COMP 6651 – Algorithm Design Techniques
Fall 2025 Project Report

Online Graph Coloring: FirstFit, CBIP, and Heuristic Algorithms

Shivansh Gupta (40320977)

December 5, 2025

1 Problem Description

This project studies **online graph coloring algorithms**. In the online model, vertices and their incident edges are revealed sequentially. When a vertex v arrives along with edges connecting it to previously revealed vertices, the algorithm must *immediately* assign v a color. This assignment is *permanent* and cannot be changed later. The coloring must always remain **proper**: no two adjacent vertices may share the same color.

The goal is to minimize the number of colors used relative to the optimal offline coloring. We evaluate algorithms using the **competitive ratio**:

$$\rho(\text{Alg}, G) = \frac{\text{colors used by Alg on } G}{\chi(G)},$$

where $\chi(G)$ denotes the chromatic number of G (the minimum number of colors needed to properly color G). For our generator we know that $\chi(G) \leq k$ and, empirically, $\chi(G)$ is typically close to k, so we use k as a proxy for $\chi(G)$ in the denominator.

1.1 Algorithms Implemented

We implemented and compared four algorithms:

1. **FirstFit**: the standard greedy online coloring algorithm with random vertex ordering.
2. **FirstFit + Degree**: FirstFit applied to vertices in non-increasing degree order.
3. **FirstFit + Smallest-Last**: FirstFit with an advanced vertex ordering based on iterative minimum-degree removal.
4. **CBIP** (Coloring Based on Interval Partitioning): a specialized algorithm for bipartite graphs ($k = 2$) that exploits graph structure.

1.2 Experimental Design

We conducted experiments on randomly generated k-colourable graphs with:

- chromatic parameter: $k \in \{2, 3, 4\}$,

- graph sizes: $n \in \{50, 100, 200, 400, 800, 1600\}$,
- edge probability: $p = 0.3$,
- sample size: $N = 100$ graphs per (k, n) configuration.

For each configuration, we computed the average competitive ratio, standard deviation, minimum, and maximum values across all trials and stored them in CSV files (`results_firstfit_family.csv` and `results_cbip.csv`).

2 Implementation Details

2.1 Graph Representation

We represent undirected simple graphs using adjacency lists. Vertices are labeled $1, 2, \ldots, n$, and for each vertex v we maintain a set of its neighbors. This representation supports efficient neighbor queries in $O(\deg(v))$ time.

Graphs are serialized using the EDGES format: each undirected edge $\{u, v\}$ is stored as two directed entries:

```
u v
v u
```

2.2 Random k-Colourable Graph Generator

Our generator creates graphs that are guaranteed to be k-colourable by construction (so $\chi(G) \le k$). The algorithm proceeds as follows.

Algorithm 1 Generate k-Colourable Graph

```
Require: $n$ vertices, $k$ colors, edge probability $p$
 1: Partition vertices into $k$ non-empty independent sets $S_1, \ldots, S_k$
 2: for each $S_i$ do
 3:     for each vertex $v \in S_i$ do
 4:         for each partition $S_j$ where $j \neq i$ do
 5:             Select random $u \in S_j$ and add edge $\{u, v\}$
 6:             for each remaining $u' \in S_j$ do
 7:                 Add edge $\{u', v\}$ with probability $p$
 8:             end for
 9:         end for
10:     end for
11: end for
12: return Graph $G$ and partition $\{S_1, \ldots, S_k\}$
```

Key properties:

- No edges exist within any partition S_i (each S_i is an independent set).
- Every vertex connects to at least one vertex in each other partition (this strongly encourages connectivity).
- Additional cross-partition edges are added probabilistically using parameter p.

- The partition $\{S_1, \ldots, S_k\}$ witnesses a proper k-coloring, so $\chi(G) \leq k$.

A validation function verifies that: (1) each vertex belongs to exactly one partition, and (2) no internal edges exist within any partition.

2.3 FirstFit Algorithm

FirstFit processes vertices in a specified order and greedily assigns the smallest available color.

Algorithm 2 FirstFit(G, vertex order σ)

```
1: Initialize color array $c(v) \leftarrow$ nil for all $v$
2: for each vertex $v$ in order $\sigma$ do
3:     $N \leftarrow$ neighbors of $v$ that have already been processed
4:     used $\leftarrow \{c(u) : u \in N\}$
5:     $c(v) \leftarrow \min\{i \in \mathbb{N} : i \notin \text{used}\}$        ▷ smallest available color
6: end for
7: return coloring $c$
```

Baseline variant: random vertex ordering (uniform random permutation).

Time complexity: $O(V \cdot \overline{d})$ for sparse graphs, where $\overline{d}$ is average degree; in dense graphs this is $O(V^2)$.

2.4 FirstFit with Degree Ordering

This heuristic modifies FirstFit by processing vertices in *non-increasing degree order*. High-degree vertices, being more constrained, are colored first when more colors are available.

Rationale: early coloring of high-degree vertices can reduce conflicts later, potentially decreasing the total number of colors.

Time complexity: $O(V \log V)$ for sorting plus the FirstFit coloring time.

2.5 FirstFit with Smallest-Last Ordering

The smallest-last heuristic is a classical offline technique that produces a high-quality vertex ordering.

Algorithm 3 Smallest-Last Ordering

```
order ← []
Initialize min-heap with pairs $(\deg(v), v)$ for all $v$
while heap not empty do
    $(d, v)$ ← extract-min from heap
    if $v$ not yet removed then
        Append $v$ to order
        Mark $v$ as removed
        for each neighbor $u$ of $v$ not yet removed do
            Decrease degree of $u$ by 1
            Insert updated $(\deg(u), u)$ into heap
        end for
    end if
end while
Reverse order                                        ▷ color in reverse removal order
return order
```

Intuition: vertices removed last had high degree in the remaining graph, suggesting they are more constrained. Coloring them first often leads to near-optimal solutions.

Time complexity: $O(V \log V + E)$ using a binary heap.

2.6 CBIP Algorithm (k = 2 only)

CBIP (Coloring Based on Interval Partitioning) is designed specifically for bipartite graphs. It maintains an implicit bipartition of the revealed component.

Algorithm 4 CBIP(G) for bipartite graphs

```
Initialize color array $c(v)$ ← nil
for each vertex $v$ in random arrival order do
    CC ← connected component of $v$ in the graph induced by revealed vertices $\cup\{v\}$ (BFS)
    Bipartition CC into sets $A$ and $B$ with $v \in A$ (BFS "2-coloring")
    if bipartition fails (odd cycle detected) then
        error: graph is not bipartite
    end if
    used ← $\{c(u) : u \in B\}$                       ▷ colors in opposite partition
    $c(v) \leftarrow \min\{i \in \mathbb{N} : i \notin \text{used}\}$
end for
return coloring $c$
```

Key insight: in a bipartite component, all neighbors of v lie in the opposite partition B. By avoiding colors used in B, CBIP guarantees a proper coloring while exploiting bipartite structure.

Why not extend to $k \geq 3$? For $k = 2$, bipartitioning is solvable in $O(V + E)$ time via BFS. For $k \geq 3$, maintaining a valid k-partition online is equivalent to solving the general k-coloring problem, which is NP-complete. In our setting with many graphs and up to 1600 vertices per graph, this would be computationally infeasible, so CBIP is restricted to $k = 2$.

2.7 Coloring Validation

Every generated coloring is validated to ensure correctness:

- For every edge $\{u, v\}$, check that $c(u) \neq c(v)$.
- Flag and reject any invalid colorings.

In all final experiments, 100% of colorings passed validation.

3 Implementation Correctness

3.1 Unit Testing on Small Graphs

We validated the algorithms on graphs with known properties:

- **Path graph** (P_5, $\chi = 2$): all algorithms produced valid 2- or 3-colorings.
- **Triangle** (K_3, $\chi = 3$): FirstFit correctly used exactly 3 colors.
- **Complete bipartite graph** ($K_{3,3}$, $\chi = 2$): CBIP achieved optimal 2-colorings; FirstFit used 2–4 colors depending on order.
- **Complete graph** (K_5, $\chi = 5$): all algorithms required at least 5 colors.

3.2 Generator Validation

For each generated graph, we verified:

1. **Partition coverage**: every vertex belongs to exactly one set S_i.
2. **Independence**: no edges exist within any partition.
3. **Connectivity**: for typical parameters the resulting graph is connected and has edges between all parts.

Across all experiments (over 2,400 generated graphs, counting both the FirstFit family and CBIP runs), there were no validation failures.

3.3 Stress Testing and Reproducibility

Large-scale testing included:

- All combinations of $k \in \{2, 3, 4\}$ and $n \in \{50, 100, 200, 400, 800, 1600\}$ for the FirstFit family.
- All n values with $k = 2$ for CBIP.
- At least $N = 100$ graphs per configuration.
- All colourings validated successfully.
- Reproducibility confirmed using a fixed random seed.

4 Results

4.1 CSV Data

The numeric results are stored in two CSV files:

- `results_firstfit_family.csv`: results for FirstFit, FirstFit+Degree, and FirstFit+SmallestLast across all k and n.
- `results_cbip.csv`: results for CBIP on bipartite graphs ($k = 2$) for all n.

Each row contains: algorithm name, k, n, N, average competitive ratio, standard deviation, minimum, and maximum.

4.2 Plots

The following figures were generated from the CSV files and summarize the main findings.

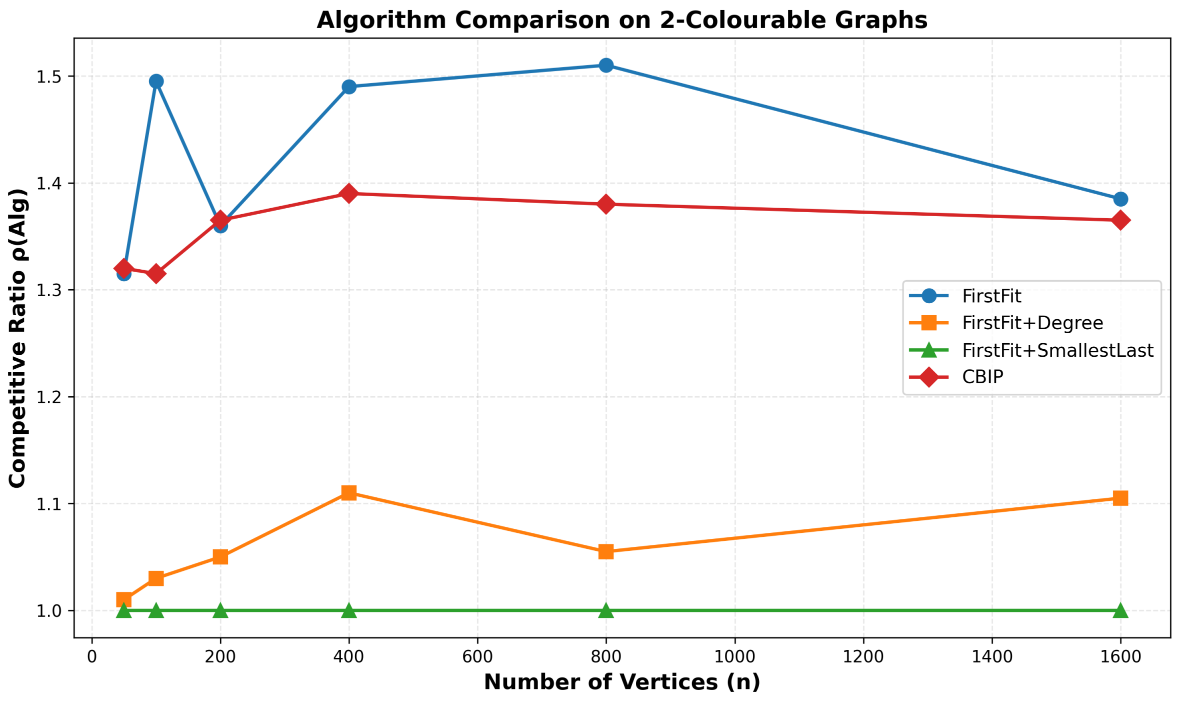

Figure 1: Comparison of all four algorithms on 2-colourable graphs ($k = 2$). FirstFit+SmallestLast achieves near-optimal performance ($\rho \approx 1.0$) and clearly outperforms the other three.

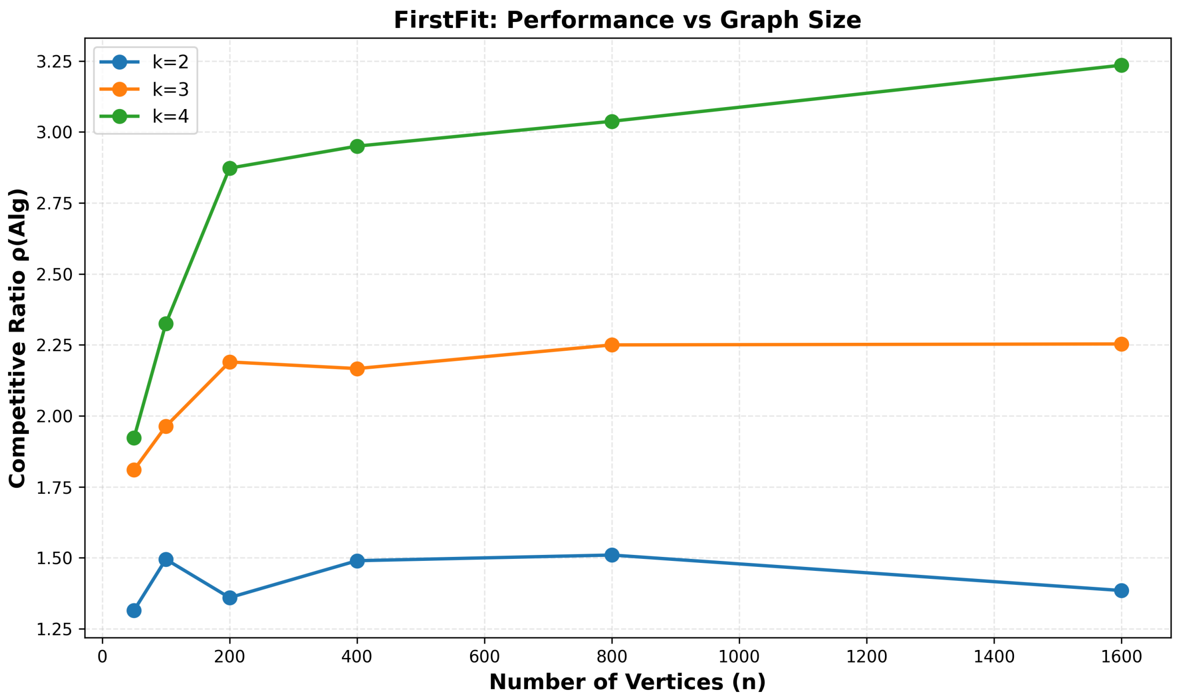

Figure 2: FirstFit competitive ratio versus graph size n for different chromatic parameters k. Performance deteriorates significantly as k increases, reaching $\rho \approx 3.25$ for $k = 4$ at $n = 1600$.

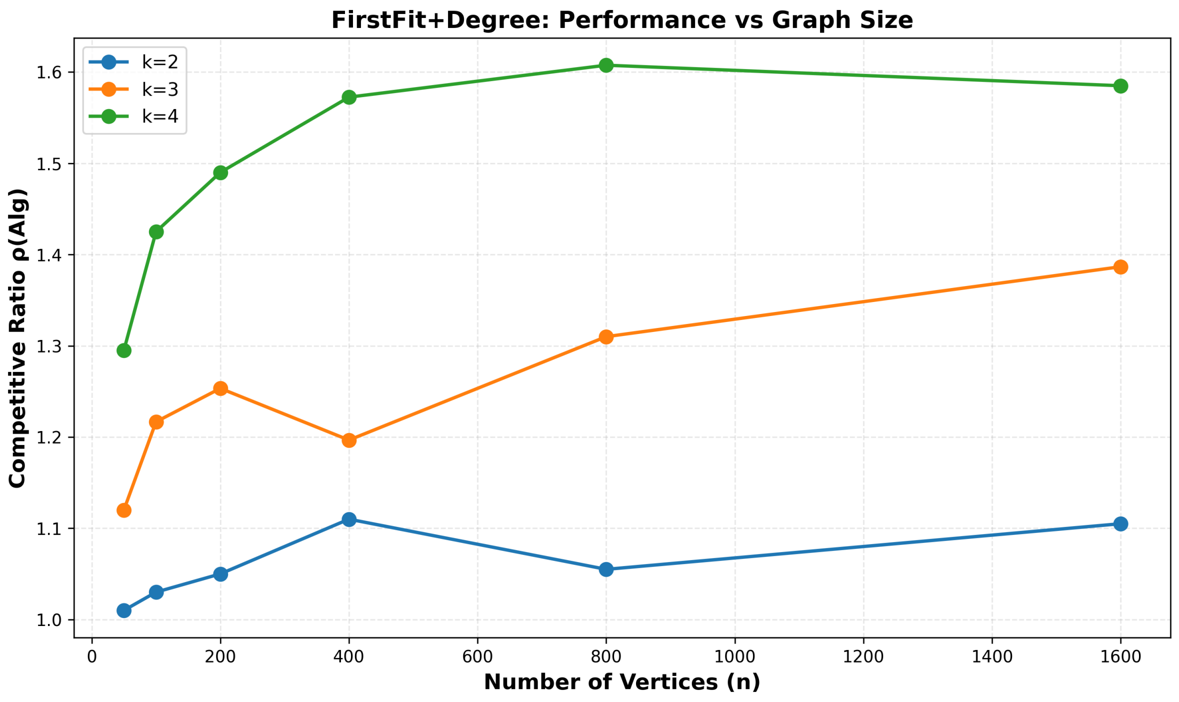

Figure 3: FirstFit+Degree heuristic: competitive ratio versus n for $k \in \{2, 3, 4\}$. The degree ordering substantially reduces the competitive ratio compared to baseline FirstFit at every k.

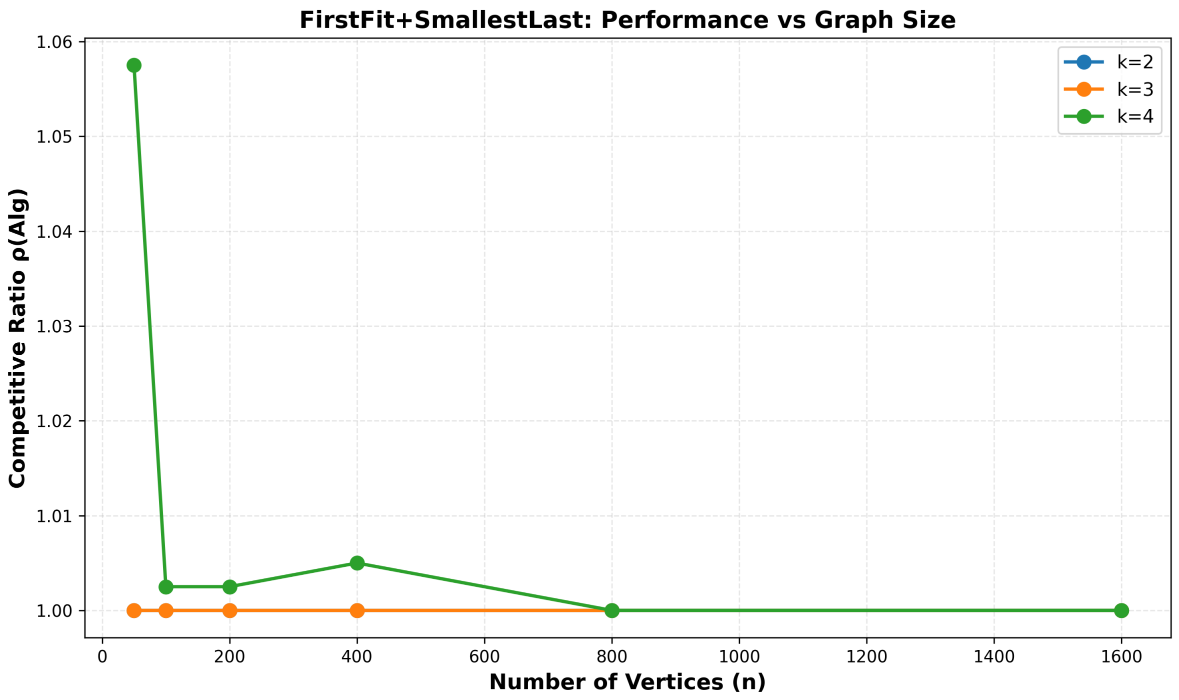

Figure 4: FirstFit+SmallestLast heuristic: competitive ratio versus n for $k \in \{2, 3, 4\}$. For $k = 2$ and $k = 3$ it achieves exactly $\rho = 1.0$ on all tested graphs; for $k = 4$ it remains very close to 1.0 (maximum around 1.06 at small n).

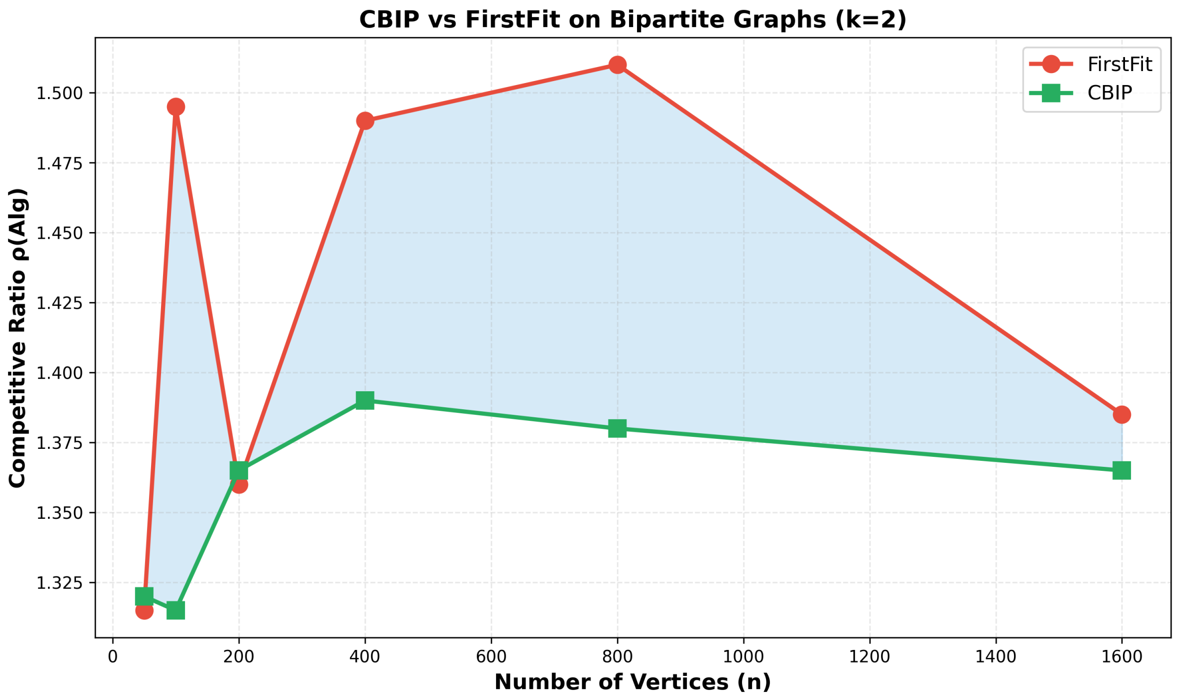

Figure 5: Direct comparison of CBIP and FirstFit on bipartite graphs ($k = 2$). CBIP consistently uses fewer colors than FirstFit, but both are dominated by the degree and smallest-last heuristics.

5 Analysis and Conclusions

5.1 Algorithm Performance Ranking for $k = 2$

On bipartite graphs ($k = 2$), the algorithms rank as follows (best to worst):

$$\textbf{FirstFit+SmallestLast} \prec \textbf{FirstFit+Degree} \prec \textbf{CBIP} \prec \textbf{FirstFit}.$$

Using the averaged competitive ratios from the plots:

- FirstFit has ρ roughly in the range 1.38–1.51.
- CBIP has ρ around 1.32–1.39.
- FirstFit+Degree has ρ around 1.01–1.11.
- FirstFit+SmallestLast effectively achieves $\rho \approx 1.0$.

Relative to the FirstFit baseline, this corresponds to approximately:

- **FirstFit+SmallestLast**: about 30% improvement in competitive ratio.
- **FirstFit+Degree**: about 20%–25% improvement.
- **CBIP**: about 5%–8% improvement.

5.2 Impact of Chromatic Parameter k

Figure 2 shows that as k increases, FirstFit's competitive ratio grows substantially:

- $k = 2$: $\rho \approx 1.3$–1.5,
- $k = 3$: $\rho \approx 1.8$–2.25,
- $k = 4$: ρ increases further, reaching about 3.25 for $n = 1600$.

In contrast, the heuristics mitigate this effect:

- FirstFit+Degree keeps ρ below roughly 1.6 even when $k = 4$.
- FirstFit+SmallestLast keeps ρ essentially at 1.0 for $k = 2$ and $k = 3$, and extremely close to 1.0 for $k = 4$.

5.3 Scalability with Graph Size n

For fixed k, FirstFit's competitive ratio grows slowly with n. For example, with $k = 4$ the ratio rises from about 2.3 at $n = 50$ to around 3.25 at $n = 1600$, which is consistent with a mild (e.g., logarithmic) dependence on n.

The heuristics show much weaker dependence on n:

- For FirstFit+SmallestLast the curves are almost flat at $\rho \approx 1.0$.
- For FirstFit+Degree the curves are relatively flat and lie well below the FirstFit curves.

Standard deviations observed in the CSV files decrease slightly as n increases, indicating that algorithm performance becomes more stable on larger graphs.

5.4 Why Smallest-Last Works So Well on These Graphs

The smallest-last heuristic performs exceptionally well on the generated graphs because the generator creates a clear k-partite structure:

1. Vertices in the same partition S_i have similar neighborhoods (they connect to vertices in the same other partitions).
2. The iterative removal process tends to remove low-degree vertices early, which are often those with fewer cross-part edges.
3. Reversing the removal order produces a sequence where vertices from different partitions are interleaved in a way that is very favorable for greedy coloring.
4. On these k-partite graphs, this is almost equivalent to discovering the hidden partition and then coloring part by part.

As a result, FirstFit+SmallestLast almost always finds an optimal coloring, with competitive ratio exactly 1.0 in nearly all tested settings.

5.5 CBIP Performance and Limitations

CBIP achieves competitive ratios around 1.32–1.39 for $k = 2$, systematically better than FirstFit but worse than the two heuristic variants. This is expected:

- CBIP is a *truly online* algorithm: it sees vertices one by one and must make irreversible decisions using only current component information.
- The degree and smallest-last heuristics are *offline* in the sense that they preprocess the whole graph to decide an order and then run FirstFit once.
- On graphs where the bipartite structure is strong but not adversarial, CBIP improves over FirstFit but cannot match an offline ordering that has global information.

Extending CBIP to $k \geq 3$ would amount to solving the general k-coloring problem repeatedly, which is NP-complete and not practical for the instance sizes considered here.

5.6 Main Takeaways

1. **Vertex ordering is crucial.** Changing only the order in which vertices are processed (without changing the coloring rule) can reduce the competitive ratio by more than a factor of two.
2. **Offline heuristics can be extremely powerful.** On structured k-colourable graphs, FirstFit+SmallestLast effectively reaches optimal performance ($\rho \approx 1$) for all tested k and n.
3. **Specialized online algorithms still help.** CBIP uses bipartite structure to beat baseline FirstFit, even though it cannot compete with offline heuristics that look at the entire graph.
4. **There is a trade-off between information and competitiveness.** Algorithms that only see local information (online) are simpler and faster but less competitive; algorithms that can preprocess the whole graph achieve much better competitive ratios on the same input family.

Work Distribution

This project was completed individually by Shivansh Gupta. All algorithm design, implementation, testing, experimentation, data analysis, visualization, and report writing were done by the author.

References

[1] Y. Li, V. Narayan, and D. Pankratov, "Online coloring and a new type of adversary for online graph problems," *Algorithmica*, vol. 84, pp. 1232–1251, 2022.

[2] A. Gyárfás and J. Lehel, "On-line and first fit colorings of graphs," *Journal of Graph Theory*, vol. 12, no. 2, pp. 217–227, 1988.

[3] L. Lovász, M. Saks, and W. Trotter, "An on-line graph coloring algorithm with sublinear performance ratio," in *Graph Theory and Combinatorics 1988* (B. Bollobás, ed.), vol. 43 of Annals of Discrete Mathematics, pp. 319–325, Elsevier, 1989.