Adiabatic Quantum Computation

Elliot Snow-Kropla
Kyriakidis Group

January 3, 2014

The most straightforward way to make an AQC circuit for a one-way function is to take advantage of the fact that out computations are reversible; thus we just make a circuit for computing the function going the easy way.

1 Introduction

Computers are very useful (duh). But some computational problems are impractical to solve on classical computers. Peter Shor[2] showed that a quantum computer could factor integers and calculate the discrete logarithm faster than a classical computer could. So the question becomes, how do we build a quantum computer? Most quantum computer proposals revolve around constructing quantum gates (in analogy with classical logical gates) to manipulate quantum states, but this is hard. The more qubits you add the harder it is to shield your computer from the outside world, so you get more decoherence. You add more qubits to counter-act decoherence ... So what I've looked at is an alternative approach to building a quantum computer based around adiabatic annealing.

2 Background

2.1 Classical Computing

2.2 Quantum Computing

2.3 Adiabatic Quantum Computing

3 Quantum Circuit Compilation

We need a way of generating Hamiltonians to solve the problems we are trying to solve. In general, creating a Hamiltonian to solve any given problem *ex nihilo* is NP-Hard. We can build up problem Hamiltonians out of the Hamiltonians of smaller sub-problems.[1]

Given this, if we can find a Hamiltonian which is universal for computation, we can build up any arbitrary Hamiltonian we need. The logical `NAND` is universal for computation, so if we can encode that we can encode any problem.

But can we encode problems in such a way that we can solve them faster than classically?

3.1 QSM Language

3.2 Quantum Circuit

3.3 Embedding

4 Results

5 Conclusion

```
def modular_power(b, e, m):
  r = 1
  while e > 0:
    if (e % 2) == 1:
      r = (r * b) % m
    e = e >> 1
    b = (b * b) % m
  return r
```

References

[1] Micah M. McCurdy and Jeffrey Egger. Gluing theorem paper. 2014.

[2] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *arXiv*, 1996.