Supplementary Verification Files for "On extremal 2-connected graphs avoiding $(0 \mod 4)$-cycles"

Project Overview and Usage Guide

Hojin Chu, Boram Park, and Homoon Ryu

July 6, 2025

1. Overview

This repository provides verification code for Proposition 2.3 of the paper "On extremal 2-connected graphs avoiding $(0 \mod 4)$-cycles" by the same authors. The code systematically generates candidate graphs and filters them according to structural constraints specified in the proposition:

- Absence of 4-cycles (exact)
- Absence of certain $(0, n-1)$-paths of lengths $\equiv l_1, l_2 \pmod 4$
- Minimum degree constraints

The resulting graphs are then deduplicated up to isomorphism, and optionally removes graphs containing 8-cycles. This tool is designed to support exploratory or extremal graph theory research.

Let $n \geq 3$, and let $(H; x, y)$ be an n-vertex graph without a $(0 \bmod 4)$-cycle such that every edge in H is contained in an (x, y)-path. If there is no (x, y)-path of length $\equiv \ell \pmod 4$ for some $\ell \in L$, then H is bipartite, and hence $e(H) \leq \frac{3n-6}{2}$ by Lemma2.5 (iii). Therefore, when $L = \ell_1, \ell_2$, we assume that H contains both an (x, y)-path of length $\equiv \ell_1 \pmod 4$ and one of length $\equiv \ell_2 \pmod 4$. To verify Proposition 2.3, we check that there is no such graph H satisfying:

$$e(H) \geq t(n) = \begin{cases} \frac{3n-4}{2}, & \text{if } L = \{0, 3\},\ n \leq 6; \\ \frac{3n-3}{2}, & \text{if } L = \{0, 1\},\ n \leq 7; \\ \frac{3n-2}{2}, & \text{if } L = \{1, 2\},\ n \leq 8; \\ \frac{3n-3}{2}, & \text{if } L = \{2, 3\},\ n \leq 9. \end{cases}$$

If such a graph exists, then it must be reversing-equivalent to F_n shown in Figure 1.

The result files are named as `*_dedup_noC8.txt`.

- `exceed_*_dedup_noC8.txt` contains graphs H with $e(H) > t(n)$.
- `tight_*_dedup_noC8.txt` contains graphs with $e(H) = t(n)$.

If such a file does not exist, then no such graph was found. Each graph in `tight_*_dedup_noC8.txt` is small enough to be manually verified as reversing-equivalent to F_n.

2. File Descriptions

- `main.py`: Main execution file that runs all test cases and coordinates the workflow.
- `example_generator.py`: Core filtering logic, handles $e_{\max}$ filtering and writes graph files.
- `graph_generator.py`: Generates all graphs with a given set of fixed edges by enumerating over free edges.
- `graph_utils.py`: Implements graph condition checks: 4-cycle, path modulo 4, degree checks, etc.
- `dedup_tight_graphs.py`: Deduplicates graph files using NetworkX isomorphism checks.
- `filter_8cycles.py`: Further filters deduplicated graphs to remove those containing 8-cycles.

3. Execution Instructions

To run the project, execute:

```
python main.py
```

This will:

1. Generate graphs for various test cases
2. Save graphs in `tight_l1l2_n.txt` or `exceed_l1l2_n.txt`
3. Deduplicate and write `*_dedup.txt`
4. Filter 8-cycles to produce `*_dedup_noC8.txt`

4. Test Case Format

Each test case is of the form:

```
(l1, l2, i, j, k, fixed_edges_fn)
```

Where:

- ℓ_1, ℓ_2: included path lengths modulo 4
- i, j: range of n to consider
- k: edge threshold parameter, $e_{\max} = \lfloor (3n - k)/2 \rfloor$
- `fixed_edges_fn(n)`: function returning fixed edges for the given n

5. Output Files

- `tight_l1l2_n.txt`: Valid graphs with edge count $= e_{\max}$
- `tight_l1l2_n_dedup.txt`: Non-isomorphic graphs among the above
- `tight_l1l2_n_dedup_noC8.txt`: Further filtered to exclude 8-cycles
- Same for `exceed_l1l2_n.txt`, when graphs exceed $e_{\max}$

6. Dependencies

- Python 3.8+
- `networkx`
- `numpy`
- `os`

To install:

```
pip install networkx numpy os
```

7. Purpose

This tool was developed as part of a broader mathematical investigation into extremal structures in graphs, particularly those avoiding (0 mod 4)-cycles. The filtered graphs serve as gadgets in the study of structural theorems involving modular path lengths and cycle exclusions.