Fault-tolerant coloring of the asynchronous cycle

Pierre Fraigniaud  Patrick Lambein-Monette  Mikaël Rabie

IRIF, Université Paris Cité

DUCAT + ESTATE meeting
18 March 2022
Setup
Takeaway

Contributions

- define the **asynchronous k-coloring problem** for asynchronous networks
- propose a **wait-free** algorithm for any \((n \geq 3)\)-nodes cycle \(C_n\)
- using a **6-color** palette
- running in \(O(\log^* n)\) (asynchronous) rounds

Unifies

- synchronous **graph k-coloring**
  - **LOCAL model**
  - \(\sim \Omega(\log^* n)\) temporal lower bound
- asynchronous **k-renaming**
  - **immediate snapshot** shared-memory model
  - \(\sim\) coloring \(C_3\) **requires** \((k \geq 5)\) **colors**
### Model

**async-LOCAL**
- $n$ asynchronous **processes** $p_1, \ldots, p_n$
- connected **graph** $G = (V := [n], E)$
- **schedule** $\sigma = \sigma(1), \sigma(2), \ldots \in \Sigma \subseteq 2^V$
- $i \in \sigma(t) \iff p_i \text{ activated at } t$:
  1. **writes** a value
  2. **reads** values of $p_j, j \sim_G i$
  3. privately **computes** a next state

**Within one step $t$**
1. **first** activated process **all write**
2. **then** activated process **all read**
   - if $i \sim j$ and $i, j \in \sigma(t)$, then $i$ reads $j$'s **step $t$ value**

---

**no schedule constraints** ($\sigma$ arbitrary)

![Diagram of process activation and communication](attachment:diagram.png)

- **write**
- **read**
- **think**

---

**time**

**instant $t$**

Problem

async $k$-coloring

For a graph $G = (V, E)$ and scheduler $\Sigma \subseteq V^{\mathbb{N}}$.

- **uniform termination** $\exists B : |\sigma|_i \geq B \implies p_i$ outputs $c_i \neq \bot$
- **validity** if $p_i \sim p_j$ both output, then $c_i \neq c_j$
- **$k$-palette** $c_i = \bot \lor c_i \in \{1, \ldots, k\}$

Assuming initial unique identifiers $(X_u)_{u \in V}$.

\[ \sim \text{round complexity} \text{ (# of activations before a process returns)} \]

Definition (wait-free)

An algorithm solves async-$k$-coloring **wait-free** over the graph $G$ if it solves it for the complete scheduler $\Sigma = 2^{V(G) \times \mathbb{N}}$.

...also for a graph class $\mathcal{G}$:

- e.g., **cycles** $\mathcal{C} = \{C_n : n \geq 3\}$
- e.g., **cliques** $\mathcal{K} = \{K_n : n \geq 2\}$
## Takeaway

### Contributions

- Define the **asynchronous k-coloring problem** for asynchronous networks
- Propose a **wait-free** algorithm for any \((n \geq 3)\)-nodes cycle \(C_n\)
- Using a **6-color** palette
- Running in \(O(\log^* n)\) (asynchronous) rounds
Related works
The **LOCAL** model

- **LOCAL**
  - \( n \) nodes \( p_1, \ldots, p_n \) with **unique identifiers** \( X_1, \ldots, X_n \)
  - connected graph \( G = (V = [n], E) \)
  - in each round \( t \geq 1 \), every node \( p_i \):
    1. **sends** a message to its neighbors
    2. **receives** each neighbor’s **round** \( t \) message
    3. privately **computes** a next state
  - all nodes run for \( T \) rounds, then output

- **What can be computed **locally**
  \((\equiv T = o(n))\)
The $k$-coloring problem

| **graph $k$-coloring** | **Fundamental results on $C$** |
|------------------------|------------------|
| For a graph $G = (V, E)$ | |
| ▶ **termination** every node $p_i$ outputs some color $c_i$ | ▶ 2-coloring **non-local** |
| ▶ **validity** if $i \sim j$, then $c_i \neq c_j$ | ▶ 3-coloring $C_n$ **requires** $\frac{1}{2} \log^* n + O(1)$ rounds (Linial 92) |
| ▶ **$k$-palette** $c_i \in \{1, \ldots, k\}$ | ▶ 3-coloring $C_n$ **can be solved** in $\frac{1}{2} \log^* n + O(1)$ rounds (Cole+ 86) |

typically, $k = \Delta + 1$, $\Delta := \deg(G)$
Cole and Vishkin’s algorithm

**Algorithm 1: 3-coloring, code for $p_i$**

1. **Input**: $X_i \in \text{Poly}(n)$, unique identifier
2. **for** $T = \Theta(\log^* n)$ **rounds** **do**
   3. **write**($X_i$) and **read** ($X_{i-1}, X_{i+1}$)
   4. $X_i \leftarrow f(X_i, X_{i+1})$
5. **▷** Here $X_i \leq 5$
6. **for** $k \in (5, 4, 3)$ **do**
   7. **write**($X_i$) and **read** ($X_{i-1}, X_{i+1}$)
   8. **if** $X_i = k$ **then**
      9. $X_i \leftarrow \min \mathbb{N} \setminus \{X_{i-1}, X_{i+1}\}$
9. **return**($X_i$)

$f(x, y) = 2\ell + x\ell$, $\ell := \min\{i : x_i \neq y_i\}$

$x = \sum_{i \geq 0} 2^i x_i$, $y = \sum_{i \geq 0} 2^i y_i$

- each application of $f$ logarithmically reduces $\max|X_i|$
- ...as long as $X_i \geq 6$
- final phase in $O(1)$
### The IS (*immediate snapshot*) model

| immediate snapshot | Within one step $t$ |
|--------------------|-------------------|
| $n$ asynchronous **processes** $p_1, \ldots, p_n$ | **1.** **first** activated process **all write** |
| shared-memory **array** $M$ | **2.** **then** activated process **all read** |
| **schedule** $\sigma(1), \sigma(2), \ldots \subseteq [n]$ |  
| $i \in \sigma(t) \iff p_i \text{ activated at } t$: |  
| 1. **writes** in $M[i]$ | **if** $i, j \in \sigma(t)$, **then** $i$ reads $M[j](t)$ |
| 2. **reads** entire array $M$ |  
| 3. if $M \models \mathcal{P} \text{ terminates}$ with | **what can be computed** given $(n, \Sigma)$? |
| output $f(M)$ | e.g., no wait-free consensus |
| 4. else privately **computes** a next state |
The $k$-renaming problem

| async $k$-coloring | Fundamental results |
|---------------------|---------------------|
| For initial unique names $X_1, \ldots, X_n < M$ and scheduler $\Sigma \subseteq V^\mathbb{N}$. | $k$-renaming is impossible when $k < 2n - 1$ and $n = p^m$, $p$ prime (Herlihy+ 99, Castañeda+ 10) |
| - **uniform termination** $\exists B :$ $|\sigma|_i \geq B \implies p_i$ outputs $c_i \neq \bot$ | - (2$n - 1$)-renaming can be solved for all $n \geq 2$ (Attiya+ 90, Attiya+ 04) |
| - **unicity** if $p_i \sim p_j$ both output, then $c_i \neq c_j$ | |
| - **$k$-palette** $c_i = \bot \lor c_i \in \{1, \ldots, k\}$ | |
| **wait-free** when $\Sigma = 2^{V \times \mathbb{N}}$ | |
Attiya and Welch’s algorithm

Algorithm 2: $2n - 1$ renaming, code for $p_i$

1. **Input:** $X_i \in \{0, 1, \ldots, M - 1\}$
2. **Initially:** $c_i \leftarrow 0$
3. **Forever:**
   - write($X_i, c_i$)
   - read($((X_1, c_1), \ldots, (X_n, c_n))$)
   - if $c_i \notin \{c_j : j \neq i\}$ then return($c_i$)
   - else
     - $r_i \leftarrow |\{j : X_j < X_i\}|$
     - $c_i \leftarrow r_i$-th min of $\mathbb{N} \setminus \{c_1, \ldots, c_n\}$

- $c_i \leq 2n - 1$
- active process $p_i$ with smallest $X_i$ cannot work forever
- all processes eventually terminate
A tale of two models

both problems inform our study:

**LOCAL model**
- coincides with our model when $\Sigma = (V, V, \ldots)$
- any async-$k$-coloring algorithm is a $k$-coloring algorithm
- $\Omega(\log^* n)$ **rounds** necessary to color the cycle $C_n$

**IS model**
- coincides with our model when $G = K_n$
- any async-$k$-coloring algorithm is a $k$-renaming algorithm for $G = C_3 = K_3$
- **5-color palette** necessary to color the cycle $C_3$
Takeaway

Contributions

▶ define the **asynchronous** $k$-**coloring problem** for asynchronous networks
▶ propose a **wait-free** algorithm for any ($n \geq 3$)-nodes cycle $C_n$
▶ using a **6-color** palette
▶ running in $O(\log^* n)$ (asynchronous) rounds

Unifies

▶ synchronous **graph** $k$-**coloring**
  ▶ **LOCAL** model
  ▶ $\sim \Omega(\log^* n)$ temporal lower bound
▶ asynchronous $k$-**renaming**
  ▶ **immediate snapshot** shared-memory model
  ▶ $\sim$ coloring $C_3$ **requires** ($k \geq 5$) **colors**
Algorithmic contributions
async-6-coloring ("slow" worst-case)

Algorithm 3: async 6-coloring, code for $p_i$

1. **Input**: $X_i \in \mathbb{N}$, proper coloring
2. **Initially**: $c_i = (a_i, b_i) \leftarrow (0, 0)$
3. **Forever**:
   4. write($X_i, c_i$) \textbf{immediate snapshot}
   5. read($((X, c), (X', c'))$)
   6. if $c_i \notin \{c, c'\}$ then return($c_i$)
   7. else
      8. $a_i \leftarrow \min \mathbb{N} \setminus \{a_j : (X_j > X_i)\}$
      9. $b_i \leftarrow \min \mathbb{N} \setminus \{b_j : (X_j < X_i)\}$

- $a_i + b_i \leq 2 \implies 6$-colors palette
- local maxima/minima stubbornly keep $a_i = 0/b_i = 0$
- local extrema terminate in $O(1)$
- process terminate in $O(\ell)$, $\ell$ distance to a local extremum
Next?

- 5-coloration
- general graphs
- other problems
async-6-coloring ("fast" worst-case)

**Algorithm 4: async 6-coloring, code for $p_i$**

1. **Input:** $X_i \in \mathbb{N}$ > proper coloring
2. **Initially:** $c_i = (a_i, b_i) \leftarrow (0, 0), r_i \leftarrow 0$
3. **Forever:**
   - **Write**($X_i, c_i, r_i$) > immediate snapshot
   - **Read**($((X, c, r), (X', c', r'))$)
     > update $c_i$ as before
   - if $(r_i < \infty) \land (r_i \leq \min\{r, r'\})$ then
     - if $\min\{X, X'\} < X_p < \max\{X, X'\}$ then
       - $r_i \leftarrow r_i + 1$
       - $Y \leftarrow f(X_i, \min\{X, X'\})$
       - if $Y < \min\{X_q, X_{q'}\}$ then $X_p \leftarrow Y$
     - else
       - $r_i \leftarrow \infty$
       - if $X_i < \min\{X, X'\}$ then
         - $X_i \leftarrow \min\{X_i, \min(\mathbb{N} \setminus \{f(X, X_i), f(X', X_i)\})\}$