# DARPA MDOP Task 1: Topological Partitioning

## version

**Chad Peterson**

July 10, 2023

# Contents

# Welcome to Task 1's documentation!

`task1.enumeration.`**`read_edge_code`**(`stream`, `size`)

A helper function that reads a single byte from a stream and returns the corresponding edge code.

**Parameters:**
- **stream** (*io.BytesIO*) – A stream to read from.
- **size** (*int*) – The number of bytes to read.

**Returns:** The edge code.

**Return type:** int

`task1.enumeration.`**`shadows_via_plantri_by_edge_codes`**(`num_trivalent_vertices`, `num_crossings`)

A function that enumerates the shadows of an abstract graph with a given number of trivalent vertices and crossings. These shadows are in intermediate step toward enumerating Yamada classes.
Note: This is a wrapper around plantri, which must be installed separately. The plantri executable must be in the current working directory.

**Parameters:**
- **num_trivalent_vertices** (*int*) – The number of trivalent vertices in the abstract graph.
- **num_crossings** (*int*) – The number of crossings in the abstract graph.

**Returns:** A list of shadows, where each shadow is a list of edge codes.

**Return type:** list

`task1.enumeration.`**`spatial_graph_diagrams_fixed_crossings`**(`G`, `crossings`)

A function that enumerates the spatial graph diagrams with a given underlying graph and number of crossings.

**Parameters:**
- **G** (*networkx.Graph*) – The underlying graph.
- **crossings** (*int*) – The number of crossings.

**Returns:** A generator of spatial graph diagrams.

**Return type:** generator

`task1.enumeration.`**`enumerate_yamada_classes`**(`G`, `max_crossings`)

A function that enumerates the Yamada classes of a given underlying graph with a given maximum number of crossings.

**Parameters:**
- **G** (*networkx.Graph*) – The underlying graph.
- **max_crossings** (*int*) – The maximum number of crossings.

**Returns:** A dictionary mapping Yamada polynomials to spatial graph diagrams and the number of examined shadows.

**Return type:** dict, int

`task1.geometric_realizations.`**`rotate_points`**(`positions: ndarray`, `rotation: ndarray | None = None`) → ndarray

Rotates a set of points about the first 3D point in the array.

**Parameters:**
- **positions** (*np.ndarray*) – A numpy array of 3D points.
- **rotation** (*np.ndarray*) – A numpy array of 3 Euler angles in radians.

**Returns:** A numpy array of rotated points.

**Return type:** np.ndarray

`task1.geometric_realizations.`**`subdivide_edge`**(`g`, `edge`, `pos`, `n=3`)

Subdivides an edge into n edges.

**Parameters:**
- **g** (*networkx.Graph*) – The graph to subdivide.
- **edge** (*tuple*) – A tuple containing the IDs of the nodes at the ends of the edge.
- **pos** (*dict*) – A dictionary mapping node IDs to (x, y, z) tuples representing their positions.
- **n** (*int, optional*) – The number of subdivisions to make for each edge. Default is 3.

**Returns:** A tuple containing the new graph and a dictionary mapping node IDs to their new positions.

**Return type:** tuple

`task1.geometric_realizations.`**`isomorphism`**(g, pos, n=3, rotate=True)

Generates an isomorphism of a graph with subdivided edges.

**Parameters:**
- **g** (*networkx.Graph*) – The graph to generate a new realization for.
- **pos** (*dict*) – A dictionary mapping node IDs to (x, y, z) tuples representing their positions.
- **n** (*int, optional*) – The number of subdivisions to make for each edge. Default is 3.
- **rotate** (*bool, optional*) – Whether to randomly rotate the positions of the nodes. Default is False.

**Returns:** A tuple containing the new graph and a dictionary mapping node IDs to their new positions.

**Return type:** tuple

`task1.geometric_realizations.`**`generate_geometric_realizations_for_one_topology`**(spatial_graph, component_radii, num_realizations=5, plot=False)

Generates geometric realizations for a single topology.

**Parameters:**
- **spatial_graph** (*SpatialGraph*) – The spatial graph to generate geometric realizations for.
- **component_radii** (*dict*) – A dict of radii for each component in the topology.
- **num_realizations** (*int*) – The number of geometric realizations to generate.
- **plot** (*bool, optional*) – Whether to plot the geometric realizations. Default is False.

**Returns:** A dictionary mapping realization IDs to a list containing the node positions and edges.

**Return type:** dict

`task1.geometric_realizations.`**`generate_geometric_realizations_for_all_topologies`**(spatial_graphs, component_radii, num_realizations=5, plot=False)

Generates geometric realizations for all topologies.

**Parameters:**
- **spatial_graphs** (*list*) – A list of spatial graphs to generate geometric realizations for.
- **component_radii** (*dict*) – A dict of radii for each component in the topology.
- **num_realizations** (*int*) – The number of geometric realizations to generate.
- **plot** (*bool, optional*) – Whether to plot the geometric realizations. Default is False.

**Returns:** A dictionary mapping topologies to a list of geometric realizations.

**Return type:** dict

`task1.classification.`**`filter_by_environmental_factors`**(all_geometric_realizations, component_radii, environmental_factors, plot=True)

A function that filters geometric realizations by unique combinations of environmental factors.

**Parameters:**
- **all_geometric_realizations** (*dict*) – A dictionary containing the geometric realizations for each unique spatial topology.
- **component_radii** (*dict*) – A dictionary containing the radii of each component node.
- **environmental_factors** (*list of tuple*) – A list of tuples, where each tuple contains a reference point as a 1D array of shape (3,) and a color as a string. The reference points are used to classify the nodes.
- **plot** (*bool*) – A boolean indicating whether to plot the results.

**Returns:** A dictionary containing the filtered geometric realizations for each unique spatial topology.

**Return type:** dict

`task1.classification.`**`filter_by_internal_factors`**(`all_geometric_realizations`, `component_radii`, `internal_factors`, `k=3`)

A function that filters the geometric realizations based on the internal factors of the spatial graph.

**Parameters:**
- **all_geometric_realizations** (*dict*) – A dictionary containing the geometric realizations for each unique spatial topology.
- **component_radii** (*dict*) – A dictionary containing the radii of each component node.
- **internal_factors** (*list*) – A list of nodes that are considered internal factors.
- **k** (*int*) – The number of nearest neighbors to consider.

**Returns:** A dictionary containing the filtered geometric realizations for each unique spatial topology.

**Return type:** dict

`task1.utils.`**`write_output`**(`data`, `output_directory`)

Writes data to a file in the output directory.

**Parameters:**
- **data** (*dict*) – The data to write.
- **output_directory** (*str*) – The path to the output directory.

# Index