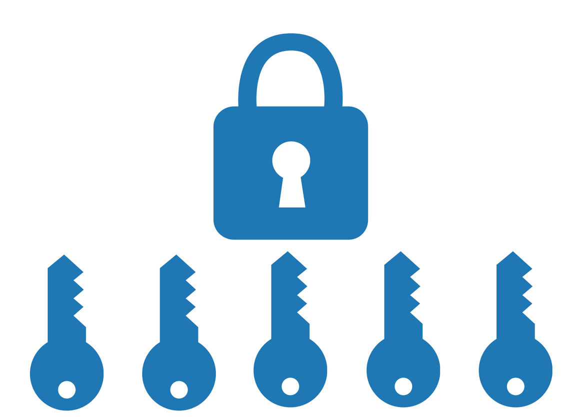
PERFORMANCE OF SIMON'S QUANTUM ALGORITHM ON NISQ DEVICES

INTRODUCTION

Quantum computers have the potential to revolutionize computer science because of their unique approach to solving problems. Daniel Simon produced a quantum algorithm that solves a specific problem exponentially faster than its classical counterpart. However, this speedup assumes access to a perfect quantum computer —a dream that has not yet been realized. Rather, the quantum computers of today, referred to as Noisy Intermediate-Scale Quantum (NISQ) devices, are plagued by a nontrivial probability of error. To better understand how imperfect hardware affects the performance of Simon's algorithm, we ran the algoirthm on many NISQ devices.

How Does a Quantum Computer Work?

It can be helpful to think about some problems that a computer solves like *finding a key for a lock*. When there are many possible keys this can take a long time. A quantum computer can solve these problems much faster than a regular computer.



A regular computer tries each key one by one. It can take a long time to find the right key.

A quantum computer represents many keys at the same time. It uses interference to converge on the right key in a short amount of time.

BACKGROUND

Simon's problem takes as input a function which operates on bitstrings of size n. The function is either one-to-one, or two-to-one with a fixed secret string s that relates each pair of inputs. The objective is to classify the provided function into one of the two classes. Simon invented a quantum algorithm which completes this task in polynomial time, while a classical computer must run a probabilistic calculation with an exponential expected runtime.

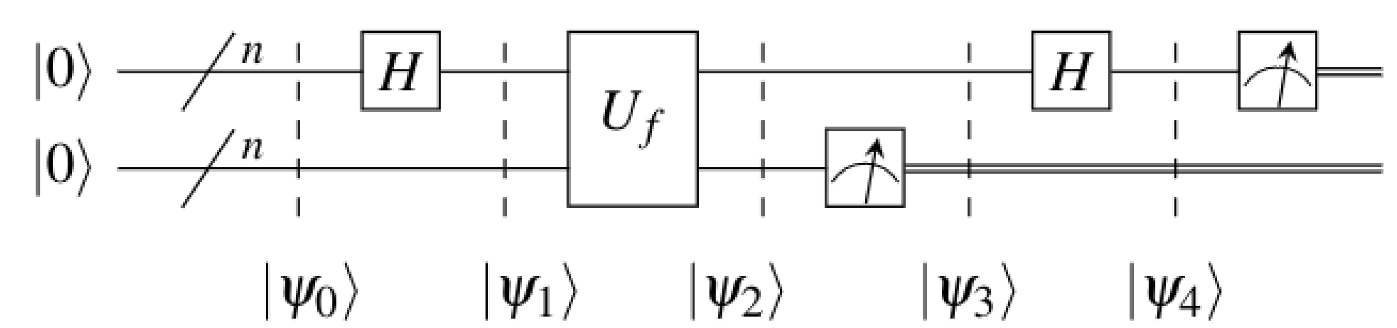


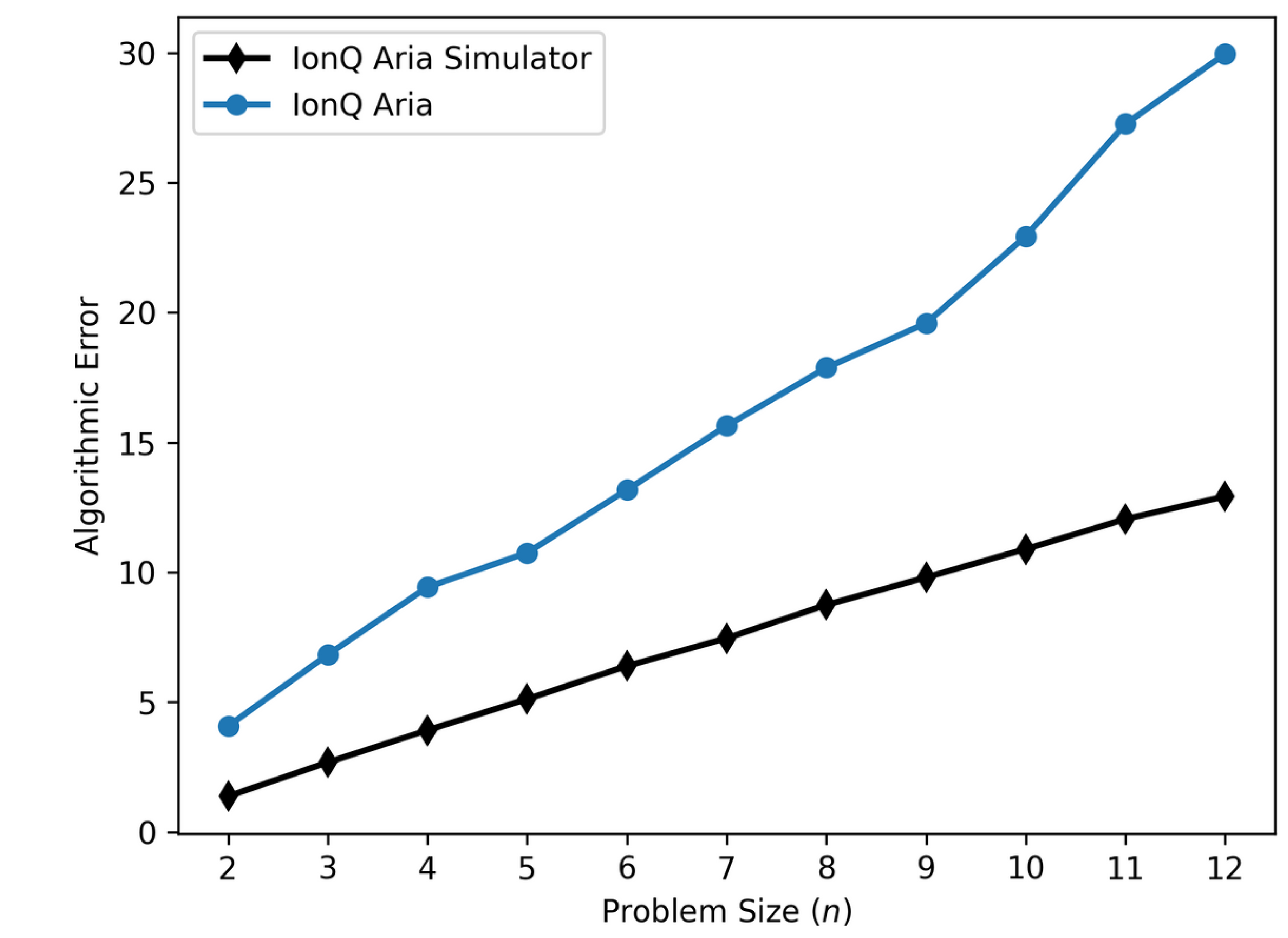
Figure: The circut diagram for Simon's Algorithm.

METHODS

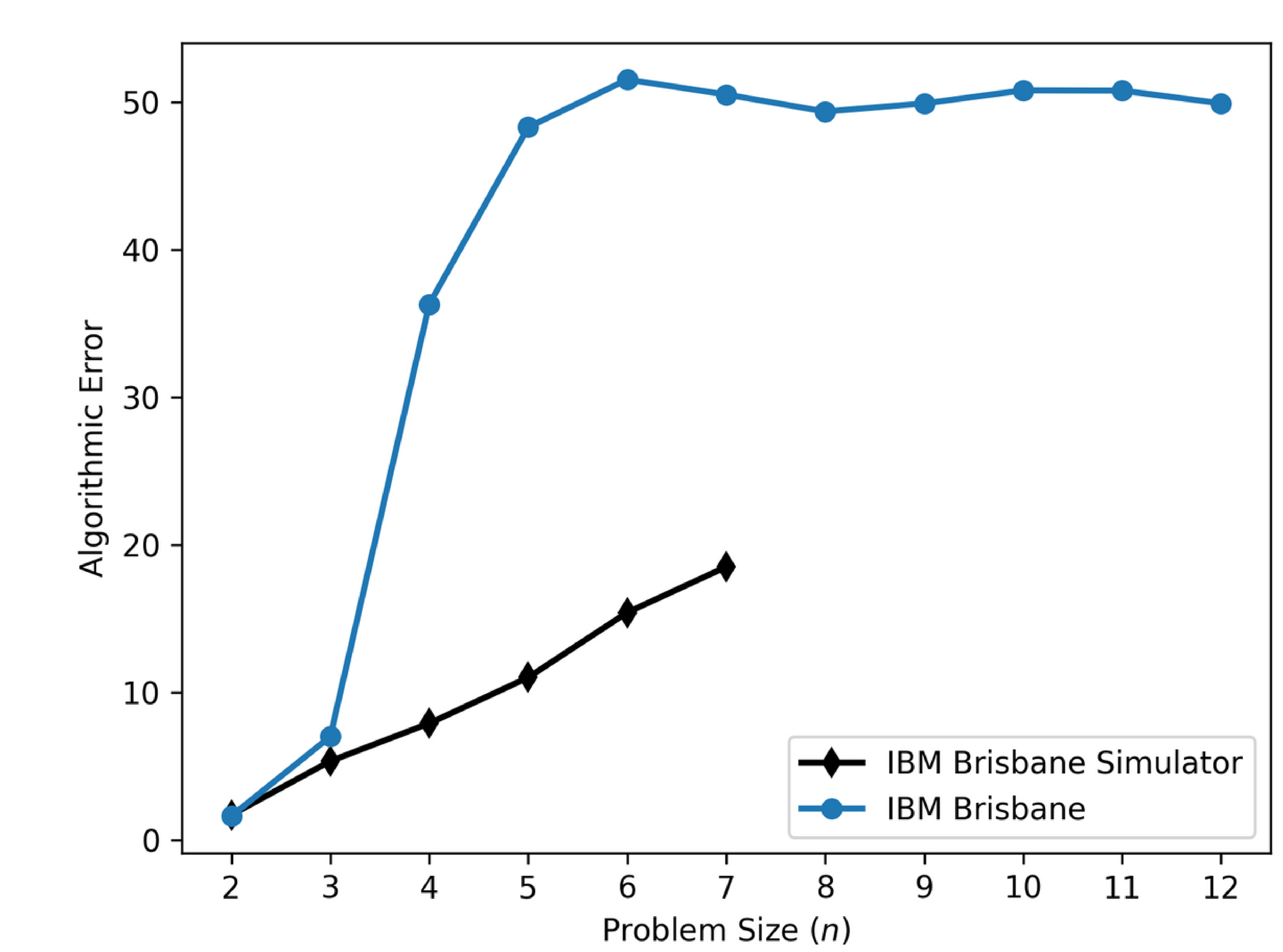
We ran Simon's algorithm on several quantum devices:

1. We initialized a variable n to iterate over the values 2—12.
2. For each n, we defined a two-to-one function using the string of n 1s as the secret string s.
3. We prepared an implementation of Simon's algorithm to classify this function.
4. We repeated the implementation 8192 times.
5. We counted the number of times the implementation misclassified the function (the "algorithmic error").
6. We took the average over several repetitions of the above.
7. We plotted the results.

RESULTS



The first plot shows the algorithmic error on the IonQ Aria scales linearly. The second plot shows that IBM Brisbane experiences a nonlinear jump at n=4, after which the hardware fails to implement the algorithm. However, with 30% algorithmic error for n=12, the IonQ device is not performing well either—we expect this to nullify any quantum advantage. In short, we find that the algorithmic error of Simon's algorithm on NISQ devices scales at least linearly, and all devices show irrecoverable error for intermediate-scale problems.



REECE J. ROBERTSON
UMBC
REECEROBERTSON@UMBC.EDU