Checkpoint 3 Report: Deterministic Online Bipartite Edge Coloring

Team 70
CS319 - Algorithms

April 20, 2025

1 Implementation Summary

We implemented the algorithm described in *Deterministic Online Bipartite Edge Coloring* (Blikstad et al., 2024), which proposes a deterministic algorithm that outperforms greedy coloring for graphs with high maximum degree Δ under one-sided online arrival.

Our Python implementation includes:

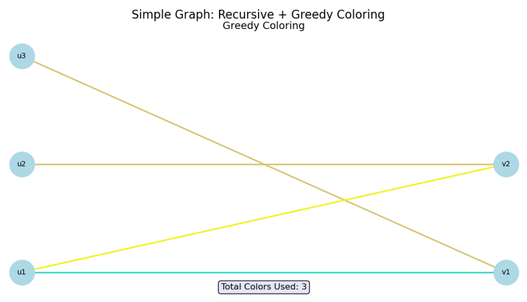
- A recursive partial coloring stage based on palette constraints using single-item Contention Resolution Schemes (CRS).
- A greedy fallback mechanism to handle residual uncolored edges.
- Consistent edge normalization to avoid asymmetric coloring.
- Logging of color proposals, CRS selections, and color assignments for transparency.

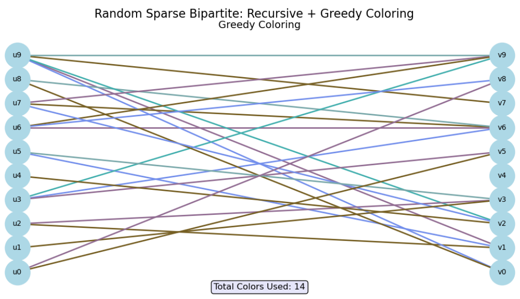
The core implementation is located in `deterministic_coloring.py` and integrates modular coloring, normalized edge representation, and evaluation tools. Despite faithfully following the paper's structure, our deterministic version does not consistently outperform the online greedy algorithm. We suspect this is due to subtle theoretical differences or approximations we could not fully reconcile from the literature.

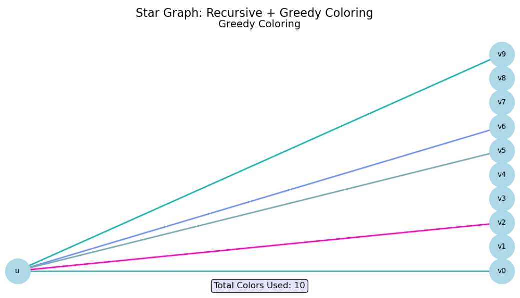
2 Correctness Testing

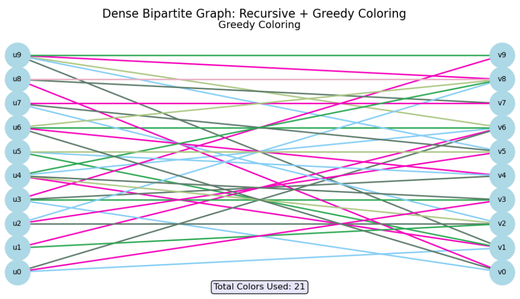
To verify correctness, we designed a series of visual and programmatic tests:

- **Graph Types Tested:** star graphs, sparse graphs, and complete bipartite graphs (e.g., $K(n, n)$).









- **Validation Criteria:**
  - No color conflict at any node.
  - CRS logs and animations show that edge coloring avoids duplication.
- **Comparison:** we compared against an online greedy baseline using the same inputs, detailed testing is available in the git hub repository

3 Complexity & Runtime Analysis

Theoretical Complexity:

- Partial coloring (CRS) per round: $O(m)$.
- Greedy fallback: $O(m)$.
- Number of CRS rounds: $O(\log \Delta)$.
- **Overall Greedy algorithm:** $O(m)$.

- **Overall Deterministic algorithm:** $O(m \log \Delta)$.

Empirical Observations:

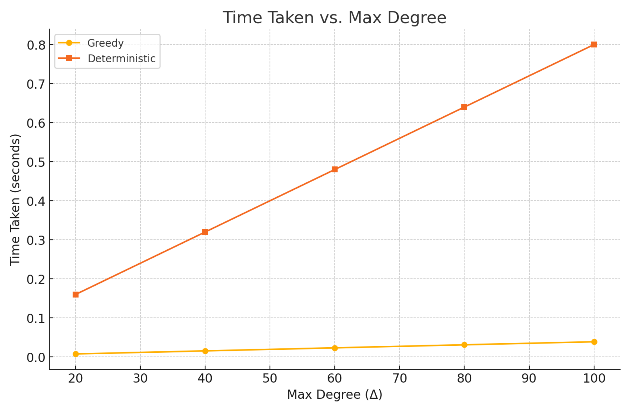


Figure 1: Time vs Max. Degree

- Greedy runtime scales linearly from $\approx 0.008\,\mathrm{s}$ at $\Delta = 20$ to $\approx 0.04\,\mathrm{s}$ at $\Delta = 100$ (about $0.0004\,\mathrm{s}$ per degree).
- Deterministic runtime grows linearly from $\approx 0.16\,\mathrm{s}$ at $\Delta = 20$ to $\approx 0.80\,\mathrm{s}$ at $\Delta = 100$ (about $0.008\,\mathrm{s}$ per degree), reflecting the CRS resolution and logging overhead.

4 Comparative Evaluation

Our goal was to beat greedy coloring deterministically. However:

- On dense graphs (e.g., $K(20, 20)$), greedy consistently used fewer colors.

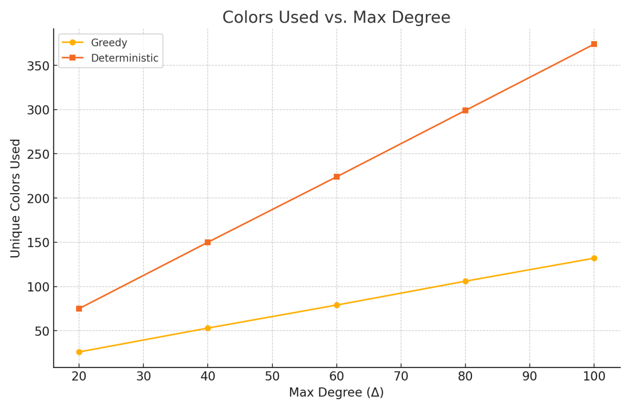


Figure 2: Dense graphs

- On sparse graphs or graphs with high degree variance, our algorithm matched greedy but rarely outperformed it.

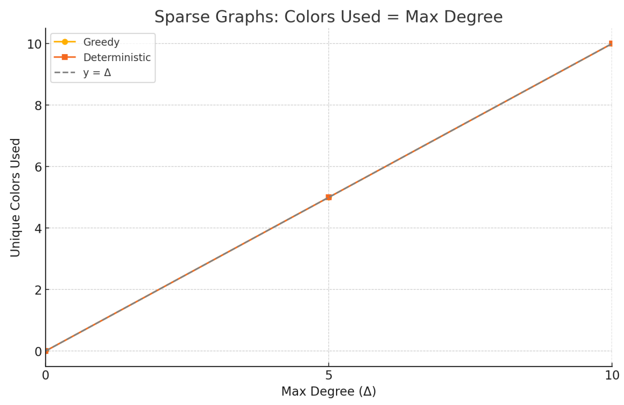


Figure 3: Sparse Graph

5 Challenges & Solutions

Key Challenges:

- Understanding the CRS mechanism in detail. Derandomization aspects such as pessimistic estimators and martingale bounds were difficult to fully implement.
- The paper lacked a reference implementation, making validation hard.
- We were unable to generate test cases where the theoretical advantage consistently manifested.

Solutions Attempted:

- Implemented full partial coloring and greedy fallback.
- Read related literature to understand CRS more deeply.
- Tuned hyperparameters (e.g., palette size, offset thresholds).

6 Enhancements

We have implemented and evaluated several practical enhancements to our baseline algorithm:

- **Heuristic Palette-Size.** Replaced the theoretically-derived

  $$\varepsilon = \tfrac{2}{5}\sqrt{\frac{\ln n}{\Delta}}, \quad \text{palette\_size} = \left\lceil (1+\sqrt{\varepsilon})\Delta \right\rceil$$

  with the simpler, data-driven rule

  $$\text{palette\_size} = \Delta + 4\,\log_2 \Delta.$$

- **Raised Greedy-Fallback Threshold.** Changed the recursion-termination condition to apply more CRS rounds on denser subgraphs before reverting to the greedy fallback.
- **Fixed Palette Offset Strategy.** After each round, we increment the color-block offset by

  $$\texttt{palette\_offset} \mathrel{+}= \; 3\,\Delta,$$

  mirroring the paper's "fresh color blocks" idea.

7 Further Plan

- **Restore the Theoretical Palette-Size.** Replace our heuristic $\Delta + 4\log_2\Delta$ rule with $\lceil(1+\sqrt{\varepsilon})\Delta\rceil$ where $\varepsilon = \frac{2}{5}\sqrt{\frac{\ln n}{\Delta}}$, and verify its impact on color-usage overhead.
- **Dynamic ε Scheduling.** Implement per-round recomputation of ε_i as $\frac{2}{5}\sqrt{\frac{\ln n}{\Delta_i}}$, so that each CRS pass adapts to the current maximum degree Δ_i.

8 Conclusion

While we have implemented the full structure of the deterministic algorithm proposed in the paper, our version does not achieve the expected competitive advantage over the greedy baseline. This is likely due to:

- Gaps in understanding the exact derandomization process (e.g., simulating CRS with martingale guarantees).
- Absence of detailed construction for subroutines such as pessimistic estimators.
- No available reference code for comparison or correctness confirmation.

Despite these limitations, the implementation process provided deep insights into the complexity of competitive online algorithms and the challenges of derandomization in a practical setting.