**Research Questions**

The main research question investigated in the thesis is

**RQ1:** How effective is guided distributed concurrency testing at uncovering distributed concurrency bugs in blockchain consensus algorithms?

How to design an effective EA to test blockchain consensus algorithms against distributed concurrency bugs?

To answer the main research questions, I plan to answer the following sub research questions:

**RQ2:** How can the problem of message reordering be encoded for use in EAs?

How can schedules be encoded to be evaluated and modified by EAs?

**RQ3:** What fitness functions provide meaningful guidance to an EA for DCT?

What fitness functions provide meaningful guidance to an EA for DCT compared to random DCT?

**RQ4:** How does GDCT compare to random DCT?

-

**RQ2 Experiment:** Trace graph edit distance?

**RQ3 Experiment below:** using the result from the trace graph edit distance experiment to determine the encoding used?

**Experiment**

To answer the three sub questions, I plan to run one big experiment. This experiment consists of running various configurations of GDCT, and random search against a bug benchmark of three bugs:

**B1 Proposal bug**: Nodes will not check the monotonicity of the sequence number carried in proposal messages from other nodes. This allows an older proposal to override a more recent proposal, thereby enabling nodes to declare consensus on different transaction sets more easily. This violates the agreement property (1).

**B2 Validation threshold bug:** By changing the validation quorum threshold from 80% to 40%, two nodes can validate two different ledgers. This violates the agreement property (2).

**B3 Validation sequence bug:** When a node switches to a ledger which is two or more (TODO: figure out exact number) ledger sequences lower than the ledger it validated last, the node will forget that it previously issued a validation for a ledger with this sequence number. This violates the integrity property (2).

I plan to evaluate the following four configurations of GDCT:

1. Delay encoding with time fitness
2. Delay encoding with highest proposal sequence fitness
3. Priority encoding with time fitness
4. Priority encoding with highest proposal sequence fitness

All configurations will be run for 10 times with a search budget of 30 minutes, for all three bugs separately. These numbers are subject to change based on the initial results. The resulting table will look something like this:

Table

Description automatically generated

The cells will contain the number of runs that successfully discovered the bug through violation of the respective consensus property. Additionally, if for multiple configurations, all runs discover the bug, the average search time can be compared to give a more detailed evaluation of the performance of these configurations.

Please let me know what you think of this setup. Do you think answering one main RQ through these three RQs is a good approach and are these three RQs adequate for answering the main question? Do you agree with running one large experiment for all questions, or would it be better to address the questions through individual experiments?