CMSC 657 Optional Project

Instructor: Daniel Gottesman

Fall 2024

The final project is *optional*. If you choose to do it, get together a group of up to 4 people (individual projects are fine as well), and decide which option you want:

- Term paper, 5–10 pages long, based on your reading of 1–2 research articles.
- Learn a quantum programming language and implement a quantum algorithm or protocol on a quantum computer and/or simulator.
- Another comparable project, with instructor approval.

IMPORTANT: If you choose to do the project, you must turn in a proposal by October 29, 5 PM by sending me an email at ¡dgottesm@umd.edu¿. Otherwise, your project will not be accepted. See below for the contents of the proposal.

1 Deadlines

- Proposal: October 29, 5 PM (by email).
- Outline: November 19, 5 PM (by email).
- Project final due date: December 6, 11:59 PM (submitted on Gradescope).

2 Proposal

The proposal should consist of a short e-mail saying clearly the following information:

1. Your name and the names of everyone in your group
2. Which project option (paper or programming) you are choosing
3. Either
   (a) If you choose a paper, which research paper or papers you wish to read and write up.
   (b) If you choose a program, which algorithm or protocol you will be implementing and what you are going to test about the program. You may also choose to mention which computer or simulator you would like to use and what language you will be using.

You are not committed to stick with the details of your proposal if they are not working out. However, if you wish to change any aspect of your proposal, you must check with me to make sure the new plan is viable and appropriate for the class.

If you are having trouble choosing some aspects of the project, please contact me in advance of the proposal due date and tell me what you have already chosen, and I will try to make some suggestions to help narrow down your options.

Once the proposal is submitted, I may also have feedback or make some suggestions in case there is some aspect of the proposal that is not appropriate for the class or if I have doubts about its feasibility.

3 Outline

The outline consists of a more detailed breakdown of what will be included in the project.

If you are doing the paper option, the outline should give the structure of the paper: section titles and the main points to be included in each section.

If you are doing the program option, the outline should include which language you will be using, plus a mention of whether you will be running on a simulator or a real computer. If you would like to use a real computer, explain what is needed for you to access that computer. (For instance, be aware that cloud quantum computers may have a queue to run programs, so you should have some idea of how long that queue is likely to be.) You should also explain in more detail how you are going to test the program. If you already have some working code, you could include that as well.

Again, you are not fully committed to decisions you make in the outline, but I want to see some progress to ensure that you are not going to be stuck trying to pull the whole project together at the last minute.

The outline will be worth 10% of the project grade.

4 General Instructions

Whether you do the paper option or the programming option, please cite all outside resources (including AI tools) used in the project. For a group project, the final project should include a description of what each group member contributed to the project.

5 Paper Option

In this case, you should read 1–2 related research papers and write up, in your own words, a discussion of the results and methods. The write-up should be 5–10 pages with reasonable font size and margins. Usually 1 research paper will be sufficient material, but you may also need to study some background papers to learn material not discussed in the paper itself.

The write-up should be at a level comprehensible to a student who has just taken this class and understood all the material in it. This means you will need to explain any background material not covered in class.

The write-up should be in your own words and should not imitate the structure or presentation of the research paper too closely. If you are having trouble doing this, try putting aside the research paper and writing up your own version without directly referencing the original paper. Then go back and check any points where you are unsure if you correctly described the point.

You should also cite in your write up the original paper(s) and any other papers whose results you need to reference. The structure of the paper should be similar to that of a research paper. In particular, it should include an introduction explaining why the problem addressed in the paper is interesting and a clear statement of the paper's main results. If the length of your write-up is too short to fully justify the results presented in the paper, you don't need to fully explain the details of the paper but can instead give a summary of the methods used in the research paper.

The topic of the chosen research paper can be basically anything in the field of quantum information provided it is not something we covered in class. Here are the major modules of the class (after the introduction to quantum mechanics), with some ideas for possible projects relating to them. You will need to investigate further to pick a specific paper on this topic. This is not intended to be exhaustive; other ideas are possible.

The paper chosen should not be one on which you are an author.

- Quantum Circuits:
  - Proofs of universality results
  - Circuit identities
  - Gate synthesis algorithms (compilation)

- Quantum Algorithms and Complexity:
  - Quantum complexity classes other than BQP
  - Quantum communication complexity
  - Lower bounds on oracle algorithms
  - Further details about any of the algorithms we discussed in class
  - Quantum algorithms other than those we discussed in class
- Implementations of Quantum Computation:
  - Results of specific experiments performed with any implementation of a quantum computer (whether it is one we discussed in class or not)
  - Quantum annealing, adiabatic quantum computing, cluster states, or other alternative models
  - Theory or experiment for quantum supremacy
- Quantum Error Correction and Fault Tolerance:
  - Quantum error correcting codes other than those we discussed in class
  - Discussions of other aspects of quantum error correction (general properties, bounds on codes, encoding circuits, decoding algorithms, etc.)
  - More detailed discussion of any aspect of fault tolerance
- Quantum Information Theory:
  - Other measures of entanglement
  - Resource theories other than entanglement
  - Additional results about channel capacities
  - More about non-locality
  - Other properties of quantum states
- Quantum cryptography:
  - Quantum key distribution protocols other than BB84
  - Security proofs for quantum key distribution
  - Aspects of real-world realizations of quantum key distribution
  - Quantum cryptographic protocols other than quantum key distribution

6 Programming Option

In this case, the project will consist of writing a program in a quantum programming language and running it on a quantum computer or a simulator of a quantum computer. You must also define a goal of something to test about the program by running it.

You will turn in a copy of the source code of your program, and a writeup explaining what you tested, how you tested it, the results of that test, and the conclusions you can draw from the test.

The program should implement either a quantum algorithm or another quantum protocol, such as a quantum error-correcting code. You can choose an algorithm or protocol that we discussed in class (or covered in a problem set) or one that we did not do in class. However, it should not be a program that duplicates an existing tutorial for the language you choose (although it is acceptable to write your own program for the same task).

You will have to choose a quantum programming language to write in. The most popular languages are written by large companies, but should be available for free. Examples of quantum languages are Qiskit (IBM), Cirq (Google), and Q# (Microsoft). Qiskit and Cirq are based on Python, whereas Q# is a separate language.

Once you write your program, run it on a quantum computer or quantum simulator of your choice. This is the stage at which you should be testing something about the program. Because quantum algorithms often have probabilistic outcomes, you will likely need to run the same program many times to get reasonable statistics to say something meaningful.

An example goal (and a good default if you cannot think of another appropriate thing to test) is to compare the behavior of the algorithm with and without noise. This can either be noise in the real device (if you used one) or simulated noise added into the simulation.

Another possibility, if you are implementing a heuristic algorithm, is to evaluate the performance of the algorithm on an assortment of tasks, ideally compared to a classical algorithm for the same thing. Non-heuristic algorithms may still perform better on some instances than others, so that is possible thing to test for.

For implementing your program, the various major quantum languages have simulators provided by the companies that designed the languages, generally built into the development environment. If you want to use a real quantum computer, there are a variety of cloud options. IBM has its own cloud service (including some limited free access), and many other companies offer cloud access through Microsoft Azure Quantum and Amazon Braket. QLab is another possibility: It is a University of Maryland facility with access to an IonQ quantum computer. If you are interested in using QLab, talk to me and we can investigate what the possibilities are for getting access.

Be aware that any classical simulation will necessarily have a severely limited number of qubits, as will the real devices that are most easily available. This should be taken into account in your plan for the project.

7 Grading

If you choose to do the final project, it will replace your lowest two problem set grades.

The project will be worth a total of 120 points, broken down as follows:

Paper project: correctness 30%, depth (including choice of material and context) 30%, clarity (includes conceptual organization, aimed at right background level, and quality of explanation) 25%, format (including proofreading) 5%, outline 10%

Program project: correctness of code 25%, design of test 20%, implementation of test and running of program 20%, writeup (including conclusions about circuit) 25%, proposal 10%