

# On Decompositions, Generation Methods, and related concepts in the theory of Matching Covered Graphs

**Student:** Janmenjaya Panda

**Roll No:** ME20B087

**Degree:** B. Tech.

## Guide: Nishad Kothari

**Degree:** is pertinent to the classical, tight cut decomposition, dependency relations, brace decomposition, brace and tight cut decomposition, and related concepts in the theory of matching covered graphs.

**Guide:** Nishad Kothari

- The creation of a new class `MatchingCoveredGraph`, in the module  `sage.graphs.matching_covered_graph` and the implementation of the following functions:
- `maximal_barrier()`: Return the (unique) maximal barrier of the graph containing the provided vertex.
  - `canonical_partition()`: Return the canonical partition.
  - `light_cut_decomposition()`: Return a maximal set of light cuts of the graph as the decomposition as the shores with these cuts: of the (matching covered) graph.
  - `bricks_and_braces()`: Return the list of (underlying) bricks and braces of the (matching covered) graph.
  - `number_of_bricks()`: Return the number of bricks.
  - `number_of_braces()`: Return the number of braces.
  - `number_of_petersen_bricks()`: Return the number of Petersen bricks. The graph is isomorphic to `graphs.PetersenGraph()`.
  - `is_3Brick()`: Check if the (matching covered) graph is a 3-brick.
  - `is_3Brace()`: Check if the (matching covered) graph is a 3-brace.
  - `is_reversible_wedge()`: Check if the edge of the (matching covered) graph is a reversible wedge.



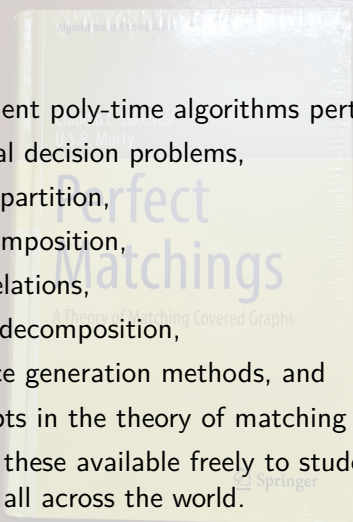
Indian Institute of Technology Madras

# Problem Statement

Implementing efficient poly-time algorithms pertaining to

- ▶ the fundamental decision problems,
- ▶ the canonical partition,
- ▶ tight cut decomposition,
- ▶ dependency relations,
- ▶ (optimal) ear decomposition,
- ▶ brick and brace generation methods, and
- ▶ related concepts in the theory of matching covered graphs,

and to make all of these available freely to students, educators as well as researchers all across the world.

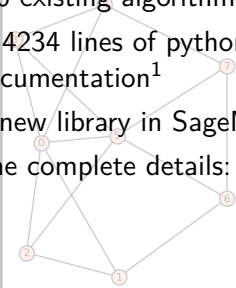


- ▶ Compilation of all existing algorithms available in the literature
- ▶ Devise of new/ equivalent definitions, theorems and derivations of efficient poly time algorithms from constructive/ existential proofs
- ▶ Implementing each of those in python/ cython with SageMath as the code base with the relevant documentations, examples and test cases
- ▶ Communicating with SageMath through Google Summer of Code (GSoC) 2024 to make all of these available publicly.

# Novelty and Results

```
1 G = Graph()
2 G.add_edges([(0, 1), (0, 2), (0, 3), (0, 4), (0, 5),
3             (1, 2), (1, 6), (2, 5), (3, 5), (3, 4),
4             (4, 7), (5, 6), (5, 7), (6, 7)])
5 M = Graph(G.matching())
6 show(G)
7
8 try:
9     answer = is_brick(G, M, matching_covered_check=True)
10    print('Is G a brick? {}'.format(answer))
11 except ValueError as error:
12    print('Error: {}'.format(error))
```

- ▶ 31 functional methods, 12 graph generator methods and 21 algorithms (10 existing algorithms, 11 formulated/ derived)
- ▶  $\approx 4234$  lines of python/ cython code and  $\approx 8807$  lines of documentation<sup>1</sup>
- ▶ A new library in SageMath: Matching Covered Graphs
- ▶ The complete details: [Link](#)



... Is G a brick? True

<sup>1</sup>An estimation

## Relevance

[illegible]