# Forward and Backward Constrained Bisimulations for Quantum Circuits (Artifact Submission)

In this file we described how to reproduce the experiments of our Paper.

The Artifact can be found in ZENODO: 10.5281/zenodo.8431443.

In this paper we provide the scripts necessary to recreate the tables in the Paper (see Section 4):

- For Table 1, we run the algorithms Grover (`q_search.py`), SAT (`q_sat.py`) and MaxCut (`q_maxcut.py`). The size $n$ of the problem ranges from 5 to 15 qubits in 50 instances. The algorithms compute the simulation of $\sqrt{N}$ iterations (where $N$ is $2^n$) using a reduction with CLUE and then iterating, or iterating directly with DDSIM (included in `mqt.ddsim`).
- For Table 2, we use the benchmarks included in `mqt.bench` to run the CLUE reduction for all entries. We also provide the time execution of one iteration of each circuit using DDSIM (included in `mqt.ddsim`). We repeat each experiment 5 times.

In both cases, we use a timeout of 500 seconds for convenience.

## *Files included*

This repository contains:

- Python scripts (files `misc.py` and `q_*.py`) containing the code to reproduce each experiment in the paper.
- A python script (`print_table.py`) that allow to print the tables that appear in the paper.
- A folder `circuits` with some necessary `.qasm` files with Quantum Assembly code for circuits used in the experiments.
- A folder `results` with an execution of the scripts in the repository.
- Four shell scripts containing:
  - `test.sh`: small and quick test for checking whether the Python code executes properly.
  - `early_light.sh`: a smaller set of examples for generating Table 1 and Table 2 in the paper. It executes faster (~20 minutes) than the full set of experiments.
  - `table1.sh` and `table2.sh`: these scripts execute all necessary experiments to reproduce tables 1 and 2 from the paper.

Moreover, we include a `LICENSE` file, a `README.md` explaining how to install use the files in this repository and a folder `packages` with all the files necessary to run the installation process in a Python 3.10 machine.

## *Installation of Artifact*

1. Download the Artifact from ZENODO: 10.5281/zenodo.8431443.

2. Once downloaded, unzip the Artifact into any folder. The artifact contains all the necessary files to reproduce the experiment in Python 3.10.
3. In a terminal opened in the Artifact folder, run the installation script

```
> make install
```

4. We can check the installation was successful by running the test script:

```
> make test
```

**Estimated time for *Test* script**: 3 minutes.

## *Early Light Review*

We provide a script for *Early Light Review*, besides the test script mentioned above. This script can be executed with the following command:

```
> make early_light
```

This script executes the same experiments as in the paper but reducing the samples from 50 to 5 and the size of the problems to a maximum of 6 qubits in Table 1 and 4 qubits in Table 2.

After executing this script, the user can recover the results necessary for the Tables 1 and 2 as normal (see below in this README the section *How to read results*).

**Estimated time for *Early Light* script**: 20 minutes.

## *Replicating results on the paper*

### 1. Executing the scripts

To recover the tables from the paper, the user can run the scripts `table1.sh` and `table2.sh` included in the artifact. These two scripts can be executed at once using the command:

```
> make all
```

**Estimated time for executing all the scripts**: 100 hours.

### 2. How to read results

The results of the script are stored in CSV files in the folder `results`. Each of these CSV files contains the result for a specific experiment with the name format `q_<<experiment>>_<<type>>.csv`, where:

- `<<expriment>>` is the name of the Python script executed (`search` for Grover, `sat` for SAT, `maxcut` for MAXCUT and `benchmark` for all experiments of Table 2)
- `<<type>>` indicates the type of execution. It can be:

- `clue`, `ddsim` or `direct`: indicates a reduction of the model is done using CLUE, DDSIM or a specific CLUE version explioting the diagonal structure of the problem.
- `full_clue`, `full_ddsim` or `full_direct`: indicates the execution of $\sqrt{N}$ iterations of the model. In `full_clue` and `full_direct` we first perform a model reduction using CLUE (in case of `direct`, we exploit the diagonal structure of the problem). In `full_ddsim` we do the simulations directly using DDSIM.

In these files, each line contains information for one execution over one instance of the problem. In order to obtain the averages displayed in the paper, we provide a Python script `print_table.py` that compiles and averages the result on one of the CSV files. This script also can create the whole tables of the paper with the following command:

This command can be executed as follows:

```
> python3 print_table.py table<<i>>
```

where `<<i>>` indicates which table of the paper want to be compiled.