Optimisation via Adiabatic Quantum Computing

Vivek Katial

22/01/2020

Introduction

Main Research Question

- What instances characteristics of optimisation problems make them predisposed to being solved on a Quantum Computer?

Background

Adiabatic Quantum Computing (AQC)

- Adiabatic Quantum Computation is a computational model which relies on the *adiabatic thereom* of quantum mechanics to compute calculations.

Adiabatic Quantum Computing (AQC)

- Adiabatic Quantum Computation is a computational model which relies on the *adiabatic thereom* of quantum mechanics to compute calculations.
- The famous *Schrödinger Equation* is well-known to describe the time evolution of a quantum state:

Adiabatic Quantum Computing (AQC)

- Adiabatic Quantum Computation is a computational model which relies on the *adiabatic thereom* of quantum mechanics to compute calculations.
- The famous *Schrödinger Equation* is well-known to describe the time evolution of a quantum state:
- $$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi t\rangle$$

Adiabatic Quantum Computing (AQC)

- Adiabatic Quantum Computation is a computational model which relies on the *adiabatic thereom* of quantum mechanics to compute calculations.
- The famous *Schrödinger Equation* is well-known to describe the time evolution of a quantum state:
- $$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi t\rangle$$
- $|\psi(t)\rangle$ is our state vector, $H(t)$ is the time dependent Hamiltonian. A Hamiltonian of an n-qubit system $H(t)$ is given by $2^n \times 2^n$ matrix.

Adiabatic Quantum Computing (AQC)

- An adiabatic computation can be expressed by specifying two Hamiltonians, denoted by H_B and H_P where H_B is our *initial* Hamiltonian and H_P is the *final* or *problem* Hamiltonian.

Adiabatic Quantum Computing (AQC)

- An adiabatic computation can be expressed by specifying two Hamiltonians, denoted by H_B and H_P where H_B is our *initial* Hamiltonian and H_P is the *final* or *problem* Hamiltonian.
- $$\begin{array}{ccc} H_B & \xrightarrow{\hspace{4cm}} & H_P \\ |\psi_B\rangle & \text{transform } H(t) & |\psi_P\rangle \\ \text{known state} & & \text{unknown} \end{array}$$

Adiabatic Quantum Computing (AQC)

- An adiabatic computation can be expressed by specifying two Hamiltonians, denoted by H_B and H_P where H_B is our *initial* Hamiltonian and H_P is the *final* or *problem* Hamiltonian.
- $$\begin{array}{ccc} H_B & \xrightarrow{\hspace{4cm}} & H_P \\ |\psi_B\rangle & \text{transform } H(t) & |\psi_P\rangle \\ \text{known state} & & \text{unknown} \end{array}$$
- Loosely speaking, the adiabatic theorem tells us that if we vary from H_B to H_P *slowly enough* the system will remain in its ground state. This fact is a direct result of the Adiabatic Theorem.

Adiabatic Quantum Computing (AQC)

- To conduct the computation we evolve $|\psi(t)\rangle$ till time $t = T$ such that $|\psi(t = T)\rangle$ encodes the answer. The computation is done using a Hamiltonian which linearly interpolates between H_B and H_P. Specifically as below:

Adiabatic Quantum Computing (AQC)

- To conduct the computation we evolve $|\psi(t)\rangle$ till time $t = T$ such that $|\psi(t = T)\rangle$ encodes the answer. The computation is done using a Hamiltonian which linearly interpolates between H_B and H_P. Specifically as below:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$

Adiabatic Quantum Computing (AQC)

- To conduct the computation we evolve $|\psi(t)\rangle$ till time $t = T$ such that $|\psi(t = T)\rangle$ encodes the answer. The computation is done using a Hamiltonian which linearly interpolates between H_B and H_P. Specifically as below:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$
- How fast?

Adiabatic Quantum Computing (AQC)

- To conduct the computation we evolve $|\psi(t)\rangle$ till time $t = T$ such that $|\psi(t = T)\rangle$ encodes the answer. The computation is done using a Hamiltonian which linearly interpolates between H_B and H_P. Specifically as below:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$
- How fast?
- $$T = \frac{1}{\min_t g(t)^2}$$

Adiabatic Quantum Computing (AQC)

- To conduct the computation we evolve $|\psi(t)\rangle$ till time $t = T$ such that $|\psi(t = T)\rangle$ encodes the answer. The computation is done using a Hamiltonian which linearly interpolates between H_B and H_P. Specifically as below:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$
- How fast?
- $$T = \frac{1}{\min_t g(t)^2}$$
- Where $g(t)$ is the difference between the first two smallest eigenvalues of $H(t)$

3SAT (Exact Cover)

- The satisfiability problem, abbreviated SAT, is a classic example of an NP-complete problem [1]

3SAT (Exact Cover)

- The satisfiability problem, abbreviated SAT, is a classic example of an NP-complete problem [1]
- The basic SAT formulation can be described as follows: Given a boolean formula (AND $\wedge$, OR $\vee$, NOT $\neg$) over n variables $(z_1, z_2, ..., z_n)$. Can one set z_i's in a manner such that the Boolean formula is true?

3SAT (Exact Cover)

- The satisfiability problem, abbreviated SAT, is a classic example of an NP-complete problem [1]
- The basic SAT formulation can be described as follows: Given a boolean formula (AND $\wedge$, OR $\vee$, NOT $\neg$) over n variables $(z_1, z_2, ..., z_n)$. Can one set z_i's in a manner such that the Boolean formula is true?
- A clause is an expression which the variables must satisfy. For example $z_1 \wedge z_2 \implies z_1 = z_2 = 1$

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:
  - $z_1 + z_2 = 1$

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:
  - $z_1 + z_2 = 1$
  - $z_1 + z_3 = 1$

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:
  - $z_1 + z_2 = 1$
  - $z_1 + z_3 = 1$
- Here we have 8 possible assignments, namely $\{000, 001, 010, 011, 100, 101, 110, 111\}$

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:
  - $z_1 + z_2 = 1$
  - $z_1 + z_3 = 1$
- Here we have 8 possible assignments, namely $\{000, 001, 010, 011, 100, 101, 110, 111\}$
- However, our **satisfying assignments** are $z_1 = 1, z_2 = 0, z_3 = 0$ or $z_1 = 0, z_2 = 1, z_3 = 1$

3SAT (Exact Cover Example)

- Consider a 3-bit number with two clauses:
  - $z_1 + z_2 = 1$
  - $z_1 + z_3 = 1$
- Here we have 8 possible assignments, namely $\{000, 001, 010, 011, 100, 101, 110, 111\}$
- However, our **satisfying assignments** are $z_1 = 1, z_2 = 0, z_3 = 0$ or $z_1 = 0, z_2 = 1, z_3 = 1$
- The 3-SAT problem in particular is a problem where each clause is comprised of 3 literals.

Mapping 3SAT to AQC

- How do we construct H_B and H_P so that we can solve our optimisation problem?

Mapping 3SAT to AQC

- How do we construct H_B and H_P so that we can solve our optimisation problem?
- Farhi et al describes it very well.

Mapping 3SAT to AQC (H_P)

- Finally we express our problem Hamiltonian as follows:

Mapping 3SAT to AQC (H_P)

- Finally we express our problem Hamiltonian as follows:
- $$H_P = \sum_C H_P^C$$

Mapping 3SAT to AQC (H_P)

- Finally we express our problem Hamiltonian as follows:
- $$H_P = \sum_C H_P^C$$
- We then evolve our system with the following interpolation:

Mapping 3SAT to AQC (H_P)

- Finally we express our problem Hamiltonian as follows:
- $$H_P = \sum_C H_P^C$$
- We then evolve our system with the following interpolation:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$

Mapping 3SAT to AQC (H_P)

- Finally we express our problem Hamiltonian as follows:
- $$H_P = \sum_C H_P^C$$
- We then evolve our system with the following interpolation:
- $$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P$$
- Finally, complete measurement

Proposed Research

- Depending on different structures of instances, classical algorithms may perform differently to Quantum ones.

Proposed Research

- Depending on different structures of instances, classical algorithms may perform differently to Quantum ones.
- We are looking to investigate which types of instances are more pre-disposed to being solved on Quantum Computers:

Proposed Research

- Depending on different structures of instances, classical algorithms may perform differently to Quantum ones.
- We are looking to investigate which types of instances are more pre-disposed to being solved on Quantum Computers:
- To do this we will simulate AQC and measure the "quantum-ness" of each instance:

Proposed Research

- Depending on different structures of instances, classical algorithms may perform differently to Quantum ones.
- We are looking to investigate which types of instances are more pre-disposed to being solved on Quantum Computers:
- To do this we will simulate AQC and measure the "quantum-ness" of each instance:
    1. Minimum Energy Gap g_{min}

Proposed Research

- Depending on different structures of instances, classical algorithms may perform differently to Quantum ones.
- We are looking to investigate which types of instances are more pre-disposed to being solved on Quantum Computers:
- To do this we will simulate AQC and measure the "quantum-ness" of each instance:
    1. Minimum Energy Gap g_{min}
    2. Maximum Entropy of Entanglement

Proposed Research

- This will involve large amounts of generating experimental data

Proposed Research

- This will involve large amounts of generating experimental data
- To survey the instances we will deploy the instance space analysis methodology

Proposed Research

- This will involve large amounts of generating experimental data
- To survey the instances we will deploy the instance space analysis methodology
- Ultimately, we can build a prediction model which maps instance features of different 3SAT instances to their inherent "Quantumness"

Proposed Research

- Given that we have access to IBM's Quantum Computer, we may also apply a discretized version of AQC, QAOA and produce results from a *real quantum computer*

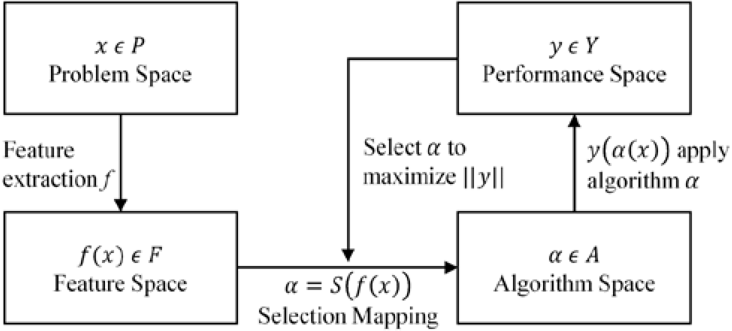
Proposed Research

- Given that we have access to IBM's Quantum Computer, we may also apply a discretized version of AQC, QAOA and produce results from a *real quantum computer*
- This methodology can then be further extended to other optmisation problems that can be mapped to Quadratic Unconstrained Binary Optimisation (QUBO) problems.

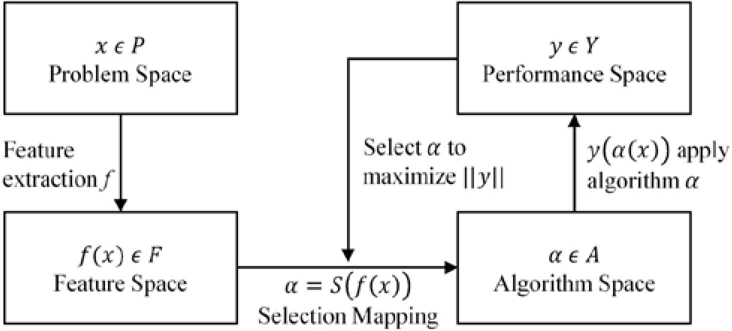
Algorithm Selection

- Given a set of problem instances, predicting which algorithm is most likely to best perform was first explored by Rice [2].

Algorithm Selection

- Given a set of problem instances, predicting which algorithm is most likely to best perform was first explored by Rice [2].
- 

Algorithm Selection

- Given a set of problem instances, predicting which algorithm is most likely to best perform was first explored by Rice [2].
- 

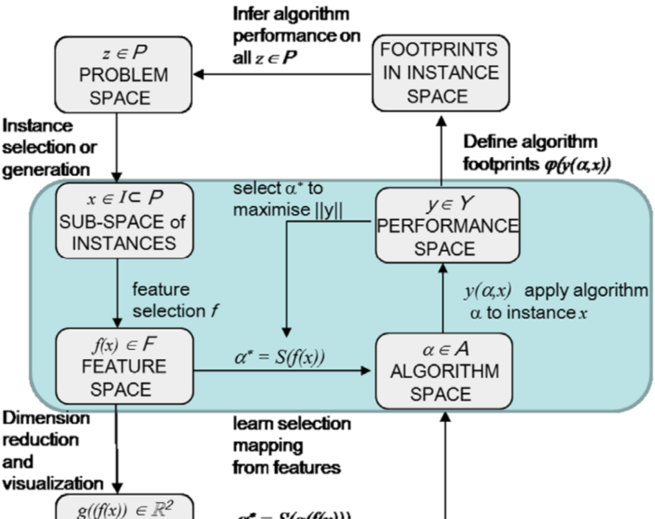
- However, what we are interested in is probing the strengths and weaknesses of AQC for different instances of SAT.

Instance Space Methodology

- The instance space methodology presented in [3–5] extends Rice's framework.

Instance Space Methodology

- The instance space methodology presented in [3–5] extends Rice's framework.



Hardness Features for 3SAT

- Selman et. al demonstrated [6] the existence of a phase transition for random 3SAT problems.

Hardness Features for 3SAT

- Selman et. al demonstrated [6] the existence of a phase transition for random 3SAT problems.
- SATZilla and Nudelman et. al have identified 84 different features for SAT instances [7,8].

Hardness Features for 3SAT

- Selman et. al demonstrated [6] the existence of a phase transition for random 3SAT problems.
- SATZilla and Nudelman et. al have identified 84 different features for SAT instances [7,8].

Hardness Features for 3SAT

- Selman et. al demonstrated [6] the existence of a phase transition for random 3SAT problems.
- SATZilla and Nudelman et. al have identified 84 different features for SAT instances [7,8].

1. Problem Size Features
2. Variable-Clause Graph Features
3. Variable Graph Features
4. Clause Graph Features
5. Proximity to Horn Formula

Hardness Features for 3SAT

- Selman et. al demonstrated [6] the existence of a phase transition for random 3SAT problems.
- SATZilla and Nudelman et. al have identified 84 different features for SAT instances [7,8].

1. Problem Size Features
2. Variable-Clause Graph Features
3. Variable Graph Features
4. Clause Graph Features
5. Proximity to Horn Formula
6. Balance Features
7. DPLL Probing Features
8. Local Search Probing Features
9. LP-Based Features

Quantum Computing Research into Hard SAT problems

- Different Paths (Farhi 2009)
- Hamiltonian Parameters (Choi)
- Volks group research on SAT

Current Progress

1. Learning Quantum Mechanics and Quantum Computing

Current Progress

1. Learning Quantum Mechanics and Quantum Computing
2. Conducting literature review on current advances in Adiabatic Quantum Computing.

Current Progress

1. Learning Quantum Mechanics and Quantum Computing
2. Conducting literature review on current advances in Adiabatic Quantum Computing.
3. Developed working environment, set up GitHub, scripts, organizational tools enabling an efficient working environment.

Current Progress

4. Developed an architecture to run reproducible experiments at scale

Current Progress

4. Developed an architecture to run reproducible experiments at scale
5. Provisioned all infrastructure required for above on Melbourne Research Cloud and SPARTAN

Current Progress

4. Developed an architecture to run reproducible experiments at scale
5. Provisioned all infrastructure required for above on Melbourne Research Cloud and SPARTAN
6. Developed a working implementation of Adiabatic Quantum Computing on different instances of 3SAT.

Next Steps (till Confirmation)

1. Complete Literature Review of current advances in AQC.

Next Steps (till Confirmation)

1. Complete Literature Review of current advances in AQC.
2. Implement scripts to generate **hard** instances of 3SAT with respect to *instance characteristics*.

Next Steps (till Confirmation)

1. Complete Literature Review of current advances in AQC.
2. Implement scripts to generate **hard** instances of 3SAT with respect to *instance characteristics*.
3. Apply Instance Space Methodology from MATILDA to find decision boundaries for *Entanglement Entropy* and *Minimum Energy Gap*

Next Steps (till Confirmation)

4. Extend implementation to work on QAOA and VQE (discretized version of AQC)

Next Steps (till Confirmation)

4. Extend implementation to work on QAOA and VQE (discretized version of AQC)
5. Learn C++ effectively, this will be required when implementing larger simulations.

Next Steps (till Confirmation)

4. Extend implementation to work on QAOA and VQE (discretized version of AQC)
5. Learn C++ effectively, this will be required when implementing larger simulations.
6. Apply Tensor Network methodology on current simulations

Next Steps (till Confirmation)

4. Extend implementation to work on QAOA and VQE (discretized version of AQC)
5. Learn C++ effectively, this will be required when implementing larger simulations.
6. Apply Tensor Network methodology on current simulations
7. Further advance knowledge of Quantum Physics (possibly through taking a course in Quantum Computing)

Next Steps (till Confirmation)

4. Extend implementation to work on QAOA and VQE (discretized version of AQC)
5. Learn C++ effectively, this will be required when implementing larger simulations.
6. Apply Tensor Network methodology on current simulations
7. Further advance knowledge of Quantum Physics (possibly through taking a course in Quantum Computing)
8. Extend research to look at other optimisation problems

About Me

- Vivek Katial (vkatial@student.unimelb.edu.au)
  - PhD Candidate (Optimisation on Quantum Computers)
- Check out the slides at https://tinyurl.com/vkatial-preconfirmation

References

References I

1. Cook SA (1971) The complexity of theorem-proving procedures, In, *Proceedings of the third annual acm symposium on theory of computing*, ACM, 151–158.

2. Rice JR, others (1976) The algorithm selection problem. *Advances in computers* 15: 5.

3. Smith-Miles K, Bowly S (2015) Generating new test instances by evolving in instance space. *Computers & Operations Research* 63: 102–113.

4. Smith-Miles K, Lopes L (2012) Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research* 39: 875–889.

References II

5. Smith-Miles K, Baatar D, Wreford B, et al. (2014) Towards objective measures of algorithm performance across instance space. *Computers & Operations Research* 45: 12–24.

6. Selman B, Mitchell DG, Levesque HJ (1996) Generating hard satisfiability problems. *Artificial Intelligence* 81: 17–29.

7. Xu L, Hutter F, Hoos HH, et al. (2008) SATzilla: Portfolio-based algorithm selection for sat. *Journal of artificial intelligence research* 32: 565–606.

8. Nudelman E, Leyton-Brown K, Hoos HH, et al. (2004) Understanding random sat: Beyond the clauses-to-variables ratio, In: Wallace M (Ed.), *Principles and practice of constraint programming – cp 2004*, Berlin, Heidelberg, Springer Berlin Heidelberg, 438–452.