UNIVERSITY OF LJUBLJANA
FACULTY OF MATHEMATICS AND PHYSICS

Financial mathematics

Jon Pascal Miklavčič, Nik Živkovič Kokalj

Packing Coloring of Subcubic Planar Graphs

Term Paper in Finance Lab

Long Presentation

Advisers:
Assist. Prof. Dr. Janoš Vidali,
Prof. Dr. Riste Škrekovski

Ljubljana, 2025

1. Introduction

The *packing coloring problem* involves assigning colors to vertices of a graph such that any two vertices of color i are at distance greater than i. The minimal number of colors required is called the *packing chromatic number* (PCN). This problem is NP-hard, as determining the PCN is equivalent to solving an integer linear program. For subcubic planar graphs, the computational complexity remains an obsticle, requiring both exhaustive and heuristic approaches. In this report we will anlyze two distinct approaches we used for estimating PCN: a complete search and a randomized local search.

2. Complete Search Algorithm

2.1. **Methodology.** The `complete_search` function systematically evaluates all connected subcubic planar graphs with up to m vertices. It leverages `nauty_geng` to generate these graphs with parameters `-c -D3`, enforcing connectivity and keeping the degree of all vertices subcubic. Planarity of the graph is checked separately. For each graph G that fits these requirements, the PCN is computed with an ILP-based `packing_coloring` function. The details of this function are described in the short presentation. The algorithm then tracks graphs with the highest PCN encountered and saves progress periodically to handle interruptions.

2.2. **Strengths and Limitations.** This approach guarantees identification of the maximal PCN within the specified vertex range. However, its time complexity scales exceptionally quickly as we increase the number of vertices n, as the number of subcubic planar graphs grows exponentially. The reliance on ILP for each PCN computation further compounds this inefficiency, as even individual graph evaluations are computationally intensive. Consequently, it is not sensible to use this function from graphs with much more than 12 vertices.

3. Randomized Local Search Algorithm

3.1. **Methodology.** The `random_search` function employs a stochastic strategy to explore the PCN of graphs without an exhaustive search. Starting from an initial subcubic, planar and connected graph on n vertices, that is generated by the `initialize_base_graph` function, it iteratively applies a modification function `modify_planar_subcubic_graph` to generate neighboring graphs. Each modified graph is uniquely labeld with its graph6 string to improve efficiency. The PCN of new graphs is evaluated, and the function keeps track of all the graphs with high PCN. Progress is saved periodically to handle interruptions. The modification function is detailed in the next section.

3.2. **Strengths and Limitations.** This method circumvents some of the limitations od the exhaustive search method by focusing on a localized search within the graph space. While it cannot guarantee global optimality, it efficiently explores graphs with larger vertex counts than feasible for the complete search. The primary constraint remains the ILP-based PCN computation, which limits the number of iterations.

4. Graph Modification

Function: `removable_vertices`. The function `removable_vertices(G)` identifies vertices in a given graph G that can be removed while maintaining the connectivity of the graph.

We firstly initialize an empty list to store removable vertices. The function then iterates through each vertex v in the vertex set of G. Fore each iteration a copy H of G is created to prevent modifying the original graph. After that vertex v is removed from a copy. If H remains connected after the removal of v, then v is appended to the `removable` list. At the end the function returns the list of removable vertices.

Function: `modify_planar_subcubic_graph`. The function `modify_planar_subcubic_graph(G)` modifies a planar subcubic graph G into a new subcubic planar graph while ensuring that the total number of vertices remains constant.

(1) A copy of G is created to avoid altering the original input.
(2) The function verifies that G is both planar and subcubic. If not, function raises an error.
(3) The set of faces in G is retrieved. If no faces exist, the function returns G unchanged.
(4) A random face is selected, and its vertices are extracted into a list 'face_vertices'.
(5) A random number (between 1 and 3) of edges in the face is selected for subdivision. Subdivision is the insertion of a new vertex in the middle of an exiting edge, which keeps graph subcubic and planar.
    - A list of edges within the face is created. If no such edges exist, the subdivision step is skipped.
    - A random edge (a, b) is chosen.
    - A new vertex is introduced between a and b and connected to each one of them, while edge (a, b)
    - The function `removable_vertices` is called to check if any vertex can be removed.
    - If a removable vertex exists, a random one is deleted.
    - If no removable vertex is found, the newly inserted vertex is deleted, and the original edge (a, b) is restored since we want to maintain the number of vertices in the input graph the same as in the modified graph.
(6) A new vertex is introduced:
    - The set of eligible vertices (those of degree at most 2) within the selected face is identified. These vertices are eligible for a new vertex to be connected to them. We only choose vertices from a selected face because we do not want to compromise planarity.
    - A new vertex is added to G.
    - The new vertex is connected to a subset of eligible vertices with one, two or three edges.
    - The function `removable_vertices` is used again to check for possible vertex removals.
    - If a removable vertex exists, a random one is deleted.
    - If no removable vertex is found, the newly added vertex is deleted.
(7) The modified graph G is returned.

This approach ensures that the graph remains planar and subcubic while making modifications that preserve its connectivity.

5. Conclusions

Number of Vertices	Samples	Mean Running Time (s)
5	100	0.00967
6	100	0.02545
7	100	0.05766
8	100	0.10046
9	100	0.16859
10	100	0.24124
11	100	0.36945
12	100	0.52123
13	100	0.78666
14	100	1.17038
15	100	1.82392
16	100	2.70610
17	100	3.78613
18	100	5.70335
19	100	7.23080
20	100	10.11315

Table 1. Mean running time of the `packing_coloring` function by number of vertices

The table illustrates the computational challenges posed by solving this problem using ILP-s. The increasing mean running time of the coloring function highlights two key limitations:

(1) **Increasing number of graphs:** As the number of vertices increases, the number of possible subcubic planar graphs grows quickly. A complete search, which evaluates all possible graphs, quickly becomes infeasible.
(2) **Increasing complexity of ILPs:** Even if the number of graphs didn't increase, computing the PCN remains NP, with running time increasing significantly as graph size grows.

Here we use random search as a feasible alternative. Both algorithms tackle the PCN problem under NP constraints. Complete search provides exact maximal value for small graphs but is impractical for larger ones. Randomized search sacrifices completeness for scalability, allowing exploration of larger graphs at the cost of optimality guarantees.

6. Experimental Framework

6.1. **Hardware and Methodological Constraints.** The investigation analyzed connected subcubic planar graphs with 5-20 vertices. Due to hardware limitations, the complete search algorithm was restricted to graphs with up to 11 vertices. The highest PCN identified was 6, achieved on a number of 11-vertex graph (stored in `Results/complete_search_1-11_vertices.pkl`).

6.2. **Sampling.** For $n \leq 11$, `complete_search` evaluated all graphs; for $n > 11$, `random_search` with 10,000 iterations was used.

7. Conclusion

This study empirically identifies structural properties correlated with higher PCNs in subcubic planar graphs. Hardware constraints limited exhaustive verification, but heuristic evidence suggests PCN ≤ 7 for $n \leq 20$.

- **Elongated paths, cycle-based structures:** Graphs that achieve high PCN typically feature elongated paths, cycle-based structures, and relatively sparse local connectivity. These properties contribute to greater distances between identically colored vertices, increasing the overall PCN. More cycles means more connections between nodes, which limits the coloring possibilities. If a graph has many cycles, then there are densely connected subgraphs that require more colors because identically colored nodes must be far enough apart. If a graph contains many short cycles, then it is more difficult to use the same colors for nodes in the graph, as the coloring constraints spread faster.
- **Higher degree of vertices:** If there were many nodes of degree 1 or 2, the graph would contain many tree structures, which lower the packing coloring number. Nodes with degree 3, however, ensure that the graph remains sufficiently dense and has many cycles. Almost all nodes have degree 3 because this is the best option for creating dense cyclic structures that raise the packing coloring number while keeping the graph in the subcubic and planar category.

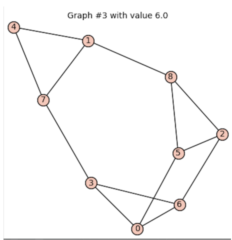

Figure 1. Graph with PCN 6 on 11 vertices

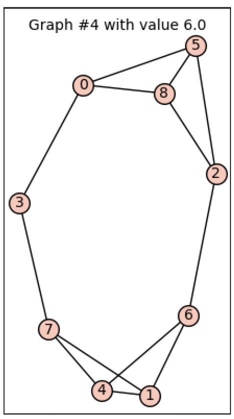

Figure 2. Graph with PCN 6 on 11 vertices

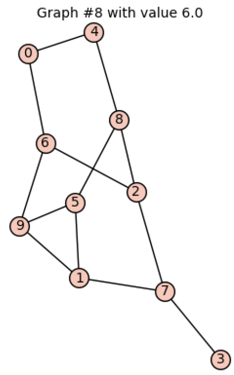

FIGURE 3. Graph with PCN 6 on 11 vertices

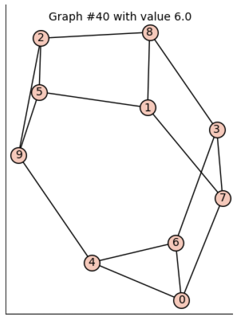

FIGURE 4. Graph with PCN 6 on 11 vertices

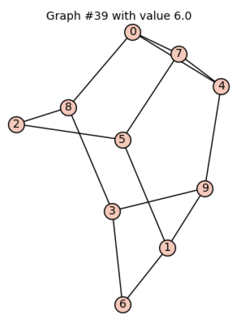

FIGURE 5. Graph with PCN 6 on 11 vertices

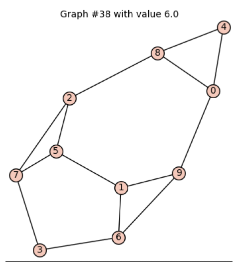

FIGURE 6. Graph with PCN 6 on 11 vertices

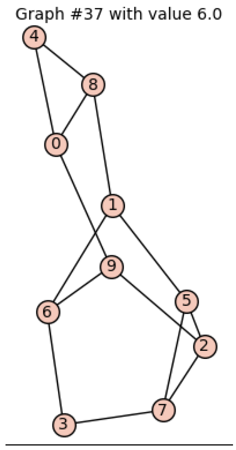

Figure 7. Graph with PCN 6 on 11 vertices

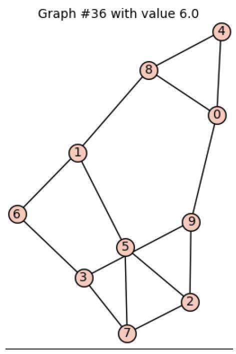

Figure 8. Graph with PCN 6 on 11 vertices

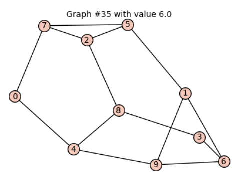

Figure 9. Graph with PCN 6 on 11 vertices

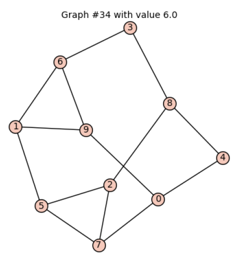

Figure 10. Graph with PCN 6 on 11 vertices

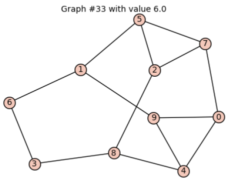

FIGURE 11. Graph with PCN 6 on 11 vertices

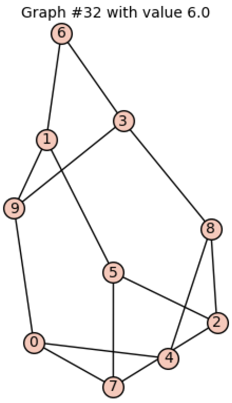

FIGURE 12. Graph with PCN 6 on 11 vertices

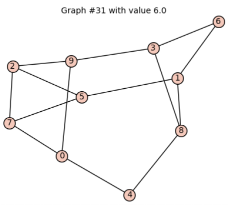

FIGURE 13. Graph with PCN 6 on 11 vertices

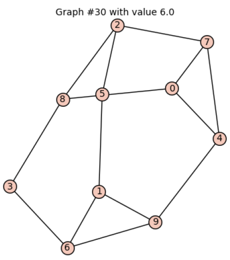

FIGURE 14. Graph with PCN 6 on 11 vertices

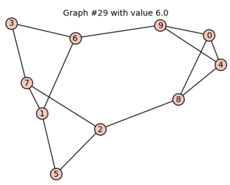

FIGURE 15. Graph with PCN 6 on 11 vertices

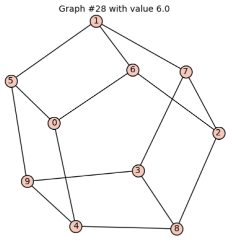

FIGURE 16. Graph with PCN 6 on 11 vertices

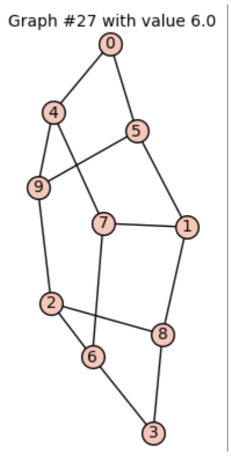

FIGURE 17. Graph with PCN 6 on 11 vertices

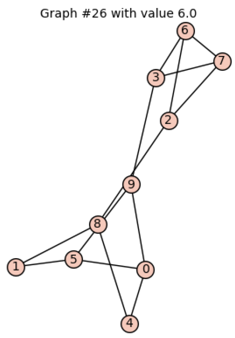

FIGURE 18. Graph with PCN 6 on 11 vertices

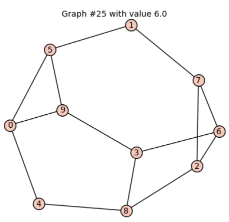

FIGURE 19. Graph with PCN 6 on 11 vertices

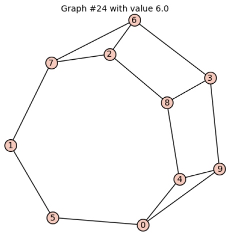

FIGURE 20. Graph with PCN 6 on 11 vertices

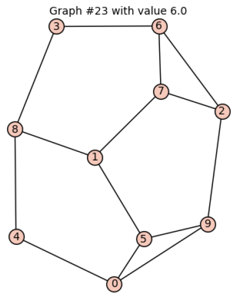

FIGURE 21. Graph with PCN 6 on 11 vertices

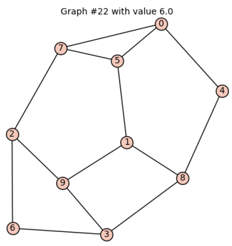

FIGURE 22. Graph with PCN 6 on 11 vertices

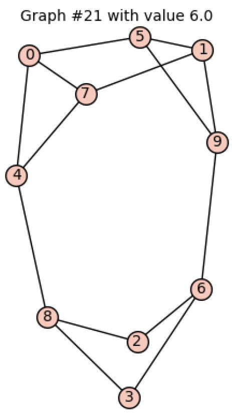

FIGURE 23. Graph with PCN 6 on 11 vertices

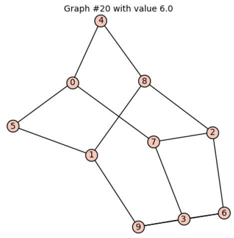

FIGURE 24. Graph with PCN 6 on 11 vertices

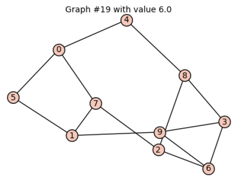

FIGURE 25. Graph with PCN 6 on 11 vertices

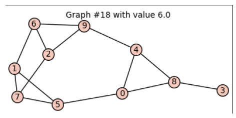

FIGURE 26. Graph with PCN 6 on 11 vertices

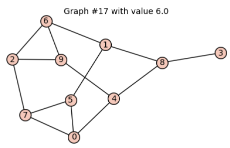

FIGURE 27. Graph with PCN 6 on 11 vertices

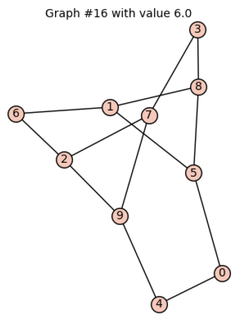

FIGURE 28. Graph with PCN 6 on 11 vertices

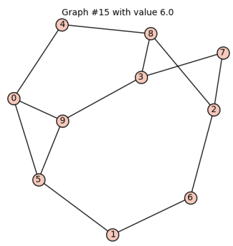

FIGURE 29. Graph with PCN 6 on 11 vertices

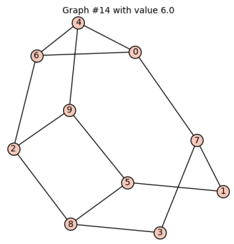

Figure 30. Graph with PCN 6 on 11 vertices

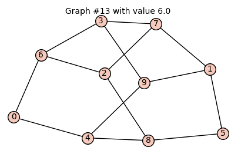

Figure 31. Graph with PCN 6 on 11 vertices

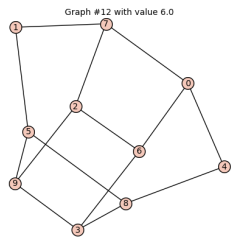

Figure 32. Graph with PCN 6 on 11 vertices

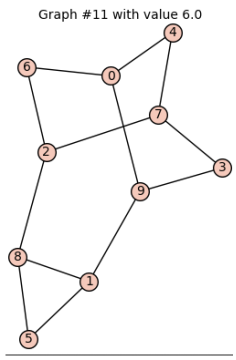

Figure 33. Graph with PCN 6 on 11 vertices

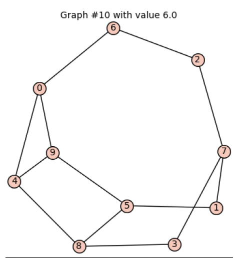

Figure 34. Graph with PCN 6 on 11 vertices

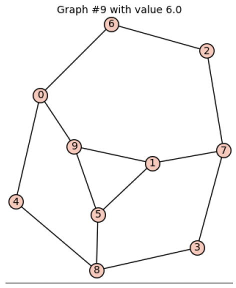

FIGURE 35. Graph with PCN 6 on 11 vertices

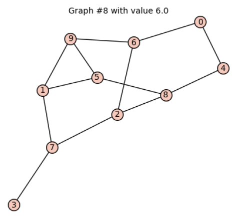

FIGURE 36. Graph with PCN 6 on 11 vertices

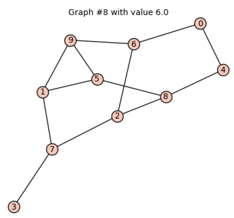

FIGURE 37. Graph with PCN 6 on 11 vertices

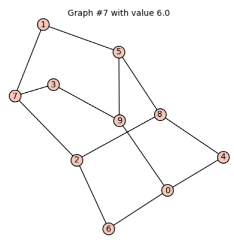

FIGURE 38. Graph with PCN 6 on 11 vertices

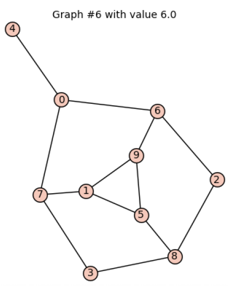

FIGURE 39. Graph with PCN 6 on 11 vertices

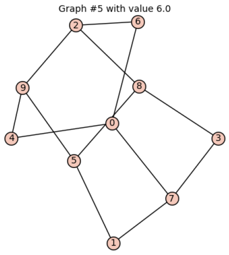

FIGURE 40. Graph with PCN 6 on 11 vertices

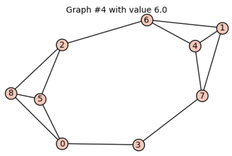

FIGURE 41. Graph with PCN 6 on 11 vertices

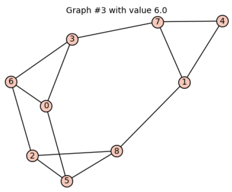

FIGURE 42. Graph with PCN 6 on 11 vertices

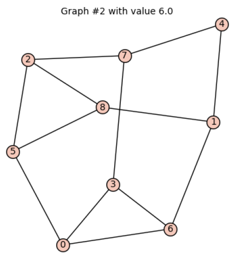

FIGURE 43. Graph with PCN 6 on 11 vertices

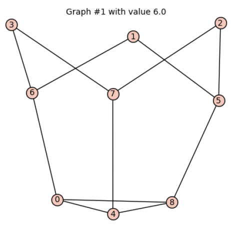

FIGURE 44. Graph with PCN 6 on 11 vertices