Quantum Computing (CST Part II)

Lecture 15: Adiabatic Quantum Computing

For years experts questioned whether the [D-Wave] devices were actually exploiting quantum mechanics and whether they worked better than traditional computers.

The Economist

The quantum adiabatic theorem

Adiabatic quantum computing is founded upon the quantum adiabatic theorem:

The quantum adiabatic theorem

If we have a time-varying Hamiltonian, $\mathrm{H}(t)$, which is initially H_I at $t = 0$, and subsequently H_F at some later time, $t = t_F$, then if the system is initially in the ground-state of H_I, and as long as the time-evolution of the Hamiltonian is sufficiently slow, the state is likely to remain in the ground-state throughout the evolution, therefore being in the ground-state of H_F at $t = t_F$.

That is, if a quantum system starts in a ground-state, so long as we evolve the state slowly, it is likely to remain in a ground-state.

Proof of the quantum adiabatic theorem, which includes exactly what is meant by “evolving the state slowly” is rather complex and physicsy, and thus beyond the scope of this course: for the purposes of appreciating adiabatic quantum computing as a model of computation, it suffices to simply know the existence of the quantum adiabatic theorem.

Adiabatic quantum computing

Unlike gate-based quantum computing which, by the quantum circuit model, bears some resemblance to classical digital computing, adiabatic quantum computing is a completely different approach to computation, in which it is necessary to specify:

- An initial Hamiltonian, H_I, whose ground-state is easy to prepare.
- A final Hamiltonian, H_F, whose ground-state encodes the solution to the problem of interest.
- An adiabatic evolution path, $s(t)$ where $s(0) = 1$ and $s(t_F) = 0$, which defines the Hamiltonian evolution:

  $$\mathrm{H}(t) = s(t)\mathrm{H}_I + (1 - s(t))\mathrm{H}_F$$

  i.e., such that $\mathrm{H}(0) = \mathrm{H}_I$ and $\mathrm{H}(t_F) = \mathrm{H}_F$. For example, the linear path $s(t) = \left(1 - \frac{t}{t_F}\right)$, such that:

  $$\mathrm{H}(t) = \left(1 - \frac{t}{t_F}\right)\mathrm{H}_I + \frac{t}{t_F}\mathrm{H}_F$$

Therefore, because of the quantum adiabatic theorem, the final state is very likely to be the ground-state of H_F, and so encodes the solution to the problem in question.

Adiabatic quantum computing: a very simple example

Consider the (informally stated) mathematical problem: output two bits, x_1 and x_2 such that $x_1 \neq x_2$. We can associate this with the function:

$$f(x) = \begin{cases} 0 & \text{if } x_1 \neq x_2 \\ 1 & \text{otherwise} \end{cases}$$

which can be “encoded” in the Hamiltonian:

$$\mathrm{H}_F = \frac{I + Z \otimes Z}{2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where I is the identity, and Z is the Pauli-Z matrix. Note that the Pauli matrices are both unitary and Hermitian – and it the latter property that is significant here.

Adiabatic quantum computing: simple example continued

We have that the eigenstates of H_F are:

$$\begin{aligned}\mathrm{H}_F\,|00\rangle &= 1\,|00\rangle\\ \mathrm{H}_F\,|01\rangle &= 0\,|01\rangle\\ \mathrm{H}_F\,|10\rangle &= 0\,|10\rangle\\ \mathrm{H}_F\,|11\rangle &= 1\,|11\rangle\end{aligned}$$

So we have that the lowest eigenvalue (ground) states of H_F are $|01\rangle$ and $|10\rangle$, and so if we adiabatically evolve to H_F and measure we will obtain an outcome where $x_1 \neq x_2$, as desired.

For completeness, we must also specify a suitable initial Hamiltonian. We know that the equal superposition of two-qubit computational basis states, $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ is an easy state to prepare, and it can be verified that this is a ground state of:

$$\mathrm{H}_I = \frac{(I-X)\otimes I + I\otimes(I-X)}{2} = \begin{bmatrix} 1 & -0.5 & -0.5 & 0\\ -0.5 & 1 & 0 & -0.5\\ -0.5 & 0 & 1 & -0.5\\ 0 & -0.5 & -0.5 & 1\end{bmatrix}$$

Applications of adiabatic quantum computing

Adiabatic quantum computing is an alternative approach to the gate model of quantum computing that we have studied in this lecture course.

Adiabatic quantum computing is a universal computational model, and in terms of computational complexity is polynomially equivalent to gate-based quantum computing. This means that any Turing-solvable problem can be solved using adiabatic quantum computing, using only "polynomially more time" (in the problem size) than computing the solution on a gate-based quantum computer. Conversely, any adiabatic quantum algorithm can be discretised and computed on a gate-based quantum computer (again with a polynomial-time overhead at worst).

Nevertheless, adiabatic quantum computing has received significantly less attention than its gate-based counterpart, with the following notable exceptions:

- Adiabatic state preparation as a subroutine in quantum chemistry algorithms.
- Optimisation using quantum annealing – in particular, D-Wave.

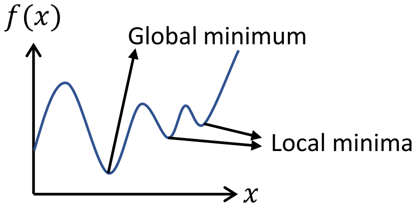
Adiabatic state preparation in quantum chemistry

- Although adiabatic quantum computing requires a continuous state evolution, we previously saw that continuous evolution can be simulated on a gate-based quantum computer, by discretising the evolution into sufficiently short-duration intervals.
- Moreover, even though we previously only looked at state evolution according to a time-invariant Hamiltonian, the same principle applies to state evolution according to a time-varying Hamiltonian – again with the caveat that the duration of the discretised time-intervals must be sufficiently short.
- We also previously saw that in quantum chemistry, for both quantum phase estimation and the variational quantum eigensolver algorithm, it is necessary to prepare the state of the second register in the ground-state of the system of interest.
- So it follows that one way to do this is to use adiabatic state evolution, approximated by discretised evolution on a gate-based quantum computer, from the ground-state of a Hamiltonian that is easy to prepare, to the ground-state of the Hamiltonian of interest.

Optimisation

Mathematical optimisation is the process of finding x such that $f(x)$ is minimised, for some $f : \mathbb{R}^n \to \mathbb{R}$ of interest. The optimisation can be constrained by inequality and/or equality constraints, i.e., finding x in the range $0 \leq x \leq 10$ such that some $f(x)$ is minimised. Optimisation problems broadly fall into three categories:

1. Algebraically solvable optimisation, for example finding x which minimises $f(x) = x^2 - 2x - 5$.
2. Convex optimisation, which concerns the optimisation of a class of functions which have a single "global" minimum, and for which efficient optimisation algorithms exist.
3. General optimisation, which concerns optimisation of any function, which may therefore have multiple "local" minima:

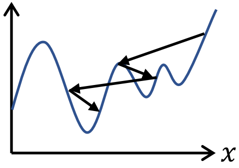


Metaheuristics

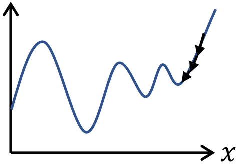
Metaheuristics are used to find “good” approximate solutions to general optimisation problems. In plain terms, a metaheuristic is a search policy that explores the optimisation function, $f(x)$, by evaluating it at certain values of x.

There are myriad metaheuristic algorithms which decide where next (at which value of x) to evaluate $f(x)$ given the history of function evaluations, but all are based on the same essential principle that good solutions are likely to be near other good solutions, or in other words that the optimisation surface has some smoothness. This in turn reveals the *exploration versus exploitation* trade-off that all metaheuristics must make.

Exploration versus exploitation



Explore



Exploit

- A metaheuristic can exploit its "current" position, by descending incrementally. The risk is that this returns a (possibly not very good) local minimum.
- Alternatively a metaheuristic can explore the optimisation surface by making "large movements" to discover whether another part of optimisation surface returns smaller values of $f(x)$. In this case, the global minimum may be found, but the value of x returned may only be a fairly poor approximation of the actual global minimum x.

Metaheuristic optimisation algorithms are typically set-up to search for a minimum in such a way that exploration is favoured at the start and exploitation at the end. Different metaheuristics do this in different ways, and so are well-suited to different types of optimisation problems.

Simulated annealing

Simulated annealing is a metaheuristic inspired by the physical process of thermal annealing. Simulated annealing applies to either combinatorial optimisation, or a continuous optimisation problem whose optimisation surface has a defined notion of neighbourhood. An initial x is chosen, and the algorithm proceeds as follows:

1. Evaluate $f(x)$
2. At random choose a neighbour, x' of x
3. Evaluate $f(x')$.
4. If $f(x') < f(x)$ then set $x \leftarrow x'$
    - Else (i.e., $f(x') \geq f(x)$) randomly decide whether to set $x \leftarrow x'$ or just keep x as it is
5. Repeat a specified number of times

Simulated annealing (cont.)

Naturally, the random decision in step 4 is such that the more that $f(x')$ exceeds $f(x)$ by the less likely it is that $x \leftarrow x'$. Additionally, this random process varies throughout the repeated iterations of the algorithm, such that "uphill" moves are more likely to be accepted at the start of the optimisation, and less likely towards the end. In this way the exploration versus exploitation trade-off is made such that exploration dominates initially, and exploitation dominates at the end.

Taking inspiration from the physical process of thermal annealing, the acceptance probability is usually determined by:

$$p(\text{accept}) = \exp\left(-\frac{f(x') - f(x)}{T}\right)$$

where T is the "temperature", which is "cooled" as the algorithm progresses, such that uphill moves become less likely.

Quantum annealing

Quantum annealing is set-up in the following way:

- We have an optimisation problem which is specified in terms of a Hamiltonian whose ground-state is the optimal solution. This constitutes a perfectly natural way to define optimisation problems in general, and we denote this the "final" Hamiltonian, H_F, as the goal of the quantum annealing is to converge on a ground-state thereof.
- We have a *transverse field* Hamiltonian, H_D, that does not commute with H_F.
- Starting in an arbitrary initial state (conventionally a uniform superposition), we evolve a system according to:

$$\mathrm{H}(t) = \mathrm{H}_F + \Gamma(t)\mathrm{H}_D$$

  where $\Gamma(t)$ is the *transverse field coefficient*, which is initially very high, and reduces to zero over time.

Quantum annealing and adiabatic quantum computing

Quantum annealing essentially performs an adiabatic evolution to optimise some function, however there is a clear distinction:

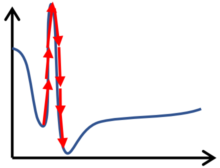
- In adiabatic quantum computing, a system is initialised in an easy to prepare ground-state, and the Hamiltonian is evolved slowly, so that the system remains in a ground-state. The ground-state of the final Hamiltonian encodes the solution of the problem of interest.
- Quantum annealing is a metaheuristic, which starts in an *arbitrary* initial state. The transverse Hamiltonian explores the optimisation surface, eventually converging upon the ground-state of the final Hamiltonian. The system does not necessarily start in a ground-state, and isn't required to remain in a ground-state throughout the evolution, however results from adiabatic quantum computing can be used to theoretically bound the performance of quantum annealing.

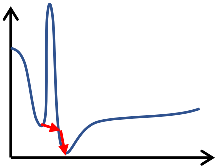
It follows that quantum annealing can readily be implemented on an adiabatic quantum computer, but it is also possible to construct purpose-built quantum annealers, but these are not universal quantum computers.

Quantum tunnelling

Simulated annealing performs poorly on optimisation surfaces with "high, narrow peaks", because simulated annealing can be visualised as exploring the surface by "walking over the top of it" and such a steep hill-climb is needed to escape a local minimum, that it is unlikely to be accepted.

By contrast, quantum annealing is better analogised as "tunnelling through peaks", and it is therefore the width not height of peaks which hinder its ability to escape local minima, and "high, narrow peaks" do not cause a problem.


Simulated annealing


Quantum annealing

It follows that quantum annealing can perform much better when optimising certain functions, to which simulated annealing would otherwise be well-suited.

D-Wave

- D-Wave Systems is a company that builds quantum annealers.
- In 2011, they announced *D-Wave one*, which they claimed to be the world's first commercially available quantum computer.
- They have shipped systems with 2048 qubits, and recently unveiled a 5640 qubit quantum annealer.

www.theverge.com

Overview of how D-Wave works

D-Wave uses quantum annealing to solve a single “native” optimisation problem. The problem in question is the optimisation of an instance of the *Ising model*, that is the minimisation of

$$f(x) = \sum_{i} h_i x_i + \sum_{i<j} J_{ij} x_i x_j$$

by adjusting $\{x\}$, where $x_i = \pm 1$ for all i. Note that h_i and J_{ij} are fixed parameters that define the instance of the Ising model that D-Wave solves natively.

Use of the Ising model is commonplace in statistical mechanics, and optimisation thereof is (the optimisation problem equivalent of) NP-hard, therefore to solve an arbitrary optimisation problem using D-Wave it is necessary to perform the following two (typically efficient) steps:

1. Map the optimisation problem of interest to the optimisation of some instance of the Ising model.
2. Map this instance of the Ising model to the instance that runs natively on D-Wave.

Quantum annealing and machine learning

Even though many experts were initially skeptical about the claims of D-Wave, as quantum computing in general matures as a field, increasing numbers of researchers are engaging with D-Wave, and one promising line of inquiry is quantum annealing for machine learning.

- Optimisation is a crucial sub-routine common to almost all machine learning tasks. It remains to be seen whether quantum annealing is especially well suited to some machine learning optimisation tasks, but if it were then this would potentially be a very significant application of quantum annealing, given the general importance of machine learning in contemporary computer science.
- Training of classifiers in one of the fundamental applications of supervised learning, and various researchers have run classifier training algorithms directly on D-Wave, with promising results.

Optimisation on gate-based quantum computers

Optimisation is ubiquitous in engineering and operations research, and there exist gate-based quantum algorithms for optimisation. The most famous of these is the Quantum Approximate Optimisation Algorithm (QAOA). After VQE it is arguably the most promising candidate for a quantum algorithm to show quantum advantage on some useful application in the NISQ era.

- Just as adiabatic quantum evolution can be discretised to allow adiabatic quantum algorithms to be run on gate-based quantum computers, so can quantum annealing: at its heart QAOA is a Trotterised version of quantum annealing.
- However, a full Trotterisation may mean that we incur an infeasibly deep circuit (because of the high noise present in near-term hardware), therefore QAOA is a variational algorithm which aims to learn the best Trotterised Hamiltonian evolution for a maximum number of steps.

Summary

In this lecture we have looked at:

- The adiabatic quantum theory.
- Adiabatic quantum computing as a universal computing model that is polynomially equivalent to gate-based quantum computing.
- Quantum annealing.
- QAOA
- D-Wave.