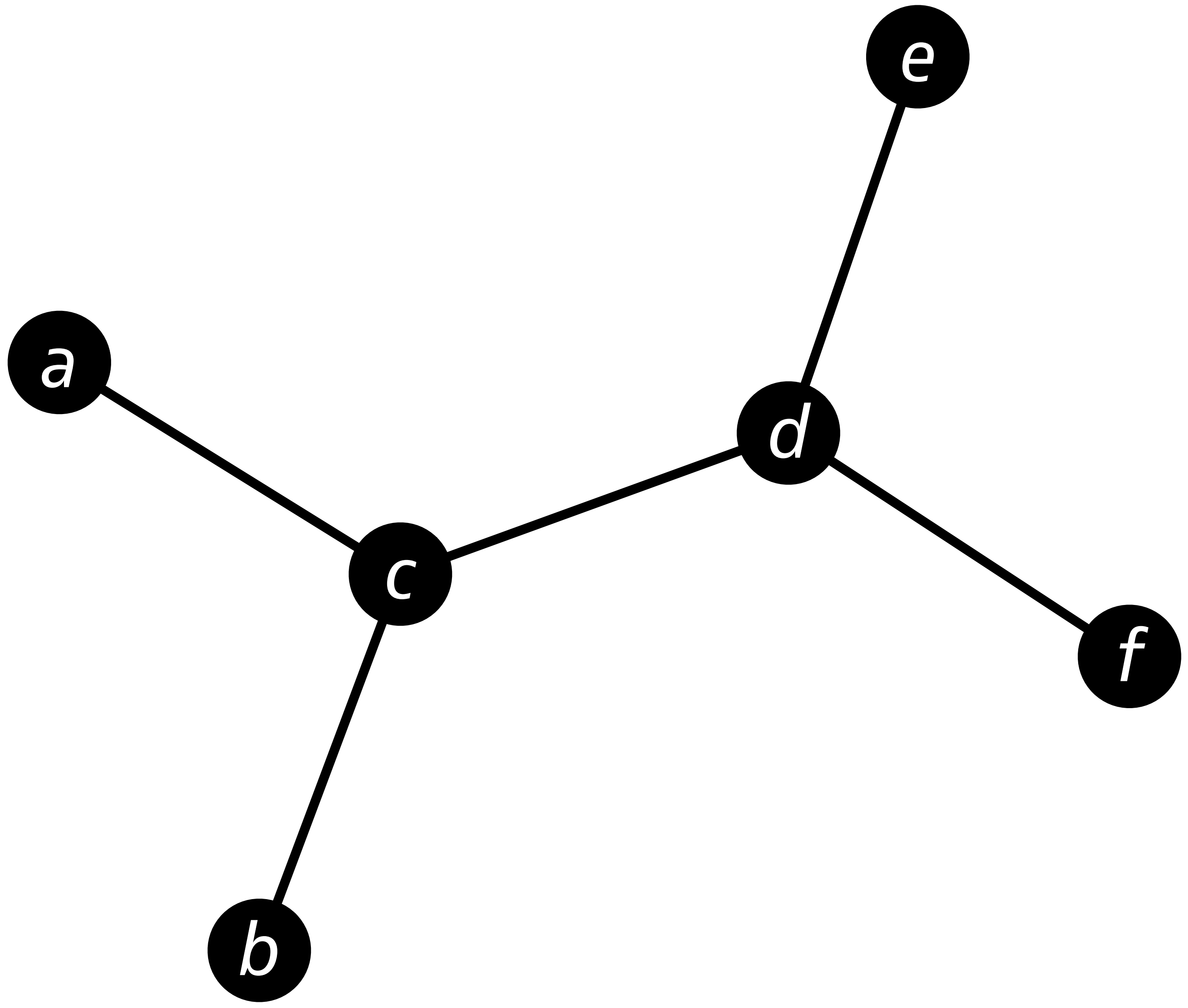


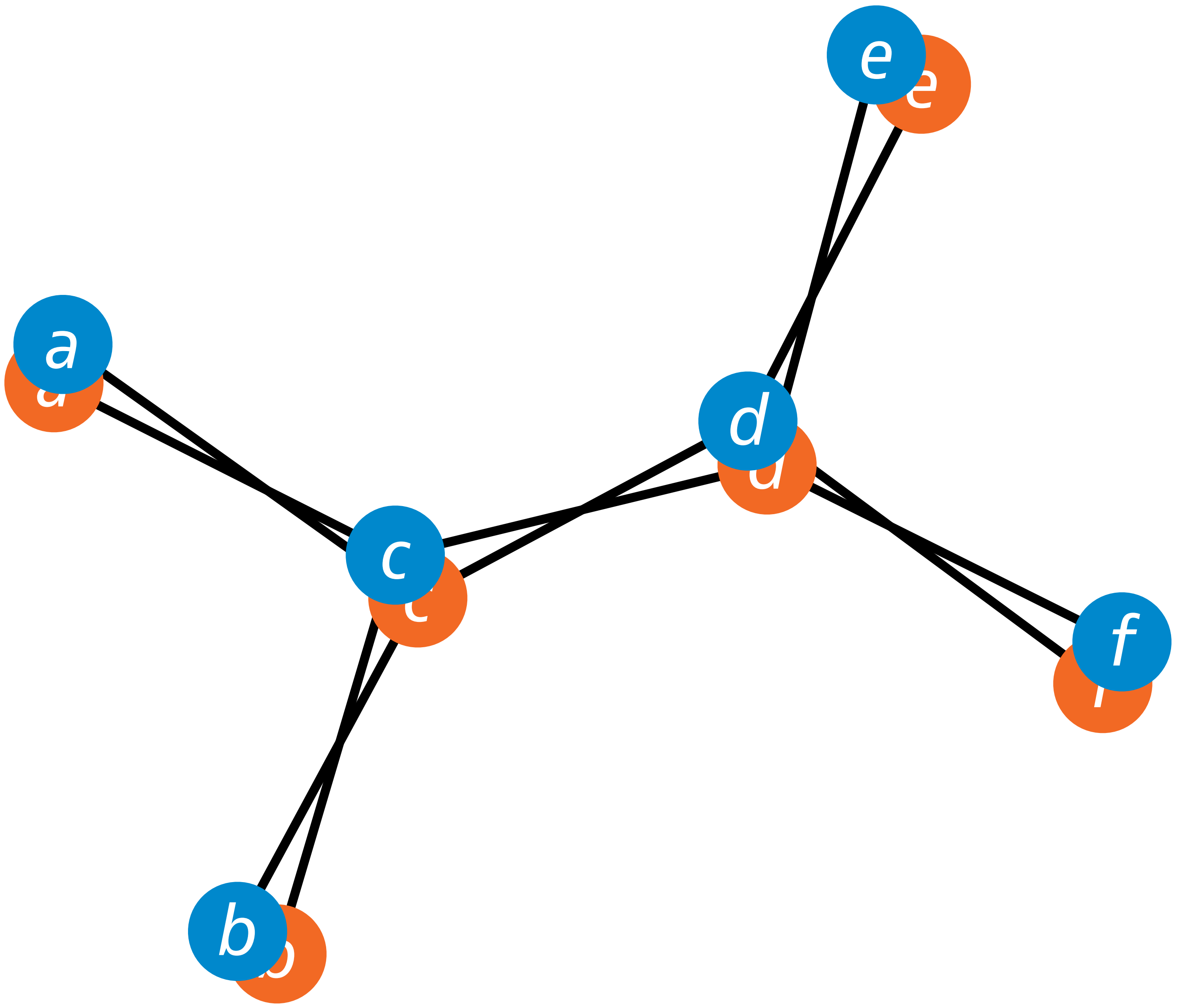
Model of computing:
PN model

Algorithm

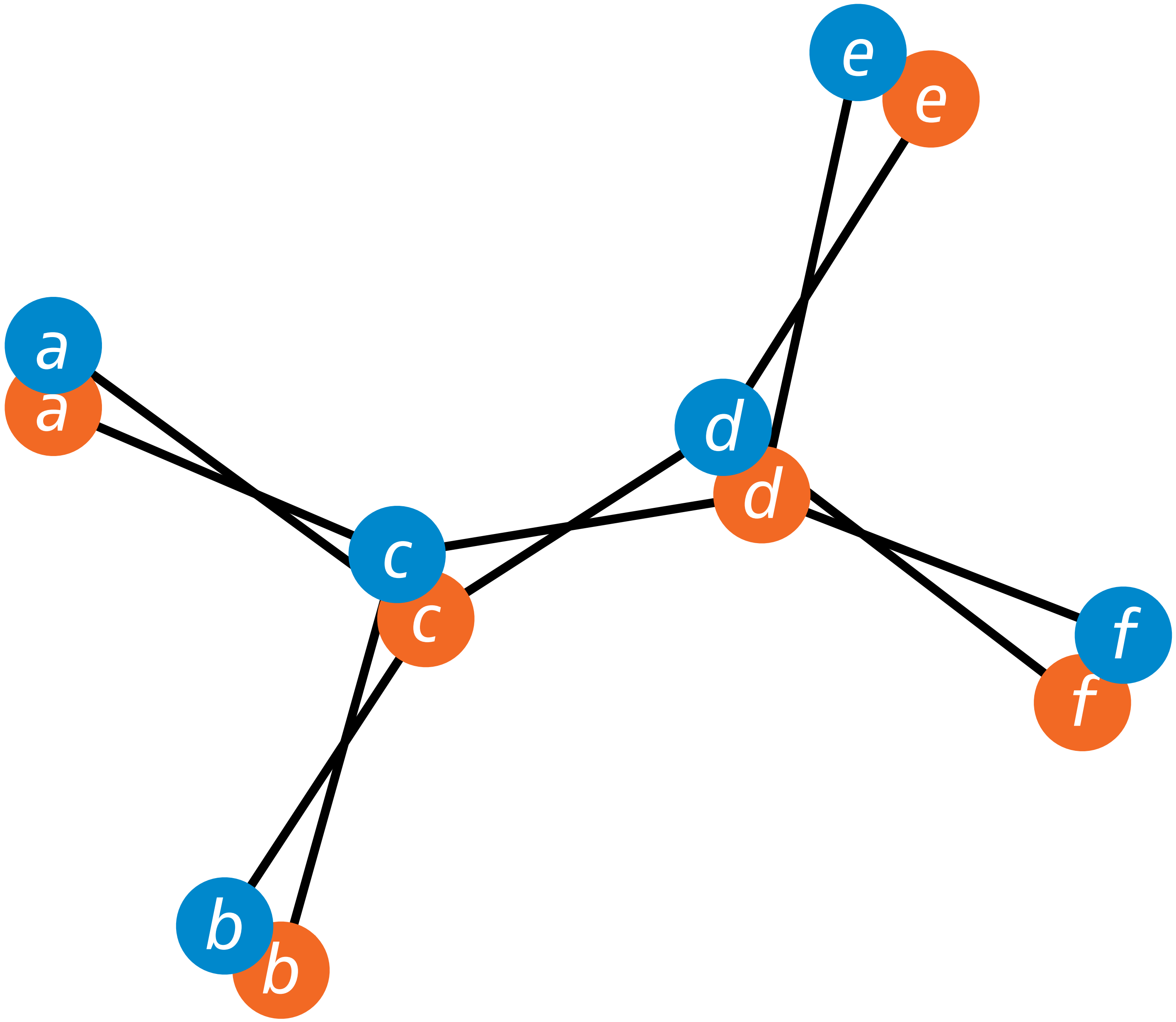
- Construct bipartite double cover G'
  - one node in G: two virtual copies in G'
  - one edge in G: two virtual copies in G'
- Find a maximal matching M' in G'
- Take all original nodes of G whose virtual copies are matched in M'

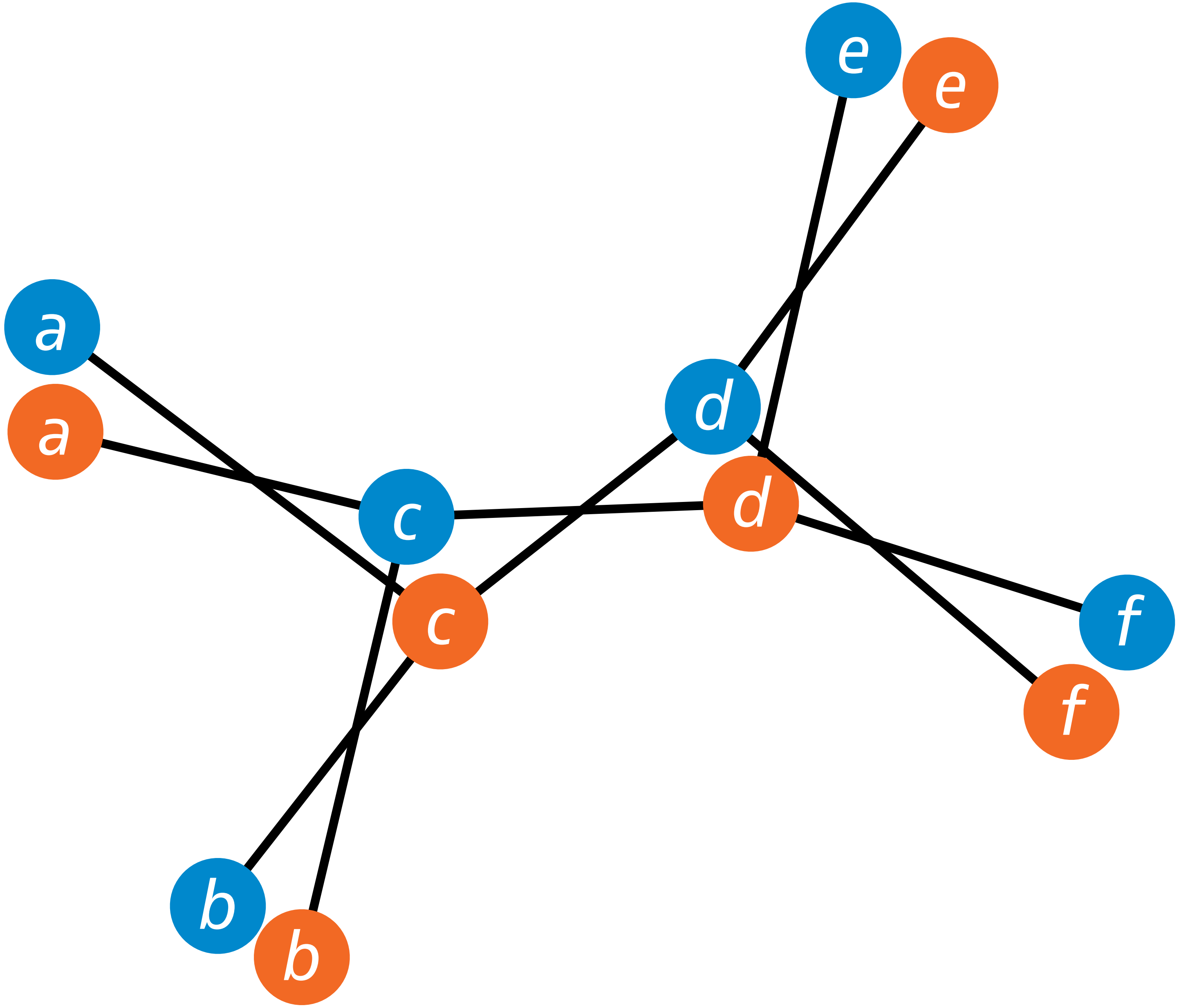
Graph G





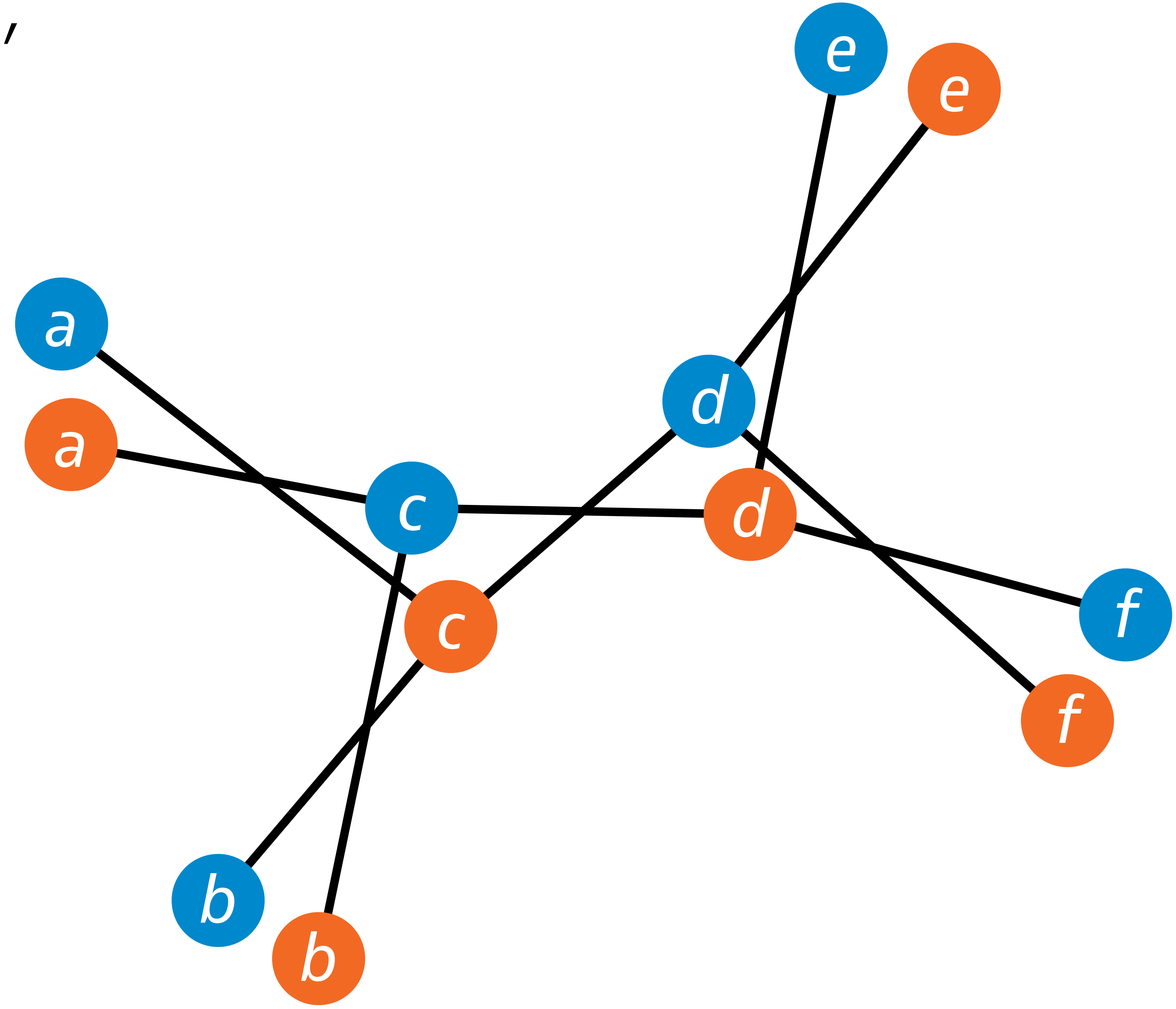
a
b
c
d
e
f


a
a
b
b
c
c
d
d
e
e
f
f

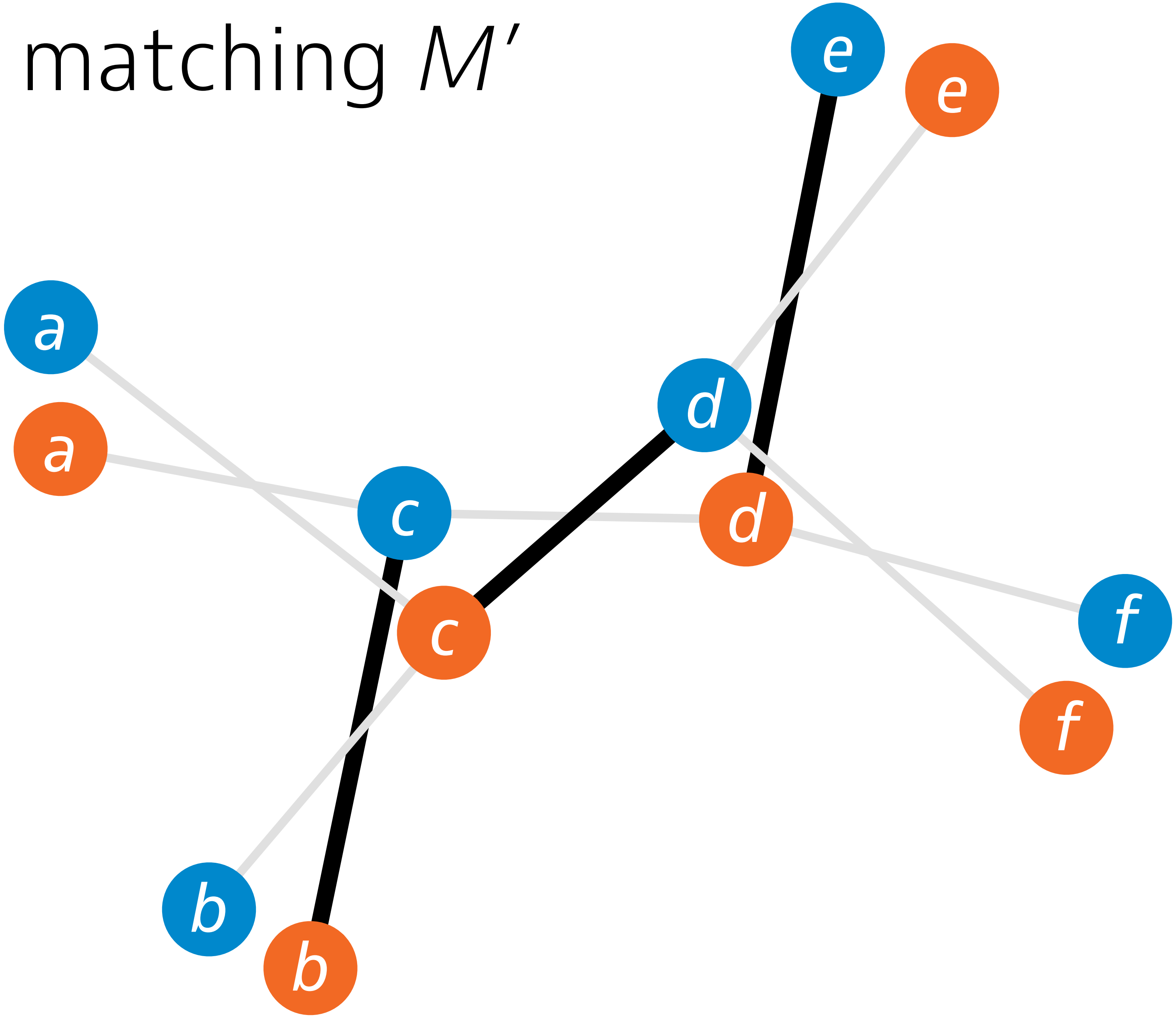


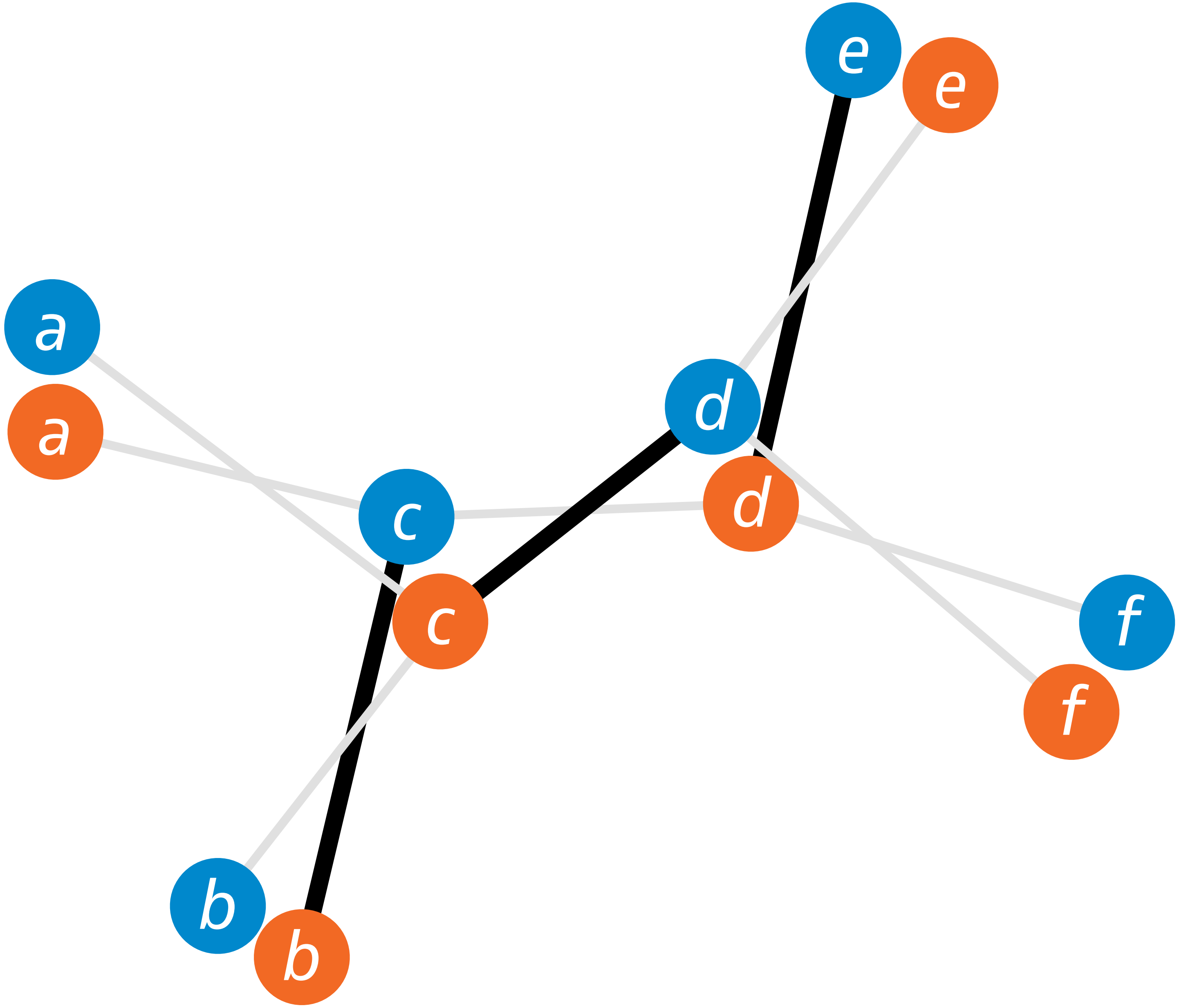
a
a
c
c
d
d
e
e
b
b
f
f

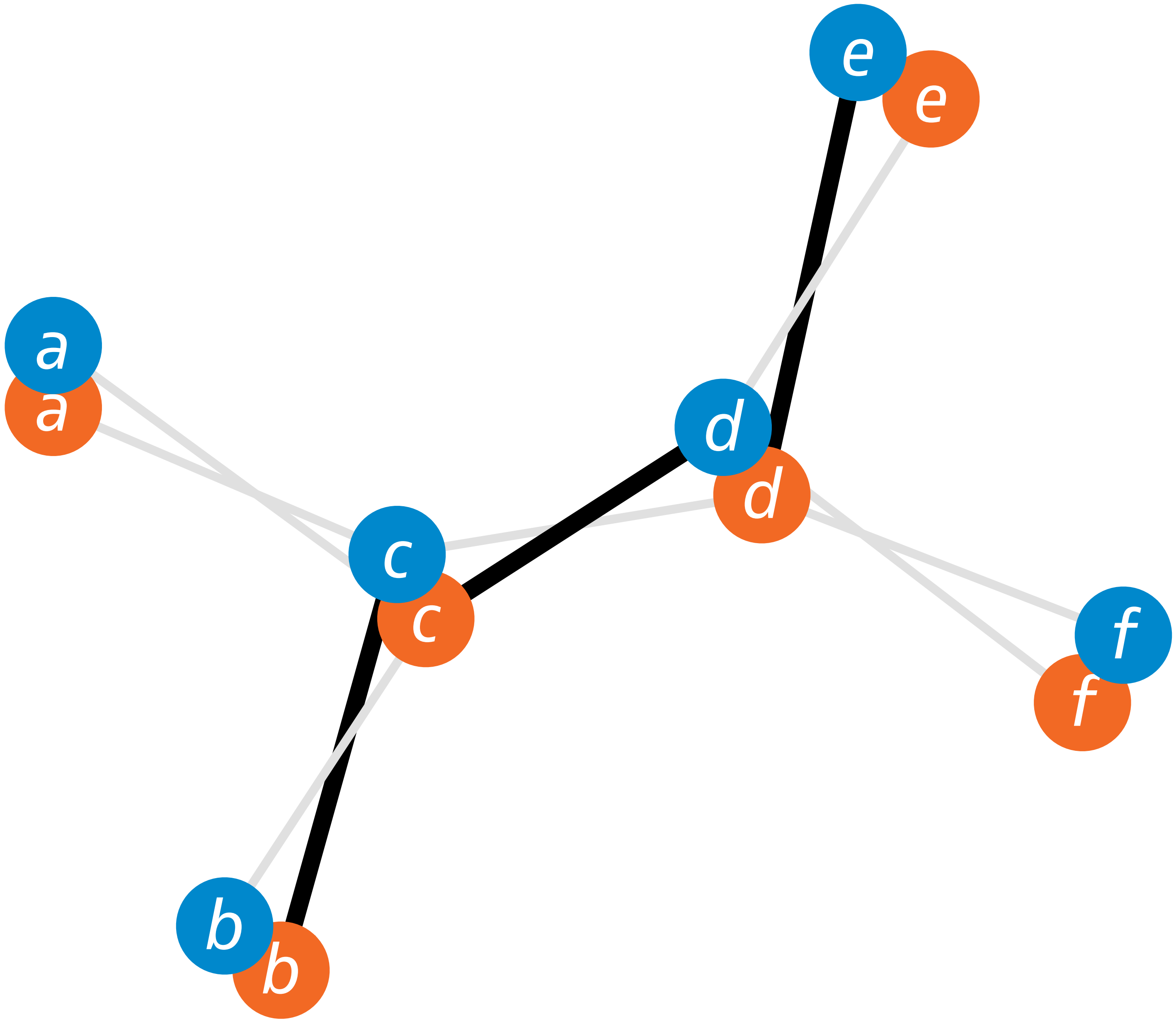
Graph *G*′

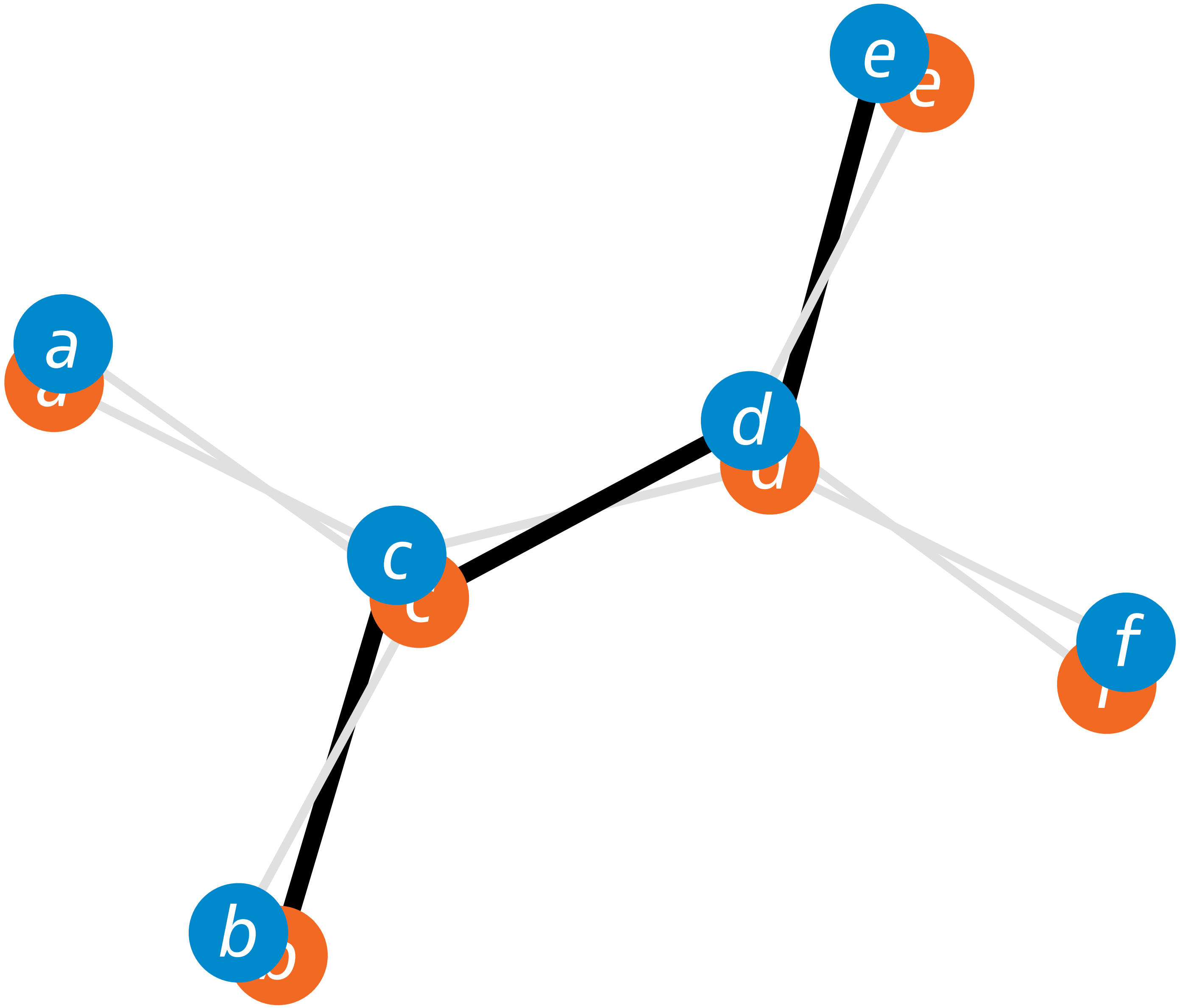


Maximal matching M'

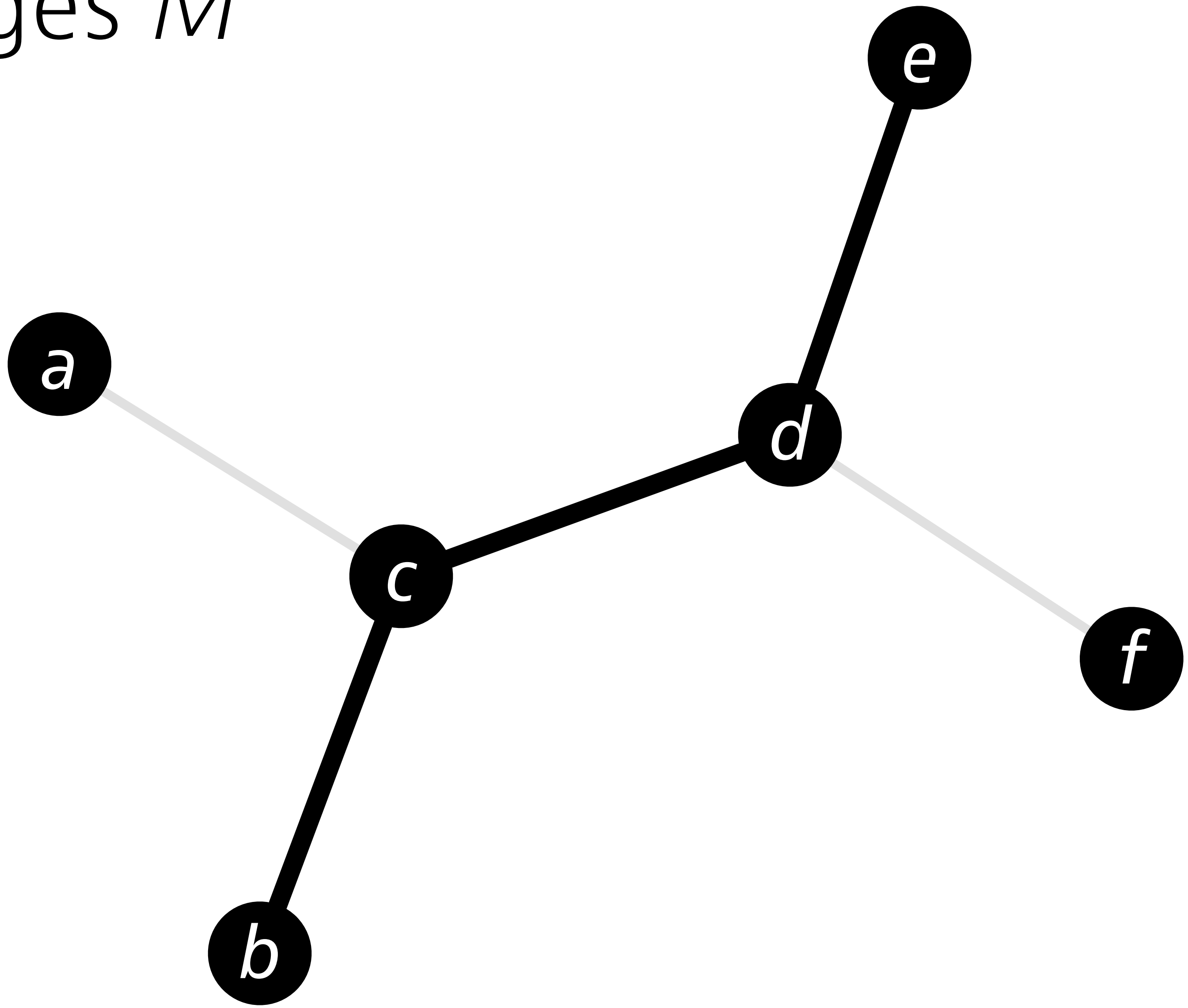



a
a
b
b
c
c
d
d
e
e
f
f

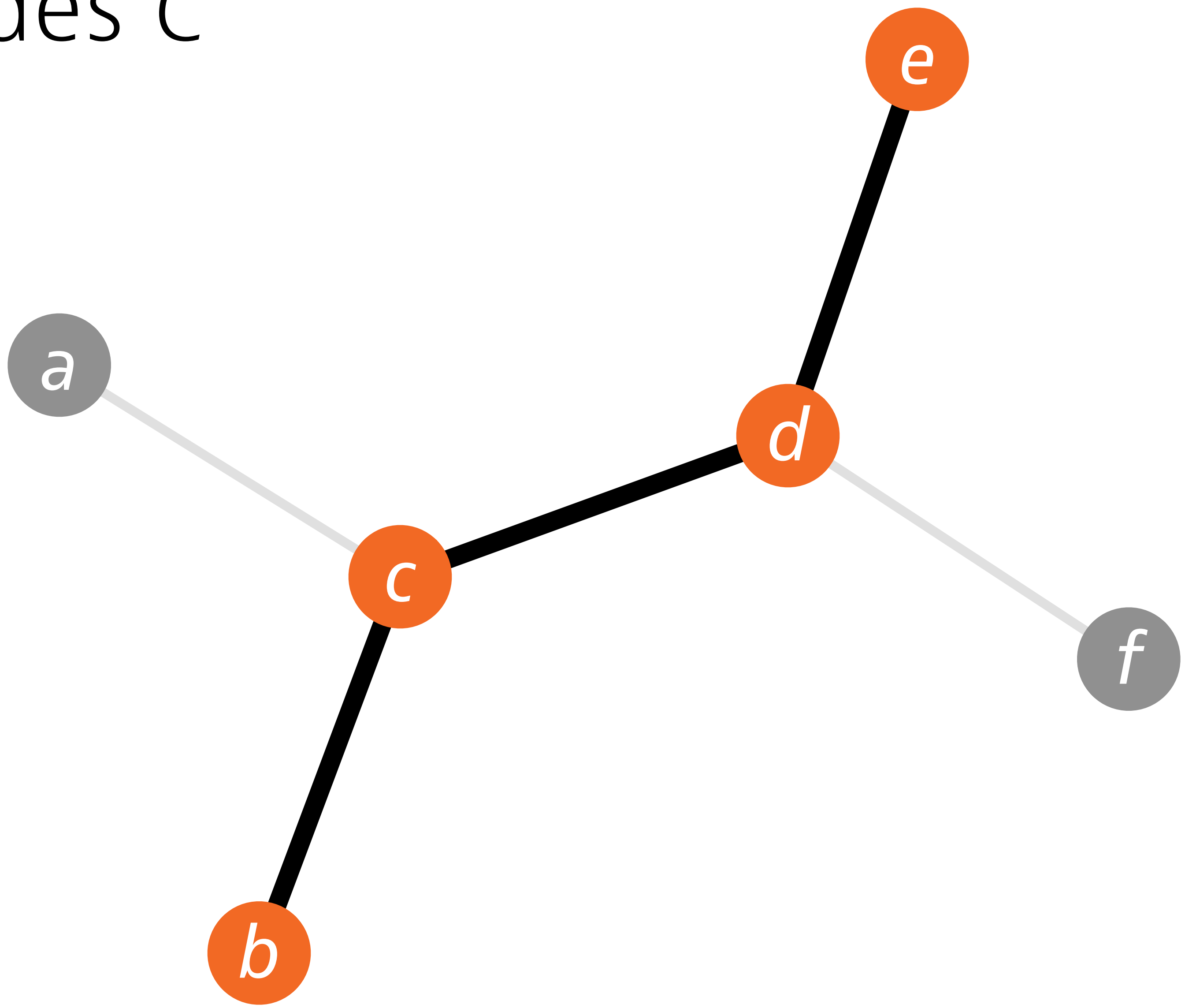

e
e
a
a
d
d
c
c
f
f
b
b


a
b
c
d
e
f

Set of edges *M*



Set of nodes C



Aalto University

Distributed Algorithms 2024

4 LOCAL model: Unique identifiers

LOCAL model
=
port-numbering model
+ unique identifiers

Nodes have distinct labels from $\{1, 2, \ldots, \text{poly}(n)\}$

LOCAL model

- Everything can be solved in diam(G)+1 rounds!
- Universal algorithm: ***"each node tells its neighbors everything it knows"***
  - **1 round:** everyone aware of its adjacent nodes and incident edges
  - ***T* rounds:** everyone aware of all nodes and edges within distance T
  - **diam(*G*)+1 rounds:** everyone knows G

LOCAL model

- Not so interesting:

  "What can be computed?"

- Very interesting:

  "What can be computed efficiently?"

  (efficient ≈ $o(\mathrm{diam}(G))$ rounds)

Coloring

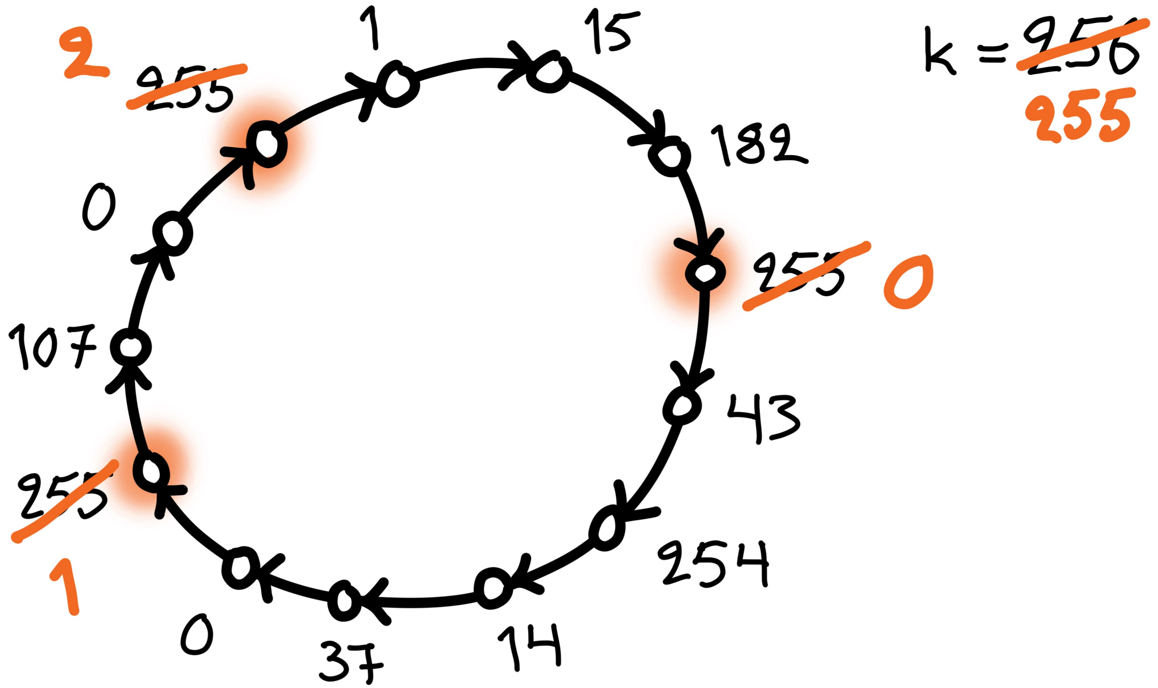
Input	Output	Rounds	Algorithm
Unique IDs	$O(\Delta^2)$-coloring	$O(\log^* n)$	Cover-free families
$O(\Delta^2)$-coloring	$O(\Delta)$-coloring	$O(\Delta)$	Rotating clocks
$O(\Delta)$-coloring	$(\Delta+1)$-coloring	$O(\Delta)$	Greedy color reduction
Unique IDs	$(\Delta+1)$-coloring	$O(\Delta + \log^* n)$	Combine these algorithms

Coloring

Input	Output	Rounds	Algorithm
Unique IDs	$O(\Delta^2)$-coloring	$O(\log^* n)$	Cover-free families
$O(\Delta^2)$-coloring	$O(\Delta)$-coloring	$O(\Delta)$	Rotating clocks
$O(\Delta)$-coloring	$(\Delta+1)$-coloring	$O(\Delta)$	Greedy color reduction
Unique IDs	$(\Delta+1)$-coloring	$O(\Delta + \log^* n)$	Combine these algorithms

Greedy color reduction

- If I am a ***local maximum***:
  - pick the smallest free color that is not used by any of my neighbors
- k+1 colors → k colors
  - provided that $k \geq \Delta+1$

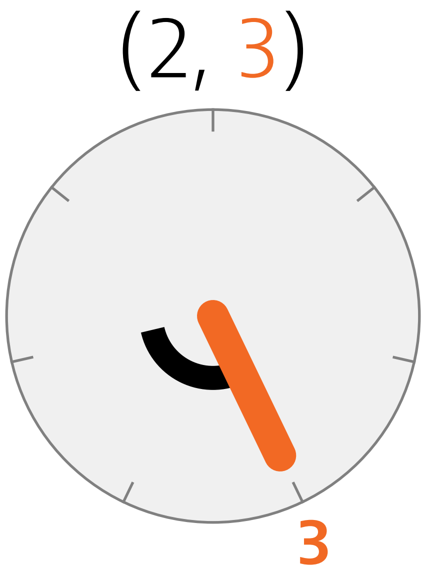


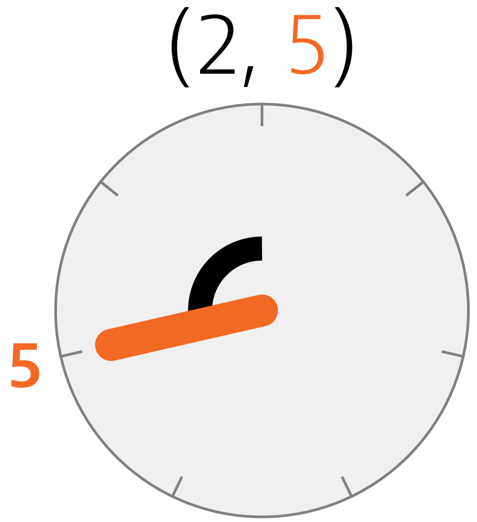
Coloring

Input	Output	Rounds	Algorithm
Unique IDs	$O(\Delta^2)$-coloring	$O(\log^* n)$	Cover-free families
$O(\Delta^2)$-coloring	$O(\Delta)$-coloring	$O(\Delta)$	Rotating clocks
$O(\Delta)$-coloring	$(\Delta+1)$-coloring	$O(\Delta)$	Greedy color reduction
Unique IDs	$(\Delta+1)$-coloring	$O(\Delta + \log^* n)$	Combine these algorithms

Rotating clocks

- q = prime, $q > 2\Delta$
- q^2 colors → q colors in q rounds
- If no conflicts:
  - $(a, b) \to (0, b)$
- Otherwise:
  - $(a, b) \to (a, b+a \bmod q)$



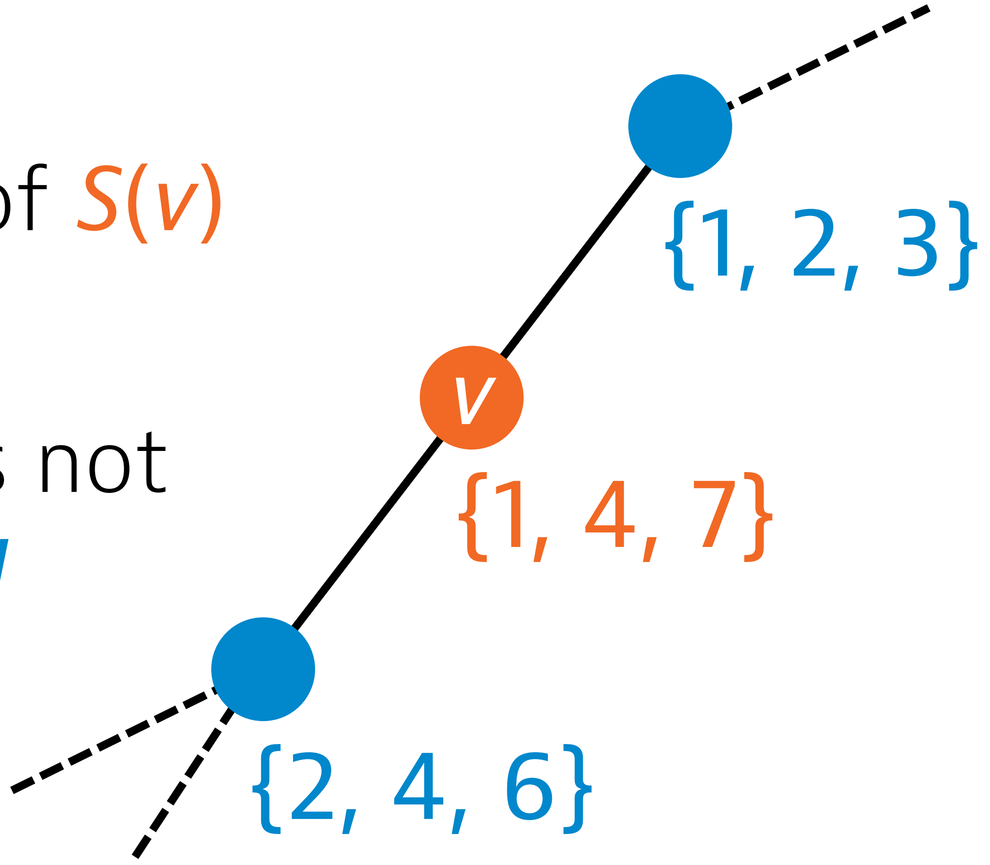


Coloring

Input	Output	Rounds	Algorithm
Unique IDs	$O(\Delta^2)$-coloring	$O(\log^* n)$	Cover-free families
$O(\Delta^2)$-coloring	$O(\Delta)$-coloring	$O(\Delta)$	Rotating clocks
$O(\Delta)$-coloring	$(\Delta+1)$-coloring	$O(\Delta)$	Greedy color reduction
Unique IDs	$(\Delta+1)$-coloring	$O(\Delta + \log^* n)$	Combine these algorithms

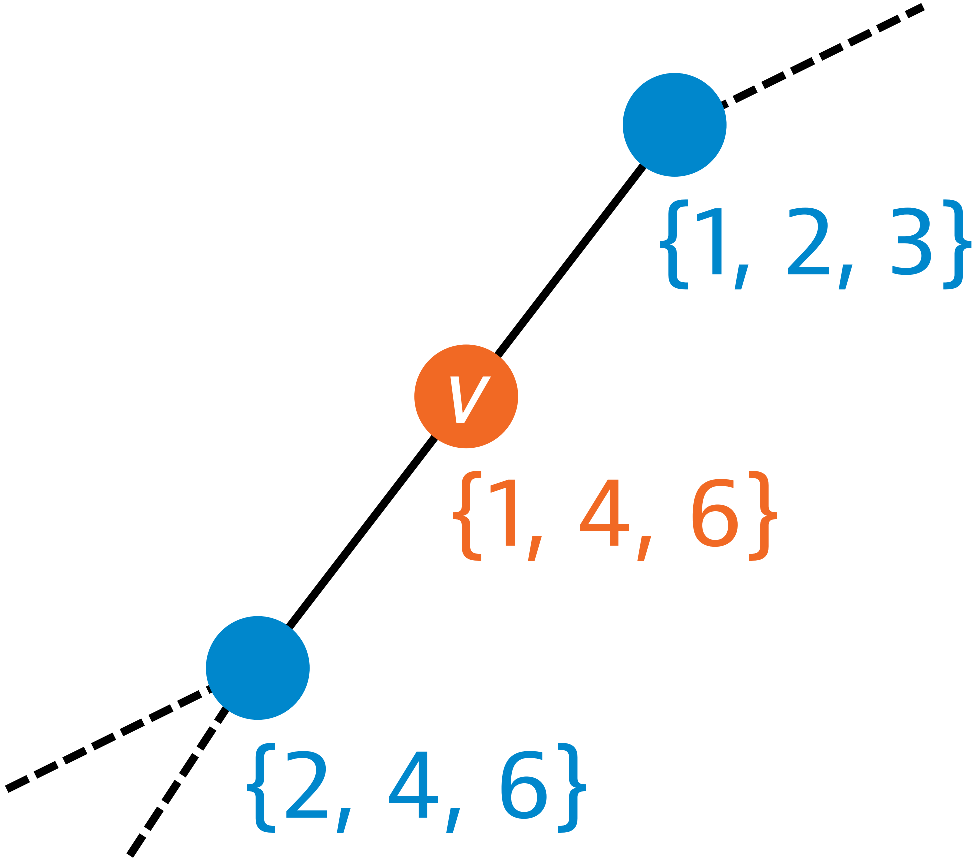
Cover-free families

- Old color of node v is a set $S(v) \subseteq \{1, 2, \ldots, m\}$
- **Promise:**
  - new color of v is an element of $S(v)$
- **Safe:**
  - pick an element of $S(v)$ that is not in any $S(u)$ for any neighbor u



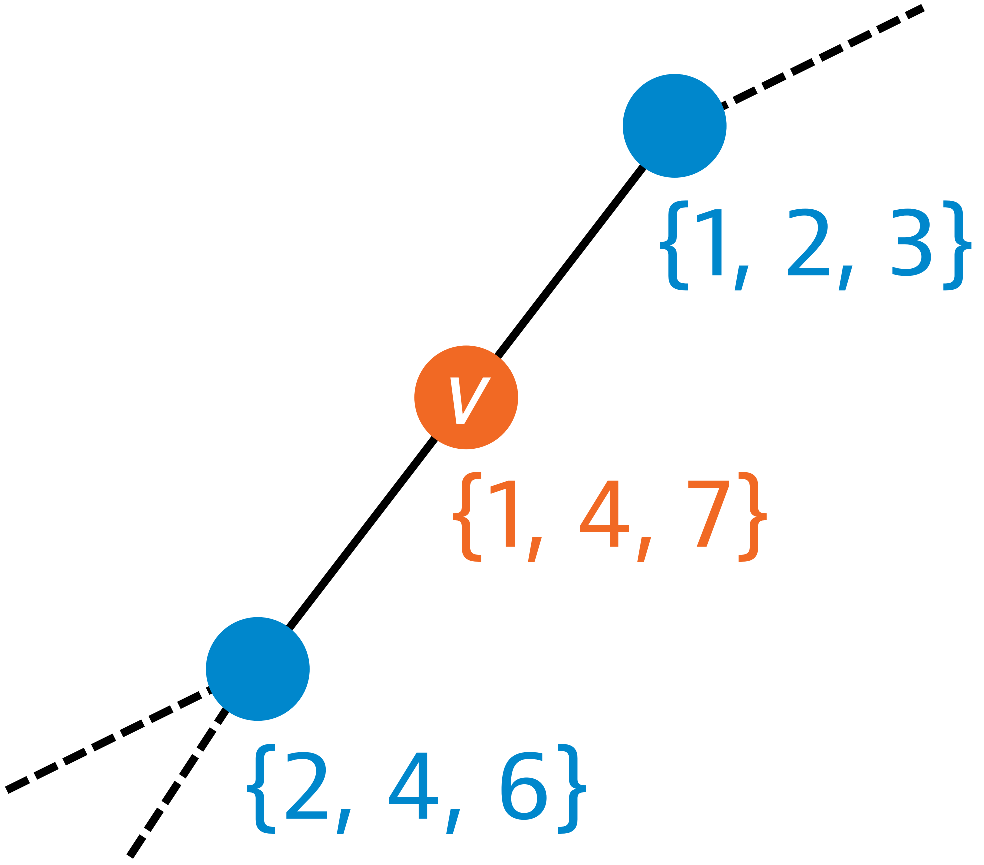
Cover-free families

- Old color of node v is a set $S(v) \subseteq \{1, 2, \ldots, m\}$
- **Bad:**
  - sets of neighbors cover all values in my set
  - no safe choice left



Cover-free families

- Old color of node v is a set $S(v) \subseteq \{1, 2, \ldots, m\}$
- **Good:**
  - my set is not fully covered by my neighbors
  - there is a safe choice



1-cover-free family

- For up to 1 neighbor these sets are good:

$$
\begin{aligned}
S_1 &= \{\,1,\ 2\,\} \\
S_2 &= \{\,1,\ 3\,\} \\
S_3 &= \{\,1,\ 4\,\} \\
S_4 &= \{\,2,\ 3\,\} \\
S_5 &= \{\,2,\ 4\,\} \\
S_6 &= \{\,3,\ 4\,\}
\end{aligned}
$$

2-cover-free family

- For up to 2 neighbors these sets are good:

$$
\begin{aligned}
S_1 &= \{\,1,\ 2,\ 3\,\} \\
S_2 &= \{\,3,\ 4,\ 5\,\} \\
S_3 &= \{\,5,\ 6,\ 7\,\} \\
S_4 &= \{\,1,\ 4,\ 7\,\} \\
S_5 &= \{\,2,\ 4,\ 6\,\}
\end{aligned}
$$

Cover-free families

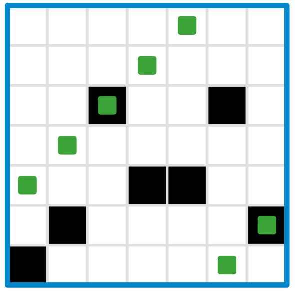
- Assume: $\boldsymbol{x}$-coloring, maximum degree $\boldsymbol{\Delta}$
- Assume: a $\boldsymbol{\Delta}$-cover-free family $S_1, S_2, \ldots, S_{\boldsymbol{x}}$
  - all subsets of $\{1, 2, \ldots, \boldsymbol{m}\}$
- Nodes of color c pick set S_c
- There is always a safe choice for any node!
- Color reduction from $\boldsymbol{x}$ to $\boldsymbol{m}$

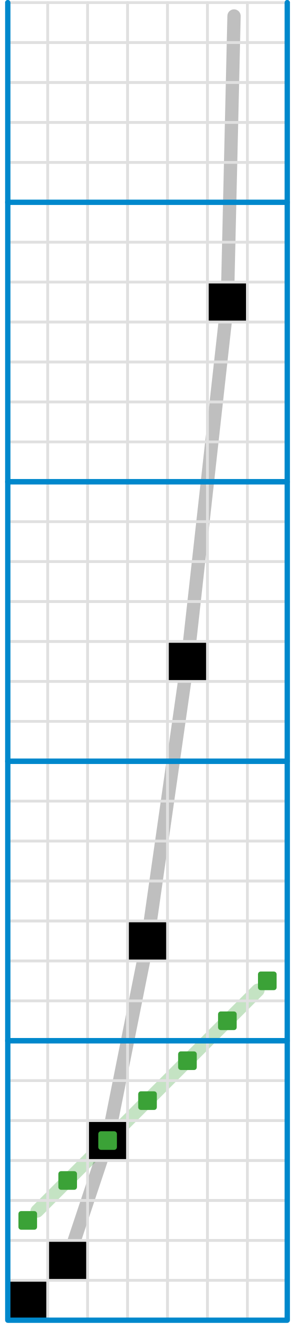
Cover-free families

- **Δ**-cover-free family $S_1, S_2, \ldots, S_{\boldsymbol{x}}$
  - all subsets of $\{1, 2, \ldots, \boldsymbol{m}\}$
- Good if:
  - **Δ** large → works in high-degree graphs
  - $\boldsymbol{x}$ large → tolerates many input colors
  - $\boldsymbol{m}$ small → produces a good output coloring
- E.g. $x = m$ is trivial (why?)

Constructing cover-free families

- q = prime, **GF(q) ≈ integers modulo q**
- f = degree-d **polynomial** over GF(q)
  - at most d points where $f(x) = g(x)$
  - q^{d+1} possible polynomials
- $S_f = \{ (x, f(x)) \mid x = 0, 1, \ldots, q-1 \}$
  - base set: all q^2 possible pairs (x, y)
  - q^{d+1} possible subsets, each with q elements
  - intersection of S_f and S_g has size at most d
- If $q > \Delta d$: a Δ-cover-free family
  - why?





Cover-free families

- Construct Δ-cover-free families with suitable parameters

- $n \to \approx \Delta^2 \log^2 n$
  $\to \approx \Delta^2 \log^2 \log n$
  $\to \approx \Delta^2 \log^2 \log \log n$
  . . .
  $\to \approx \Delta^2 \log^2 \Delta$
  $\to \approx \Delta^2$

} $O(\log^* n)$ steps

Coloring

Input	Output	Rounds	Algorithm
Unique IDs	$O(\Delta^2)$-coloring	$O(\log^* n)$	Cover-free families
$O(\Delta^2)$-coloring	$O(\Delta)$-coloring	$O(\Delta)$	Rotating clocks
$O(\Delta)$-coloring	$(\Delta+1)$-coloring	$O(\Delta)$	Greedy color reduction
Unique IDs	$(\Delta+1)$-coloring	$O(\Delta + \log^* n)$	Combine these algorithms