

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, 2024

To obtain the best possible packing chromatic number, we needed to generate as many different subcubic planar graphs as possible. To achieve this, we made a function named `modify_planar_subcubic_graph(G)` which takes subcubic planar graph  $G$  as an input and returns different subcubic planar graph.

Description of a function `modify_planar_subcubic_graph(G)`:

Firstly, we ensure that the entered graph is subcubic and planar. If not, the function returns an error. We also added a condition that the graph must be connected. This is because, for the packing coloring number, if the graph is composed of multiple disconnected parts, the packing coloring number of the graph will equal the packing coloring number of the part that has the biggest packing coloring number. This ensures that the function is not as time-consuming as it would otherwise be. Now, we are certain that we are operating on a subcubic and planar graph. We then defined three different operations on the graph. In each iteration, one of them is executed. Which operation will be chosen is determined randomly using the function `choice` from the random library.

We named the first operation `add_vertex`. A new vertex is generated and added to the graph. We refer to the added vertex as  $e$ . Again, using the function `choice`, the algorithm selects a random edge in the given graph, for instance, the edge  $(u, v)$ . It then generates two new edges,  $(e, u)$  and  $(e, v)$ , and deletes the existing edge  $(u, v)$ . Since we are essentially adding a vertex on to an existing edge, planarity is preserved. The graph remains subcubic because the vertex degrees remain unchanged. The graph also remains connected.

The second operation is `rewire_edge`. The algorithm selects a random edge in the graph  $G$  and checks if the graph, without the chosen edge, remains connected. If so, the edge is removed. This process is repeated until two edges are removed or until there are no appropriate edges left to remove. This means that if no edges can be removed, or only one can be removed, then none or only one edge will be removed. After this step, two new edges are randomly added. This is done by sampling two random vertices and generating a new edge between them. If the resulting graph meets the requirements of being planar and subcubic, the new edge is added to the graph. Since adding edges cannot destroy the connectivity of a graph, we do not need to check for connectivity.

The third and final operation is `face`. The algorithm chooses a random face of the entered graph  $G$ . If there are no faces to choose from, the function returns an error. Selecting a face ensures that the graph remains planar, subcubic, and connected. After the face is chosen, we label this face as a new graph  $f$ . We then add a new vertex to  $f$  and refer to this vertex as  $v$ . We want to connect vertex  $v$  to the graph  $f$ . This is done by adding edges from  $v$  to vertices in  $f$ . Firstly, we check how many eligible vertices exist in  $f$  that can be connected to  $v$ .

We consider the following cases:

- (1) There are less than two eligible vertices in  $f$  that can be connected to  $v$ :  
If there is one or zero eligible vertices, we proceed as in the operation `add_vertex` by placing vertex  $v$  on an existing edge. Since only one edge

is removed and two are added, we search for another edge to remove while maintaining the connectivity of the graph. If no such edge exists, only one edge will be removed, and two will be added.

- (2) There are two or more eligible vertices in  $f$  that can be connected to  $v$ :  
 In this case, using the function `randint`, we randomly select two or three eligible vertices to connect to  $v$ . If there are only two eligible vertices, there is no need for random selection. After selecting the vertices, we connect them to  $v$ . Since we operate on a face of the graph, planarity is preserved. Because we added a vertex and edges, we also aim to remove as many edges as we added. This is achieved by choosing random edges to remove while maintaining connectivity.

Simply choosing a face reduces the graph's complexity, serving as a kind of reset.