Phase 3 Updates

SlashBurn (graph decomposition algorithm)

We successfully implemented the modified **SlashBurn** algorithm that the paper uses as step 1 out of 3 for the entire VoG algorithm. This can be seen in the `src/slashburn.py` file and can be run by calling `$ python3 src/slashburn.py`.

Implementing **SlashBurn** involved:

- Switching the adjacency list output to concentrate the hub nodes to the top left instead of the bottom right.
- Creating a global subgraphs list that only stores subgraphs of size greater than 2 and less than the GCC (Giant Connected Component).

> ```
> Note: GCC was labelled as the largest subgraph generated but we can change that if there is some other method to identify GCC.
> ```

- Adding the hub plus all immediate neighbors as a subgraph in addition to all other subgraphs generated by the original **SlashBurn** method.

MDL

We have begun to implement code that gets the total encoded length for a model *M*. That is, we have been writing code to implement the function *L(M)*. The bulk of getting the encoded length of a model *M* comes from the encoded length of the specific graph structure type *L(s)*, which is the function `get_enocoded_length_by_graph_type` we have written in `src/MDL.py`. Specifically, we have written code for the encoded length of each specific graph structure in our vocabulary omega = {fc, nc, fb, nb, ch, st}.

> Note that `MDL.py` is not runnable because we have not yet adapted the code for the different graph structure classes. However, any functions/properties that a graph object calls within `MDL.py` (such as `graph.numEdges`) already exist in the code we are planning to adapt.

Next steps

Our next steps include:

- Adapting code that deals with identifying a subgraph as a specific graph structure (e.g. star or chain) so that it works with the MDL code we have already written
- Writing a function for the overall encoded length of a model *L(M)* that incorporates the function for *L(s)* we have written
- Writing a function that deals with encoding the length of the error matrix *L(E)*
- Making sure that we can pass in data in the proper format from our non-toy datasets to our SlashBurn algorithm
- Figuring out how many subgraphs are in each model so that we can pass in a model family to step 2 of the algorithm after running SlashBurn
- Implementing step 3 of the VoG algorithm (which deals with heuristics)
- Testing our entire VoG algorithm on the datasets outlined in our project plan