P versus NP and Quantum Computing

Javier Gamarra
Luce Innovative Technologies
University of Valladolid
Email: nhpatt@gmail.com

***Abstract*—We present the state of the art in the P versus NP problem and the characteristics and basic operations of quantum computers. We resume the quantum algorithms found by Shor and Grover and expose the status of P vs. NP question in the light of quantum computing.**

I. Introduction

Since Cook introduced in 1971 the "P versus NP" problem [1] hundreds [2] of computer scientists have tried to solve what is considered to be one of the most important problems in the field. It is one of the seven Millenium Prize Problems, rewarded with 1 million dollars by the Clay Institute [3]. Mathematicians around the world have tried to solve it using different techniques and, since Feynman started researching in quantum computing, there is the persistent belief that quantum computers can improve drastically the performance of NP problems.

In this paper we briefly present the P versus NP problem along with the characteristics of quantum computers in section III. In section IV we expose the quantum algorithms and the effect in the P vs. NP problem. Finishing with the future of quantum computing in section V.

II. P versus NP

A. Description of the problem

Informally, the class P, or just P, is composed by the problems that can be 'quickly' solved by a computer. 'Quickly' refers to a solution that runs in polynomial time, increases proportionally with the size of the input. "Class P (are) the problems solvable by an algorithm with a number of steps bounded by some fixed polynomial in the length of the input" [3].

A polynomial time solution can be computationally expensive, given the magnitude of the constants and the factor affecting the input. It is referred as 'quickly' comparing it against exponential algorithms.

The class NP is composed of the problems that can be verified 'easily' (quickly), in polynomial time. A P problem is contained in NP (we can solve the problem in polynomial time, checking its solution). "Every problem whose solution can be quickly verified by a computer (NP)" [3].

For example the Knapsack problem (given a X Kg bag and some items, Can a value of at least V be achieved without exceeding the weight W?) is NP (easily to test) but we don't know if it is quickly solvable (in P) [4].

There are some extra interesting complexity classes like NP-complete. This class is composed by equivalent problems, those who can be expressed like they were other, as hard, problem. A NP-complete problem can be transformed into another NP-complete problem (without losing information) in polynomial time. If someone finds a polynomial answer to one of those problems, he could solve all at the same time. There are hundreds of NP-complete problems, the knapsack problem or the traveling salesman are examples of NP-complete problems. NP-complete problems are also presented in nature, like:

1) Finding a DNA sequence that best fits a collection of fragments of the sequence.
2) Finding Nash Equilibriums with specific properties in a number of environments.
3) Determining if a mathematical statement has a short proof

There are also problems that are neither in P nor NP-complete. They are called NPI (NP-intermediate problems). The integer factorization problem is believed to be one of them. It can not be solved in polynomial time nor be reduced to a NP-complete problem.

A positive answer to P = NP would mean that all the easily-to-verify problems are also quickly solved. A negative answer would mean that there are some problems easily verifiable but hard to solve.

Figure 1 shows the complexity classes, along BQP who will be introduced later, in a box called P-SPACE (the problems solved in polynomial space).

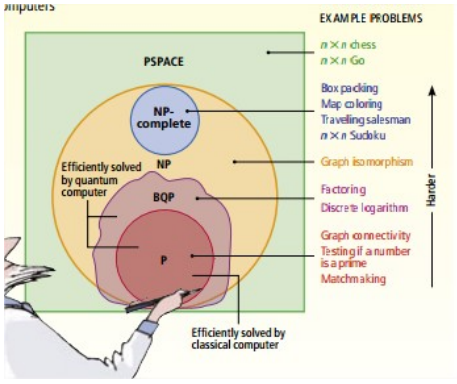

Figure 1. Quantum complexity classes, assuming P!=NP

B. Relevance

Mathematicians have studied P vs. NP in the last 40 years and it seems we aren't close to a proof yet. Although it seems

it was the right question to ask [5]. Along the way we have discovered lower bounds that can anticipate how a proof of $P \neq NP$ will be: it will have to contain all those lower bounds as special cases.

Along those years, scientists have used and discarded some techniques:

1) Diagonalization: can we construct an NP language L designed so that every polynomial algorithm fails to compute L on some input? [6].
2) Circuit Complexity: try to show some NP-complete problem can not be solved by small circuits of logic gates.
3) Relativization: "relativizing proofs can only prove statements that are uniformly true with respects to all possible oracles" [4], they can not be used to solve P = NP.
4) Algebrization: Aaronson [5] proved that algebrization can not be use to solve P = NP.
5) Natural Proofs: Razborov proved that natural proofs alone can not resolve P = NP.

If a proof to P = NP exists it would mean that a lot of 'hard' problems in routing, logistics, cryptography and optimization are able to be solved in less time (depending on the factors). Everything would be more efficient [6] (supposing a P algorithm is found).

It would make verification and finding of a mathematical proof viable for a computer (for example, the other five Millenium problems). Automatic proofs could be generated. The result would kill creativity for many philosophers, as checking a problem could be as hard as solving it.

If $P \neq NP$ was found, it would mean that there are really hard problems that can be easily verified. And Science could 'move on'.

C. State of the problem

All those years of investigation seems to indicate that $P \neq NP$.

One argument (not very convincing) is [7] that nobody has yet found a polynomial-time algorithm after all those years.

Gasarch [8], in a poll to 100 scientists in 2002 concluded that the main opinion was $P \neq NP$ followed by 61 scientists against 9 that thought that $P = NP$ (22 offered no opinion).

Peter Shor said, in that regard: "Possibly the right question for practice is whether NP is contained in BPP or BQP (my response to both of these questions is no".

Asked about when will be answered, 57 said after 2020 and 5 never, 17 between 2002 and 2019.

Fortnow tried to find if P versus NP is formally independent [9] and concludes that if exists a proof it would require new mathematical techniques.

Although it seems to be a way of P = NP [10] by having nonlinear quantum mechanics. Abrams proposed a way to solve NP-complete problems in polynomial time using nonlinear quantum logic gates. But nonlinear time evolution or nonlinear quantum mechanics is at present hypothetical, it also suffers from having theoretical difficulties like superluminal communication or violating the Heisenberg principle [10]. Abrams believes that quantum mechanics is exactly linear so we are not close to P = NP in this way too.

Another way to achieve P = NP is time travel, specifically closed timelike curves (CTCs) [11]. A CTC is a path through space where time can travel along to create a close loop, without paradoxes. Current physical theory is inconclusive on whether CTCs can exist, but if they were possible, we could solve exponential algorithms instantly, sending us back the results of the computation.

Computer scientist have some hope of solving P = NP with Geometric Complexity Theory [6]. This approach needs extensive mathematics and could need decades of work.

The question, then, remains unsolved.

III. Quantum Computers

A quantum computer uses quantum physics and quantum mechanical phenomena to perform operations on data [4]. A theorical model to represent a computer with quantum properties is the quantum Turing machine (QTM), the quantum equivalent of the Turing machine.

A. Characteristics

Digital classical computers store information in bits, a binary representation with two states, 0 or 1. Quantum computers store information in qubits, that can be in multiple states simultaneously.

A single qubit can represent a one or zero or any quantum superposition of those qubit states. A quantum computer with n qubits can be in any superposition of up to 2^n states.

Figure 2 shows the graphical representation of a qubit.

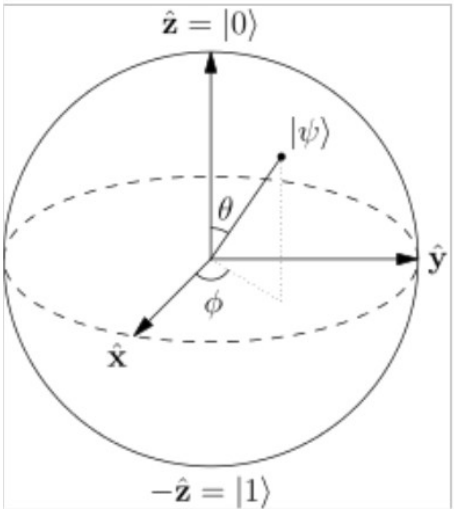

Figure 2. Graphic representation of a qubit

Large scale quantum computers will be able to store huge volumes of information, a quantum computer with 1000 qubits could store 10^{300} bits. They will be able to solve specific algorithms way faster than a classic computer, using quantum algorithms like Shor or Grover.

It is incorrect to think that qubits are at the same time in multiple states. That is correct only in a probabilistic superposition, each state has some probabilistic possibility. The act of measuring affects the outcome, there will be only a final configuration when the qubit is measured.

A classical computer with 2 bits can be in one of 4 states (00,01,10,11), it is a deterministic computer in the sense that, after initializing it and without external interference, it can only be in one of those states, with probability 1. A quantum computer, however, after initialing it, could be in state 00 with $0.8 + 2i$ probability and $0.6 + 3i$ in the other states. The sum of the squares of those probabilities have to sum 1. The phase between any state represents a meaningful parameter [4].

B. Computation

Quantum model of computation consists in quantum logical gates, unitary matrix (rotations). Since rotations can be inverted, quantum computing is reversible. When the algorithm finishes the result has to be read. When you measure the qubits, the state is collapsed to a classic state. That state has some probability associated but, by repeating the algorithm, the probability of getting the right answer increases.

The quantum algorithm starts setting the initial state (a hard problem itself) of the qubits and then letting them through a series of quantum logic gates. The effect of the gates in the information is known but the qubits can be in a superposition of states before and after the gates. When the algorithm finishes we measure the result, collapsing the state to a static representation.

C. Problems

The biggest problems in physical implementations of quantum computers are errors and decoherence [12].

The errors made by a quantum computer propagate and can destabilize the computer. There have been remarkable progress in fault-tolerant quantum computation [13], although error correction codes doesn't work so well as expected, because of the extra storage and processing speed needed. In classical computers, error correction algorithms have to preserve the binary state whereas in quantum computers the information is stored by a superposition of states.

Decoherence is the effect produced by the environment interacting with the qubits, that damage the state of the information. This effect is irreversible. Decoherence times range between nanoseconds and seconds at low temperature. It is crucial that quantum operations run in substantial less time than decoherence time.

Removing decoherence its only possible by isolating the computer from its environment. If the error rate is small, quantum error correction algorithms can also prevent error produced by decoherence.

The reason large-scale quantum computers doesn't exist yet are because the difficulty of controlling quantum systems [13].

D. Computing models

There are four basic models of quantum computers [4]:

1) Quantum Gate Array: computation based on sequences of few qubits quantum gates.
2) One-Way Quantum Computer: computation based on measurements applied to high entangled initial states.
3) Adiabatic Quantum Computer: based on Hamiltonian paths whose ground states contains the solution.
4) Topological quantum computer: computation decomposed intro the braiding of anyions in a 2D lattice.

All four proposed models and polynomial equivalents, meaning it is possible to represent one model with another one, just performing polynomial transformations.

David DiVicenzo [4] list some requirements needed for building a practical quantum computer:

1) Scalable physically to increase the number of qubit.s
2) Qubits can be initializaed to arbitrary values.
3) Quantum gates faster than decoherence time.
4) Universal gate set.
5) Qubits can be read easily.

E. Hardware implementations

There are a lot of ways of implementing small quantum computers, this field has exploded since Feynman proposed quantum computing.

We list some technologies pursued:

1) Superconductor based quantum computers
2) Optical lattices.
3) Nuclear magnetic resonance on molecules in solution.
4) Optics-based quantum computer.

In 2009 [4], researches at Yale created the first solid-state quantum computer with 2 qubits.

There are claims of commercial quantum computers with 128 qubits chipsets.

There are systems with the order of 10 qubits working and systems with of order 100 qubits under way [13].

F. Other uses

Apart from implementing efficient algorithms and storing huge amounts of data, there are other field where quantum computers could be very useful.

Quantum cryptography is a viable field, theoretical models have been proposed and physical implementations are on the way. Some studies [13] show physical limitations with quantum key distribution. Quantum computers could be used as exact clocks [13].

Figure 3 lists quantum disadvantages and benefits.

IV. P versus NP and quantum computing

A. Complexity classes

When quantum computing appears the number of complexity classes grows, with the important appearance of QMA and BQP. BQP bounded error quantum polynomial time) contains P and is the quantum equivalent of P, in the sense that BQP algorithms can be resolved in polynomial time with a quantum computer. QMA (Quantum Merlin Arthur) is the quantum equivalent of NP. If P = NP, QMA, NP and P would be the same class.

Table 1. *Some possible objections to quantum computation, and some responses*

objection	response
Quantum computers are analogue devices, and hence cannot be restandardized.	Using quantum error-correcting codes and fault-tolerant error correction, we *can* restandardize encoded quantum information.
A quantum error-correcting code cannot detect or correct an error if the quantum state remains inside the protected code subspace.	If good codes are used, such uncorrectable errors are quite unlikely.
A high-performance quantum computer must prepare and maintain highly entangled quantum states, and these states are very vulnerable to decoherence.	Fault-tolerant error correction protects highly entangled encoded states from decoherence.
Error correction itself requires a complex computation, and so is bound to introduce more errors than it removes!	If the error probability per gate is below the *accuracy threshold*, then an arbitrarily long computation can in principle be performed with negligible probability of error.
Quantum error correction will slow down the computer.	With highly parallelized operation, the slowdown need not be serious.
To successfully incorporate quantum error correction, a much larger quantum computer will be needed.	The number of qubits needed increases only polylogarithmically with the size of the computation to be performed.
Any quantum computer will suffer from leakage—the quantum information will diffuse out of the Hilbert space on which the computer acts.	With suitable coding, leakage errors can be detected and corrected.
Systematic errors will accumulate over time; error-correcting codes do not deal with systematic errors as effectively as with random errors.	In principle, systematic errors can be understood and eliminated.
Coding does not protect against highly correlated errors.	Correlated errors can be suppressed with suitable machine architecture.
There are intrinsic limitations on the accuracy of quantum gates. Error correction will not work for gates of feasible accuracy.	Within the known limits, gates exceeding the accuracy threshold are possible. With suitable hardware, even these limits might be evaded.
Current quantum computing technology is inaccurate, slow, not scalable, and not easily parallelizable.	Faster gates and new ways to distribute entanglement can surely be developed. New ideas for quantum hardware would be most welcome!
In the near term, experiments with quantum computers will be mere demonstrations. They will not teach us anything.	We will learn about correlated decoherence. The performance of devices with just a few tens of qubits cannot be easily simulated or predicted.
Quantum computers will be too expensive.	But they will be worth the price if suitably broad applications are found.
The known applications of quantum computers are quite limited.	Let's think of new ones! The known quantum database search algorithm may prove to be very useful.

Figure 3. Quantum pros and cons

The scientific community since [14] believes that BQP is larger (and contains) P but it is unproven yet [15]. BQP should be compared with BPP (probabilistic polynomical-time algorithms) since quantum algorithms are in general probabilistic. Integer factorization and discrete logarithms are believed to be in BQP.

In the same way, Aharonov proved [16] that the 3-SAT problem (NP-complete) is QMA complete.

Assuming $P \neq NP$ Figure 4 represents the relation between the most important complexity classes [17].

There are over 400 complexity classes [11] and the study of them is an area itself in computer science. There are more questions unsolved in this area.

B. Algorithms

Since Feynman introduced quantum computing in 1982 [18] there have been mayor discoveries in quantum algorithms with the objective of using the special characteristics of quantum physics to improve the performance of classical computers.

Shor classifies the quantum algorithms in three large groups [19]

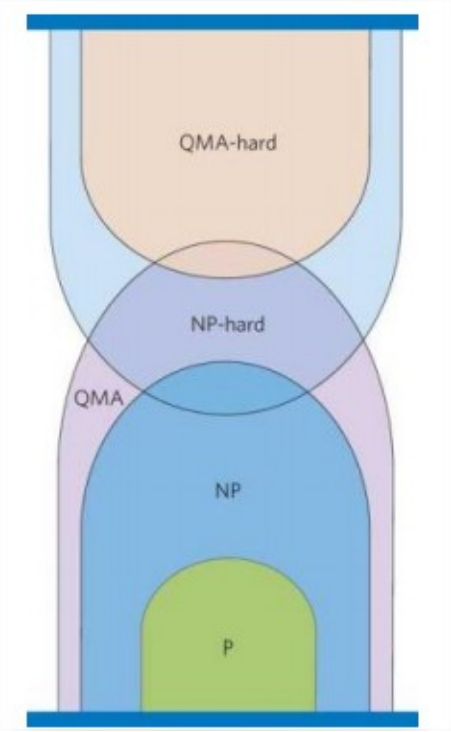

Figure 4. Quantum complexity classes, assuming P!=NP

1) Periodicity algorithms: they use the Fourier transform to locate some type of periodicity in the numbers and exploit it (white box algorithm). The typical example is the prime factorization algorithm.
2) Grover's search algorithm. A type of black box algorithm that can perform exhaustive search of N items in $\sqrt{N}$
3) Quantum algorithms: simulate real quantum physics, Feynman original idea belongs to this group.

We will explain each type in algorithm in detail.

1) Factorization Algorithm: in 1997 Peter Shor exposed a way to factor integers and find discrete logarithms in polynomial time with a quantum computer [14].

This is considered a mayor breakthrough. Integer factorization is a NP problem (verifying the result means multiplying the suspected factors and comparing) but is not believed to be a NP-Complete Problem or have a polynomial algorithm in classical computers. The best algorithm belongs to Lenstral [20] and takes exponential time depending of the logarithm of the input.

Shor presented a polynomial algorithm in a quantum computer, raising the expectations of the scientific community in finding a polynomial solution to NP-Complete problems. The proposed algorithm uses Fourier's transformation to find specific periodicity in the numbers. It is a white box algorithm, in the sense that uses specific properties of the problem and can not be reproduced in other generic problems. Its order is N^3

Public cryptography is based on factoring integers (using

them as one-way functions), factoring is supposed to be a hard problem and, with Shor algorithm, quantum computers could broke all those secret systems way faster.

However, Shor said: "In the history of computer science (...) most important problems have turned out to be either polynomial-time or NP-complete. (...) quantum computers will likely not become widely useful unless they can solve NP-Complete problems" [14].

The belief of quantum computers being able to confirm P=NP problems was supported by this paper, because of the erroneous belief that integer factorization is a NP-Complete problem. The misconception that quantum computers can solve NP-complete problems is more widespread in press and non-scientific authors (see for example the issue of Economist of February 15, 2007).

2) Grover's search: In 1996, Grover [21] introduced a method for searching an unsorted database with a quantum computer. The proposed method could find an item in a set with N items in $\sqrt{N}$ time. The speed-up is achieved by exploiting quantum parallelism and the relation between probability in quantum theory and amplitude. This means that doing a blind search in all of the possible solutions is cheaper in a quantum computer but it is not an exponential speed-up.

Bennet [22] showed that Grover's algorithm is optimal, meaning that no quantum algorithm can solve it faster (black box algorithm). Without knowing the internal structure, Grover gives us a upper bound in speed-ups, meaning that NP-complete problems only can be solved in polynomial time by exploiting an internal property of their structure.

3) Quantum physics: The original Feynman idea is simulate quantum systems with a quantum computer (because of his huge storing capabilities). Simulating quantum physics may require exponential resources for a classic computer and polynomial for a quantum computer.

Preskill says [13]: "Classical systems cannot in general simulate quantum systems efficiently. We cannot yet prove this claim, either mathematically or experimentally, but we have reason to believe it is true; arguably, it is one of the most interesting distinctions ever made between quantum and classical.".

In [13] there are other super polynomial speedups known, like simulating topological QFT (quantum field theory) or computing properties of solutions to systems of linear equations. Gaussian quantum dynamics is also easy to simulate [13].

Jordan [23] and [24] can solve QFT in polynomial time depending of the number of particles and they expect to generalize it to theories that involve fermions and the Standard Model of particle physics.

C. Feasability

Bennett [22] proved that there is an oracle chosen uniformly at random with probability 1 the class NP cannot be solved on a QTM in exponetial time. That proves that there is not a black box approach to solving NP-complete problems using quantum computers.

Shor argues [19] that since his discovery of the quantum factoring algorithm in 1994 there haven't been any progress in finding new important algorithms. Although there have been a lot of new physical implementations and progress in quantum cryptography, since 1995 no new algorithms have been discovered.

This lack of progress seems to mean two things:

1) Quantum computers are a paradigm shift in computer science and scientists can't keep the pace.
2) There aren't many problems that can be improved drastically with quantum computing. Scientists have been looking for superpolynomial speed-ups because of the relevance of the P vs. NP problem (and small speed are less glamorous and attract less people).

To achieve a super polynomial speed-up you can't look in P (polynomial solution), the first class of problems not in P are NP-complete ones but the search hasn't been fruitful. The other type of problems not NP-complete are the ones which are neither NP-hard nor in P. Those ones (like the integer factorization) can not be reduced to each other, so they are only susceptible to white-box algorithms. These include the problems of graph isomorphism and approximating short vectors in a lattice, yet unresolved.

Quantum computers may not be efficient even to simulate some problems of quantum physics, Schuch exposes in Nature [25] that simulating the Hubbard's model is a QMA problem.

There are doubts about specific implementations of quantum computers and NP-Complete problems. The scientific community have argued about adiabatic quantum optimization will fail in NP-Complete problems, for example. However Dickson [26] studies the maximum independent set (MIS) problem and shows that there is always a Hamiltonian path without crossing the local minimums.

Blier [27] introduces a new complexity class $PQMA\log(2)$ (languages with logarithm-size quantum proof) and proves that equals NP. This approach is yet to be proven and confirmed.

V. Future of quantum computing

There seems to be little progress regarding quantum algorithms solving NP-complete problems in polynomial time. Shor argues about scope and suggests computer scientists to focus on smaller improvements to existing algorithms instead of trying to prove or disprove P = NP. He says that there is a lot of room for exciting discoveries in improving existing algorithms. Solving NP-complete problems in polynomial time seems out of reach.

Grover's algorithm impose a upper bound in quantum black box algorithms but it does not mean that a better white-box algorithm exists somewhere. To be able to get super polynomial speed-ups that algorithm has to use specific internal properties of NP-complete or NP hard problems, in the same way that Shor algorithm uses periodicity in integer factorization.

There are also other NP-Intermediate problems without a quantum polynomial solution. Because they can not be reduced to equivalent problems they have to be studied on their own.

Simulating quantum physics seems a promising field in quantum computers, there have been important progress [23] in that area and is a perfect reason to build a physical computer. Simulating efficiently physics have many real applications and can help scientists the improve the knowledge of the rules governing our environment.

Another good reason to build a physical large computer is using the already found speed ups in integer factorization and exhaustive search. Those results alone can revolutionize the Internet world. Public key cryptography as we know will be flawed when large quantum computers are build. Grover's search algorithm effectively reduces the key to a half, transforming 1024 key algorithms in breakable 512 ones. Shor's result is even more important in that regard, large quantum computers could factor integers in polynomial time, rendering private keys useless. It is important to notice that there are alternatives to public key cryptography not based on integer factorization and one way functions, which the scientist community believe exists, could fill the gap left by integer factorization too.

Quantum cryptography has taken the lead and shows promising theorical and practical results. On paper seeems unbreakable and being able to replace integer factorization. There are some limitations regarding physical distance but repeaters can solve those practical problems.

Quantum physical implementations are a thriving field, both academical and commercial small solutions are appearing [28]. The field is in its infancy so there is a lot of room for exciting discoveries.

VI. Conclusion and future work

We have presented the status of the P vs. NP problem, unsolved since 1971. It is regarded as the most important problem in computer science and attracts amateur mathematicians and computer scientists hoping to earn the million Clay award.

There are hundreds of papers about the subject, making it hard for a non mathematician to differentiate the real ones from the unscientific ones (see [29] or the family of papers by ohya [30] [31] [32]). P vs NP has permeated to popular cultural, appearing in TV shows and being the subject of movies [33]. In this regard, we think this article presents an up-to-date state of the art in the subject and its relation with quantum computers.

We argue against the common belief, outside scientific circles, of quantum computers solving NP-complete problems in polynomial time and expose the quantum algorithms discovered.

There is room for improvement, we haven't talk in detail about quantum complexity classes, mainly because the mathematical knowledge required to be able to judge the quality of a paper. The quantum model of computing is presented in brief detail as it is hard to comprehend for a computer scientist used to the classical model.

Quantum computing is a new paradigm with great investigation potential. Although scientist do not think we will achieve super polynomial speed-ups in NP-complete problems, quantum computers can improve drastically human knowledge in physics.

Meanwhile, P = NP will be still a mystery.

References

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, (New York, NY, USA), pp. 151–158, ACM, 1971.

[2] G. J. Woeginger, "The p-versus-np page." http://www.win.tue.nl/~gwoegi/P-versus-NP.htm. Accessed: 13/01/25.

[3] S. Cook, "The p versus np problem,"

[4] Wikipedia, "Quantum computer." http://en.wikipedia.org/wiki/Quantum_computer. Accessed: 13/01/25.

[5] S. Aaronson, "Has there been progress on the p vs. np question?." http://www.scottaaronson.com/talks/pvsnp.ppt. Accessed: 13/01/25.

[6] L. Fortnow, "The status of the p versus np problem," *Communications of the ACM*, vol. 52, no. 9, pp. 78–86, 2009.

[7] P. Shor, "Progress in quantum algorithms," *Experimental Aspects of Quantum Computing*, pp. 5–13, 2005.

[8] W. Gasarch, "The p=? np poll," *Sigact News*, vol. 33, no. 2, pp. 34–47, 2002.

[9] L. Fortnow and S. Aaronson, "Is p versus np formally independent?," *Bulletin of the European Association for Theoretical Computer Science*, vol. 81, pp. 109–136, 2003.

[10] D. Abrams and S. Lloyd, "Nonlinear quantum mechanics implies polynomial-time solution for np-complete and# p problems," *Physical Review Letters*, vol. 81, no. 18, pp. 3992–3995, 1998.

[11] S. Aaronson, "The limits of quantum computers," *Scientific American*, vol. 298, no. 3, pp. 62–69, 2008.

[12] J. Preskill and J. Preskill, "Quantum computing: pro and con," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 469–486, 1998.

[13] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv preprint arXiv:1203.5813*, 2012.

[14] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[15] D. Simon, "On the power of quantum computation," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pp. 116–123, IEEE, 1994.

[16] D. Aharonov and T. Naveh, "Quantum np-a survey," *arXiv preprint quant-ph/0210077*, 2002.

[17] F. (th)E mule, "El estado actual del problema p dis tinto de np." http://francisthemulenews.wordpress.com/2009/09/05/el-estado-actual-del-problema-p-distinto-de-np/. Accessed: 13/01/25.

[18] R. Feynman, "Simulating physics with computers," *International journal of theoretical physics*, vol. 21, no. 6, pp. 467–488, 1982.

[19] P. Shor, "Why haven't more quantum algorithms been found?," *Journal of the ACM (JACM)*, vol. 50, no. 1, pp. 87–90, 2003.

[20] J. Lenstra, D. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical programming*, vol. 46, no. 1, pp. 259–271, 1990.

[21] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, ACM, 1996.

[22] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, 1997.

[23] S. Jordan, K. Lee, and J. Preskill, "Quantum algorithms for quantum field theories," *Science*, vol. 336, no. 6085, pp. 1130–1133, 2012.

[24] P. Hauke, L. Tagliacozzo, and M. Lewenstein, "Speeding up quantum field theories," *Science*, vol. 336, no. 6085, pp. 1122–1123, 2012.

[25] N. Schuch and F. Verstraete, "Computational complexity of interacting electrons and fundamental limitations of density functional theory," *Nature Physics*, vol. 5, no. 10, pp. 732–735, 2009.

[26] N. Dickson and M. Amin, "Does adiabatic quantum optimization fail for np-complete problems?," *Physical Review Letters*, vol. 106, no. 5, p. 50502, 2011.

[27] H. Blier and A. Tapp, "A quantum characterization of np," *arXiv preprint arXiv:0709.0738*, 2007.

[28] A. Harrow, "Why now is the right time to study quantum computing," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 18, no. 3, pp. 32–37, 2012.

[29] M. Freedman, "P/np, and the quantum field computer," *Proceedings of the National Academy of Sciences*, vol. 95, no. 1, pp. 98–101, 1998.
[30] M. Ohya and I. Volovich, "Quantum computing, np-complete problems and chaotic dynamics," *arXiv preprint quant-ph/9912100*, 1999.
[31] M. Ohya and I. Volovich, "New quantum algorithm for studying np-complete problems," *Reports on Mathematical Physics*, vol. 52, no. 1, pp. 25–33, 2003.
[32] M. Ohya and N. Masuda, "Np problem in quantum algorithm," *Open Systems & Information Dynamics*, vol. 7, no. 1, pp. 33–39, 2000.
[33] T. Lanzone, "Travelling salesman." http://www.travellingsalesmanmovie.com/. Accessed: 13/01/25.