Lecture 3: Overview of computational complexity
Reading: Hamiltonian complexity, T. Osborne, Reports on Progress in Physics (2012).

1 Computational complexity

With this proof of principle discussion in hand, now is a good time to discuss the computational complexity of executing tasks on classical and quantum computers, which will then tell us which problems we can expect to solve on quantum computers. From the above discussion, it is clear that simulating the dynamics of a physical quantum system is efficient on a quantum computer in that it requires a polynomial number of gates in the simulation duration. Let's make this concept more precise and expand the discussion to various other tasks like finding ground states.

We first define Hamiltonian complexity as the resources required to simulate a physical system. This term and the associated study of the computational complexity of physical simulations originally arose when a connection was realized between the equilibrium states of many-body problems and the satisfying assignments of variables for certain logic problems. For instance, the decision problem associated with finding the ground state of a classical Ising spin glass is NP-complete, meaning that any combinatorial optimization problem in NP can be mapped to this physical system. [1]

With this link in mind, let's go over the classical complexity classes. A general schematic we will refer to is given in Fig. 1 [2]. A concept we need first is that of a deterministic or non-deterministic Turing machine. Since this class is not about computer science I will give a very rough overview of this topic. A deterministic Turing machine is one that executes one command based on what it reads in a memory tape. A non-deterministic Turing machine is an imaginary machine that can execute multiple commands based on what it reads as specified by a transition function. Such a machine generates a tree of commands that grows exponentially as each command is implemented.

There are a few important classical complexity classes. First, P is the complexity classes containing all decision problems that can be decided in polynomial time on a deterministic Turing machine. It contains problems such as linear programming, finding greatest-common divisors, and so on. Problems in this class are regarded as those that can be efficiently solved. A second important classical complexity class is BPP - the class of decision problems that can be solved in polynomial time using a probabilistic Turing machine with error less than $1/3$ (an arbitrary number that should be less than $1/2$). The idea is that repeating the algorithm with enough repetitions will yield the correct solution with probability approaching unity. P is a subset of BPP, and together these classes describe problems that have an efficient classical solution.

NP is the class of decision problems for which the solution can be verified in polynomial time. However, finding the solution to a problem in the class is a problem regarded as generically "hard" in that the best known algorithms require exponential time to solve. The Merlin-Arthur class (MA) can be regarded as the equivalent of BPP for P: a probabilistic version of NP where the solution can be checked using a probabilistic non-deterministic Turing machine with error less than $1/3$. Finally, the complexity class #P is a counting class for which the problem is to count the number of solutions to a given problem in NP. As a solution must be found to count it, problems in #P are at least as hard as those in NP and are usually harder. A number of important problems in physics are in #P, in particular the problem of calculating expectation values for a tensor network in greater than one dimension [3].

Here is an example of the mapping between an optimization problem and a local Hamiltonian ground state search. A common optimization problem is known as 3SAT, where SAT stands for satisfiability. The goal is to satisfy clauses (logical AND) that consist of a disjunction of literals

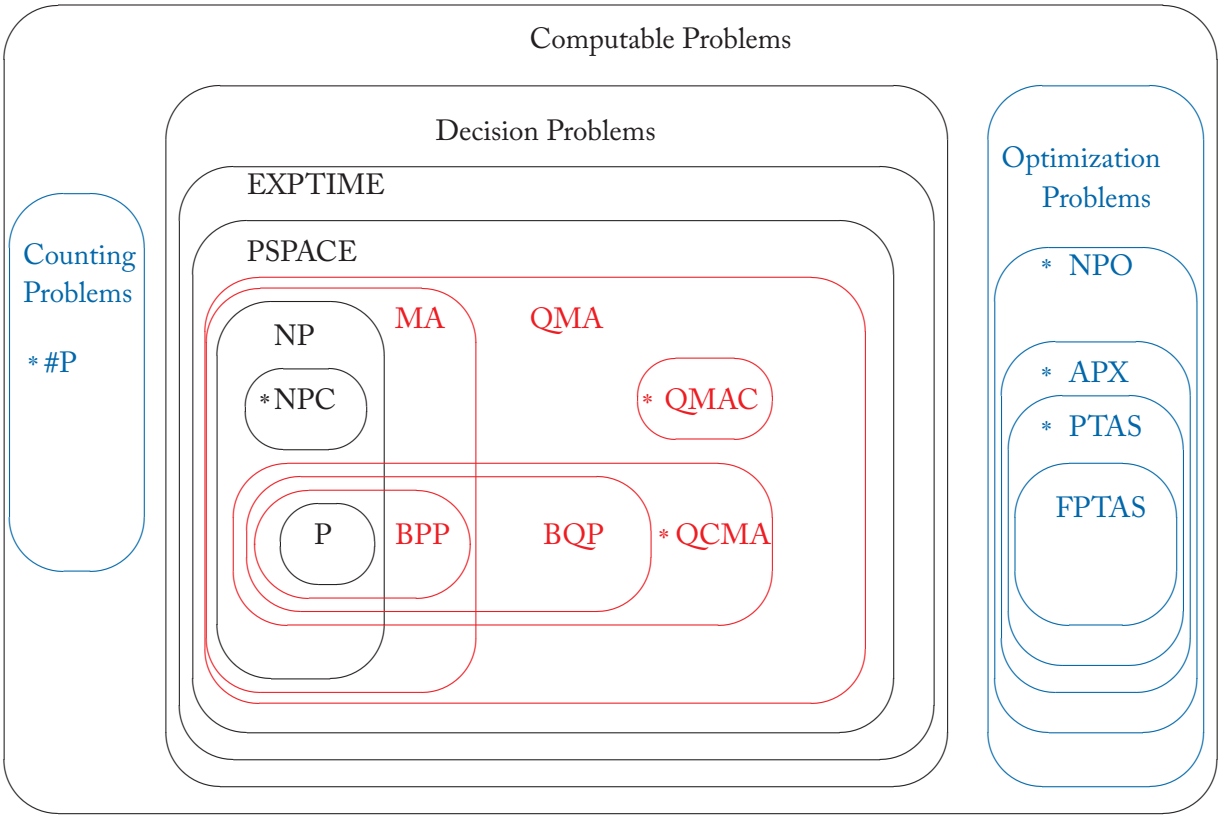

Figure 1: Schematic of classical and quantum complexity classes. [2] C at the end of the complexity label refers to "complete".

(OR). For instance, a valid clause could be $C_j(x) = \bar{x}_{j_1} \vee x_{j_2} \vee x_{j_3}$ and $f = \wedge_{j=1}^{m} C_j(x)$ where $\vee$ is logical OR, $\wedge$ is logical AND, and $\bar{x}$ denotes NOT x.

We can associate a classical Hamiltonian to this optimization problem as $H(x) = -\sum_{j=1}^{m} h_j(x)$ where each local term is the product of the x variables interpreted as Ising spins: $h_j(x) = \bar{x}_{j_1} x_{j_2} x_{j_3}$. Minimizing this function is equivalent to having all the clauses satisfied as in the original optimization problem. NOTE: I am not sure that this equality is actually correct (although that is what is in the Osborne paper). I am sure that 3SAT can be mapped to the maximal independent set problem which is equivalent to the set packing problem, which then has a Hamiltonian given in Ref. [1].

The Cook-Levin theorem tells us that 3SAT is NP-complete. Therefore, it is correspondingly unlikely that we can find the minimum of our physical Ising spin problem also! Although this result would suggest that all spin problems lack efficient solutions, that turns out not to be the case: some spin models like the ferromagnetic Ising model can be solved.

Aside: limitations of classical probabilistic computers

The following question may arise in your mind: given that quantum mechanics is probabilistic in the sense that we can only predict probabilities rather than absolute outcomes of any given experiment, why can we not use a classical probabilistic computer to simulate quantum systems? In this scenario, we would simulate a quantum process by using transition rules to describe the probability of obtaining various outcomes. Stringing these transition events together with the associated probabilities would be a way to simulate a classical probabilistic process. Why won't it work for quantum systems which are also probabilistic?

Feynman gives a nice explanation in his lecture as to why this particular scheme is not possible [4]. Briefly, the measurement outcomes on a quantum system cannot be simulated by any classical probabilistic computer due to entanglement. If you analyze what a classical analysis would predict the (probabilistic) outcome of a two-photon correlation experiment to be, it does not agree with the experimental value (classical theory predicts at most 2/3 of the time we agree on the polarization of the photon but the experiment says 3/4). It does, however, agree nicely with quantum theory that includes the full Hilbert space of the two photons (with two polarizations for a Hilbert space of dimension 4) and considers entangled photons emitted by an atom. Therefore, it is the effects of entanglement that prevent us from using a classical probabilistic computer in this manner to simulate a quantum system.

However, it is in fact possible to solve some quantum problems using classical probabilistic computers in a different way. This family of methods is generally known as quantum Monte Carlo and works very well for interacting bosonic systems. However, for fermionic systems which require an antisymmetric wavefunction, we quickly run into the problem that in the process of converting the quantum problem to a classical one that can be stochastically solved, we encounter nonpositive semidefinite weights - the sign problem. Solving the corresponding classical problem is NP-hard! [5]

1.1 Quantum complexity classes

The above discussion shows a quantum computer would be very helpful to simulate some quantum systems. We now discuss the quantum analogs of the classical complexity classes that will tell us which calculations quantum computers can carry out efficiently. First, let's more precisely define the decision problem associated with quantum simulation. To specify the problem we need to provide a Hamiltonian H, an initial state ρ_0, observables A, a (possibly complex) time t, and two proposed expectation values $\alpha_1 < \alpha_2$.

The quantum simulator should output YES if $\langle A \rangle \leq \alpha_1$ and NO if $\langle A \rangle \geq \alpha_2$, where $\langle A \rangle = Tr(\rho A)$

and

$$\rho = \frac{(e^{iHt})^\dagger \rho_0 e^{iHt}}{Tr((e^{iHt})^\dagger \rho_0 e^{iHt})} \tag{1.1}$$

If t is real, this simulation corresponds to the usual real time simulation. If t is purely imaginary, we have imaginary time evolution that eventually leads to the ground state. t could be neither purely real or imaginary - that case corresponds to a system at finite temperature evolving in real time.

First, BQP is the class of decision problems (e.g. the simulation problem above) that can be solved in polynomial time on a quantum computer with error probability less than 1/3. As discussed in the previous lecture, simulating quantum dynamics can be implemented in polynomial time on a QC, and relevant questions can be formulated as decision problems. As such, simulating quantum dynamics is in BQP.

The quantum analog of probabilistic NP, or MA, is QMA. It is the class of decision problem with solution that can be checked on a QC with error probability less than 1/3. Unfortunately, many relevant problems for physics are contained in this complexity class, meaning that QCs cannot in general solve them efficiently. An important example is the problem of finding the ground state of a local Hamiltonian, which is more precisely stated as determining whether the smallest eigenvalue of a Hamiltonian is greater or larger than some value up to some tolerance. This problem is QMA-complete for $k \geq 2$ where k is the locality of the terms composing the Hamiltonian [6]. Many other fundamental problems are in this class, including finding the universal functional for density functional theory [7], the N-representability condition for finding an electron density consistent with a valid wavefunction [8], and many others [9].

The result indicates that, just as in classical computation, we will need to make approximations or exploit some physical knowledge to solve problems even on a QC. What we have going for us with a quantum computer is the ability to represent the entire Hilbert space with a polynomial number of qubits as well as an ability to manipulate and interrogate this quantum information. Compared to classical computation, the exponential memory requirement is lifted for a QC. However, another problem is related to extracting useful information from a QC, as determining the full wavefunction of the set of qubits scales exponentially with the number of qubits. Therefore, for efficient determination of relevant properties we are restricted to choosing measurement of certain observables that we hope provide the physical information we seek.

1.2 Limited summary of relevant complexity classes

Here is a summary of the complexity classes most relevant for this class and the relationships between them.

- P - decision problems that can be solved in polynomial time on a classical computer deterministically
- BPP - bounded error probabilistic polynomial time: decision problems that can be answered with a probabilistic classical computer with error of at most 1/3 (e.g. if the answer is YES, it returns YES with probability $> 2/3$)
- BQP - bounded error quantum polynomial time: the same idea as BPP, with corresponding error constraints, but with a quantum computer.
- NP - nondeterministic polynomial time: a prover can convince a verifier of a provided solution in polynomial time.

- PSPACE - problems for which a classical algorithm can verify an answer with polynomial space (memory)
- Merlin-Arthur (MA) - verifier = Author can use a proof provided by Merlin to probabilistically verify a solution on a classical computer with the usual error restrictions.
- QMA - MA using a quantum algorithm (loosely, the quantum version of NP)
- #P - counts the number of solutions for an NP problem (not a decision class).
- EXPTIME, EXPSPACE, NPSPACE, NEXPSPACE - you get the idea.

Some relations:

- $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$
- NPSPACE = PSPACE by Savitch's Theorem

References

[1]A. Lucas, "Ising formulations of many NP problems", Frontiers in physics **2** (2014) 10.3389/fphy.2014.00005.

[2]C. C. McGeoch, "Adiabatic quantum computation and quantum annealing: theory and practice", Synthesis Lectures on Quantum Computing **5**, 1–93 (2014).

[3]N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, "Computational complexity of projected entangled pair states.", Physical Review Letters **98**, 140506 (2007).

[4]R. P. Feynman, "Simulating physics with computers", Int J Theor Phys **21**, 467–488 (1982).

[5]M. Troyer and U.-J. Wiese, "Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations.", Phys Rev Lett **94**, 170201 (2005).

[6]J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local hamiltonian problem", in *FSTTCS 2004: foundations of software technology and theoretical computer science*, Vol. 3328, edited by K. Lodaya, M. Mahajan, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum, Lecture notes in computer science (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004), pp. 372–383.

[7]N. Schuch and F. Verstraete, "Computational complexity of interacting electrons and fundamental limitations of density functional theory", Nature physics **5**, 732–735 (2009).

[8]Y.-K. Liu, M. Christandl, and F. Verstraete, "Quantum computational complexity of the n-representability problem: QMA complete.", Physical Review Letters **98**, 110503 (2007).

[9]A. D. Bookatz, "QMA-complete problems", arXiv (2012).