# Empirical study of an algorithm for finding Hamiltonian cycles in 3-regular planar graphs.

Kristoffer Højelse

February 2024

**Abstract**

An algorithm in the making considers the problem of finding Hamiltonian cycles in 3-regular planar graphs.

## 1 Description of the problem

The Hamiltonian Cycle Problem is a computational problem that comes in a few variants, with slightly different input and output types.

The Hamiltonian Cycle Decision problem asks given a simple graph, to output; yes or no to the question; Does there exist a Hamiltonian Cycle in the graph? Variants might ask for an example/witness/certificate of one such cycle, to make verifying the result easy.

The Hamiltonian Cycle Counting problem asks given a simple graph, to output; the number of unique Hamiltonian cycles. Solving this problem you get the decision variant for free.

**Definition 1.1.** *The Hamiltonian Cycle Counting Problem*

*Input: Given a simple undirected connected graph $G$.*

*Output: The number of unique Hamiltonian cycles.*

*A pair of Hamiltonian cycles are not unique if and only if you can obtain the other by circular permutations and inversions of the associated vertex sequences.*

The simplest algorithm I can come up with that solves this problem has a factorial time complexity $\Theta(N!)$.

Enumerate all $N!$ permutations of the $N$ vertices and count how many of those vertex sequences are valid Hamiltonian cycles, by checking firstly that the foremost and final vertex of the sequence are adjacent and secondly that every consecutive pair of vertices in the sequence are adjacent. Then return this count not before dividing by a factor of $2N$, as there are $N$ circular permutations and 2 inversions of a sequence of $N$ elements.

A Python program of this algorithm is in the appendix 4.1.

You cannot do much better**??** for this general problem. But if we constrain the input we can do much better.

## 2 The domain

The algorithm described in this paper solves The Hamiltonian Cycle Counting Problem with a very contained subset of all graphs; simple undirected connected planar cubic graphs.

Some informal definitions of the aforementioned properties of graphs:

1. A graph is called **simple** if and only if it has no parallel edges and no self-loops.
2. A graph is called **undirected** if and only if all its edges can be traversed in both directions.
3. A graph is called **connected** (or 1-vertex-connected) if and only if there exists a path between any two vertices.
4. A graph is called **planar** if and only if there's a way to draw the graph in 2 dimensions such that no pair of edges crosses.
5. A graph is called **cubic** if and only if every vertex has exactly three neighbors.

Most variants of The Hamiltonian Cycle Problem too require the input graph to be simple, undirected, and connected. These problems probably have no known polynomial-time solutions.

The algorithm exploits the two remaining attributes, planar and cubic, to obtain a polynomial running time.

## 3 Description of the algorithm

Overview:

1. Count Hamiltonian Cycles by Dynamic Programming over a minimal Branch Decomposition.
2. Compute a minimal Branch Decomposition from a minimal Carving Decomposition of the Medial graph.
3. Compute a minimal Carving Decomposition by repeatedly Contracting edges that doesn't increase the Carving Width.
4. Compute the Carving Width with the Rat Catching algorithm which needs to compute a dual graph.
5. Compute a dual graph.

### 3.1 Dual graph

A subroutine of the algorithm computes the dual graph of a graph.

**Definition 3.1.** *The dual graph $G*$ of a planar graph $G$ is a graph with a vertex $f*$ for each face $f$ of $G$ and an edge $e*$ for each edge $e$ that separates a face $f_1$ of $G$ and a face $f_2$ of $G$.*

**Corollary 3.2.** *If multiple edges separate $f_1$ and $f_2$ there will be multiple edges between $f_1*$ and $f_2*$.*

**Corollary 3.3.** *If $e$ separates $f_1$ and $f_2$ and are the same face, $e*$ will be a self-loop.*

For the algorithm in this paper, the class of graphs that will be given as input is medial graphs of simple undirected connected planar cubic graphs.

I therefore claim, for now without any proof or argument, that;

**Claim 3.4.** *Corollary 3.3 will be irrelevant for any implementation of the algorithm.*

### 3.2 Medial graph

A subroutine of the algorithm computes the medial graph of a graph from the class; simple connected planar cubic graphs.

**Definition 3.5.** *The medial graph $M(G)$ of a connected plane graph $G$ with a vertex $e*$ for each edge $e$ of $G$ and for each face $f$ of $G$, there's an edge $c*$ between a pair of vertices $e_1*$, $e_2*$ of $M(G)$ if $e_1$ and $e_2$ are consecutive in $f$.*

todo: argue that simple connected planar cubic graphs are a valid substitution for a connected plane.

todo: identify a more succinct description of the class; medial of simple connected planar cubic graphs

## 4 Appendix.

### 4.1 Count Hamiltonian Cycles with Brute Force

count_hamcyc_brute_force.py

```python
import itertools
from parse_graph import parse_text_to_adj

G: dict[int, list[int]] = parse_text_to_adj()
vertex_set = G.keys()
N = len(vertex_set)

def valid(cycle: list[int]) -> bool:
        for i in range(0, N-1):
                if not cycle[i+1] in G[cycle[i]]:
                        return False
        if not cycle[0] in G[cycle[-1]]:
                return False
        return True

def count_ham_cyc() -> int:
        cycles = itertools.permutations(vertex_set)
        count = 0
        for cycle in cycles:
                if valid(cycle):
                        count += 1
        return count//(2*N)

print(count_ham_cyc())
```