Simulating Quantum Systems with Qubitization, Trotterization and Linear combination of unitaries (LCU)

Alexander Del Toro Barba (PhD) · Follow

10 min read · Sep 6, 2024

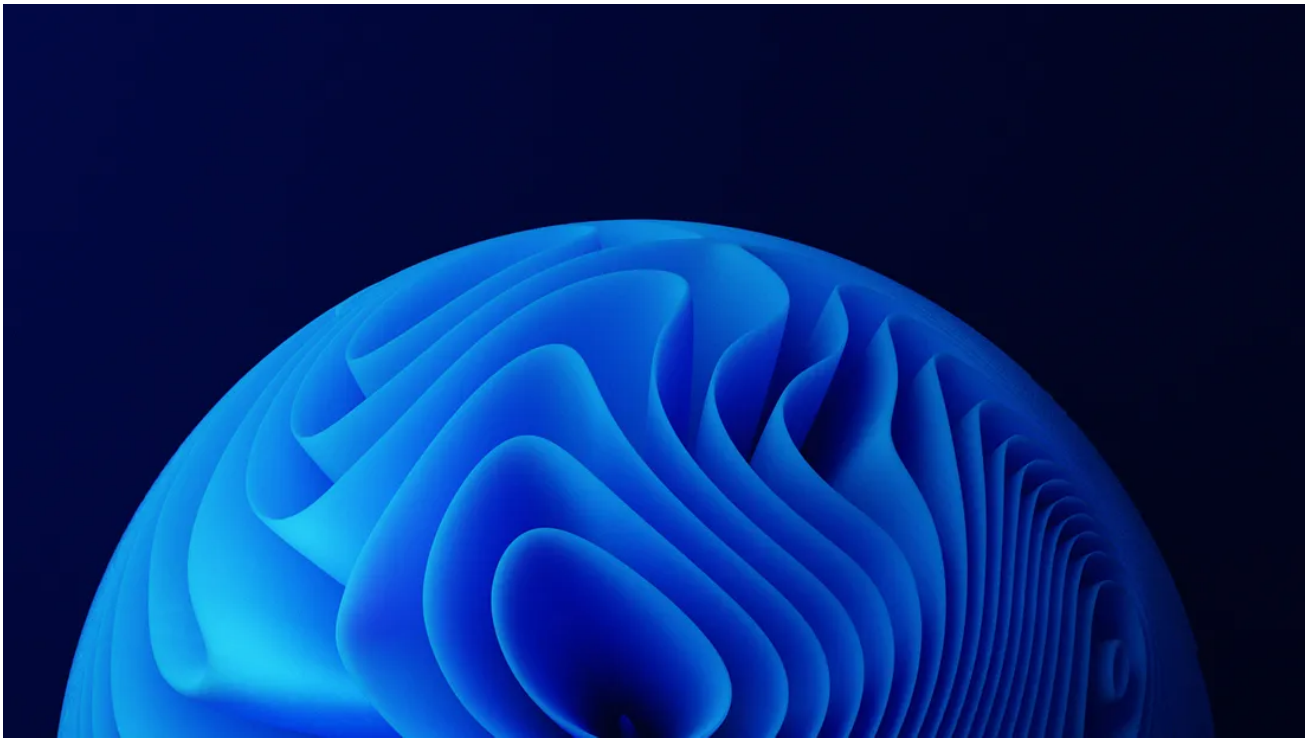


Photo by SIMON LEE on Unsplash

Quantum simulation is a critical application of quantum computing that promises to revolutionize fields such as chemistry, materials science, and physics by enabling the efficient modeling of quantum systems that are intractable for classical computers.

Many quantum systems, such as molecules and complex materials, have a large number of interacting particles, which makes their exact behavior nearly

impossible to simulate with classical methods due to the exponential growth in computational resources required.

Quantum computers leverage principles of superposition and entanglement, and can naturally represent and simulate these systems more efficiently. Accurate quantum simulations with methods like Linear Combination of Unitaries (LCU), Trotterization, and Qubitization, can help to discover new materials, efficient chemical reactions, and even advance our understanding of fundamental physics.

Linear combination of unitaries (LCU)

The LCU method allows a Hermitian matrix (like a Hamiltonian) to be expressed as a sum of unitaries, where U_i are unitary operators and c_i are real coefficients. The key idea is to represent the evolution of the system under this Hamiltonian using a weighted combination of these unitaries.

$$H = \sum_i c_i U_i$$

LCU is very **flexible** as it allows direct decomposition into unitaries and doesn't require small time steps, as in Trotterization. Additionally, unlike Trotterization, which approximates the evolution by breaking the Hamiltonian into small time steps, LCU can simulate the evolution exactly (up to the **precision** of the decomposition into unitaries).

However, LCU generally requires ancilla qubits and complex quantum circuits to handle the probabilistic nature of the linear combination. The algorithm must prepare a state proportional to the coefficients c_i and apply the corresponding unitaries, which can be resource-intensive.

LCU is particularly powerful for simulating Hamiltonians when Trotterization would require too many small steps for high accuracy. It is used in Hamiltonian simulation and algorithms like the Quantum Signal Processing (QSP) framework

and Quantum Phase Estimation (QPE).

One question often asked is: How does the algorithm select the most relevant unitaries and their coefficients from a potentially large basis (like Pauli operators I, X, Y, Z)? If we take a Hermitian matrix *H* that we want to decompose into a linear combination of Pauli operators:

$$H = \begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix}$$

that we want to decompose into a linear combination of Pauli operators *I, X, Y, Z:*

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

We can express *H* as a linear combination of Pauli matrices. This process is called **Expansion in a Basis (Pauli Operators):**

$$H = c_0 I + c_1 X + c_2 Y + c_3 Z$$

c_0, c_1, c_2, and c_3 are the coefficients to be determined. To find these coefficients, **we compute the trace (or inner product) of *H* with each Pauli operator.** This is a projection technique, which involves taking the inner product of the Hermitian matrix with each Pauli term to calculate how much “weight” each Pauli operator has in the expansion.

For a general 2 x 2 matrix *M*, coefficient for each Pauli matrix *P*:

$$c_P = \frac{1}{2}\mathrm{Tr}(MP)$$

The we compute coefficients c_0 for *I*, c_1 for *X*, c_2 for *Y*, and c_3 for *Z*:

$$c_0 = \frac{1}{2}\mathrm{Tr}(HI) = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix} = \frac{1}{2}(1+3) = 2$$

$$c_1 = \frac{1}{2}\mathrm{Tr}(HX) = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 2i & 1 \\ 3 & -2i \end{pmatrix} = \frac{1}{2}(2i+(-2i)) = 0$$

$$c_2 = \frac{1}{2}\mathrm{Tr}(HY) = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} -2 & i \\ 3i & 2 \end{pmatrix} = \frac{1}{2}(-2+2) = 0$$

$$c_3 = \frac{1}{2}\mathrm{Tr}(HZ) = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2}\mathrm{Tr}\begin{pmatrix} 1 & -2i \\ 2i & -3 \end{pmatrix} = \frac{1}{2}(1+(-3)) = -1$$

Not all Pauli terms are relevant or contribute significantly to the decomposition. The most relevant unitaries are selected based on:

- **Magnitude of coefficients**: Only Pauli operators with significant coefficients c_i are chosen. Many coefficients may be very small or zero, which can be ignored without affecting accuracy much.
- **Sparsity**: In practical applications in chemistry or condensed matter physics, many Hamiltonians of interest are relatively sparse. Only a small subset of Pauli terms have large coefficients. This sparsity helps limit the number of

terms to simulate.

Using the coefficients of our example, the original Hermitian matrix H

$$H = \begin{pmatrix} 1 & 2i \\ -2i & 3 \end{pmatrix}$$

can be reconstructed as a linear combination of Pauli matrices:

$$H = 2I - Z$$

The Pauli I matrix contributes uniformly to all entries of the matrix with a weight of 2, and the Pauli Z matrix affects only the diagonal elements, contributing with a weight of −1.

For large systems, the LCU algorithm can apply approximation techniques. It might use techniques like **Trotterization** to approximate the effect of the Hamiltonian by evolving the system under the most significant terms.

Some quantum algorithms (such as VQE) use optimization methods to adjust the coefficients iteratively. For instance, in VQE, a parameterized quantum circuit is run, and classical optimization algorithms are used to find the coefficients that minimize the energy of the system, effectively finding the most relevant Pauli terms for representing the Hamiltonian.

In general, the algorithm does not attempt to explore all possible combinations of Pauli operators. Instead, it leverages the structure of the problem (e.g., the sparsity and symmetry of the Hamiltonian) and uses classical preprocessing to narrow down the relevant unitaries.

Trotterization for Approximating Quantum Systems

Trotterization is an **approximation-based** method for simulating quantum evolution. If you have a Hamiltonian $H = A + B$, where A and B are non-commuting terms, Trotterization approximates the evolution operator e^{-iHt}, where n is the number of steps, and the error decreases as n increases:

$$e^{-iHt} \approx \left(e^{-iAt/n} e^{-iBt/n}\right)^n$$

Trotterization is relatively **simple** to implement and doesn't require ancilla qubits or complex procedures like LCU. Furthermore, it works **efficient** when the Hamiltonian can be decomposed into only a few terms with manageable step sizes.

However, the method introduces Trotter errors, which depend on the number of steps n used. To reduce these errors, smaller time steps and more repetitions are required, increasing the overall circuit depth.

Trotterization is often used in quantum chemistry and condensed matter simulations where the Hamiltonian can be naturally split into few terms and where low-order approximations suffice.

Qubitization using Quantum Walks

Qubitization is a more **advanced and efficient method** for Hamiltonian simulation that leverages **quantum walks** to perform quantum phase estimation more efficiently. It applies to Hamiltonians that can be expressed as a linear combination of unitaries (similar to LCU), but it avoids the high cost of LCU by constructing quantum walks that encode the Hamiltonian in a more resource-efficient manner.

Qubitization works by embedding the Hamiltonian in a higher-dimensional quantum walk operator that preserves the eigenvalues of the original Hamiltonian, allowing precise simulation without needing many time steps or ancilla qubits like LCU.

Qubitization achieves **optimal scaling efficiency** for Hamiltonian simulation, with the number of quantum gates scaling linearly in time tt and the norm of the Hamiltonian. Additionally, it provides **highly accurate**, exact simulation (up to discretization and phase estimation errors) and can be more resource-efficient than LCU.

However, while qubitization is highly efficient, it requires a sophisticated quantum walk construction and precise control over quantum circuits, making it more complex to implement than Trotterization. That is why, Qubitization is preferred for **large-scale, high-accuracy simulations of quantum systems**, especially in quantum chemistry, condensed matter, and physics simulations where long time evolution is needed without accumulating approximation errors.

Qubitization in Quantum Topological Data Analysis

In the quantum TDA algorithm, the (block-encoded) Dirac operator connects different simplices (e.g., vertices, edges, triangles) in the simplicial complex, and the goal is to isolate the eigenstates corresponding to zero eigenvalues. This means we need to project onto the subspace spanned by the zero eigenvalue states which represents Betti numbers. This is done by a quantum simulation. We will apply quantum walks constructed from Qubitization to efficiently simulate the evolution of the system.

We want to map the eigenvalues of the Dirac operator into a phase estimation problem on a quantum computer. The goal is to estimate the phases (eigenvalues) of a **quantum walk operator**. Zero eigenvalues in the Dirac operator B are mapped to ±1 eigenvalues in the quantum walk operator. The **quantum walk operator** acts on a quantum state and efficiently simulates the evolution of the system.

The Quantum TDA algorithm can provide an exponential speedup because it can **simulate the high-dimensional Dirac operator exponentially faster** than classical computers. The quantum walk operator explores the structure of the simplicial complex, where spectral properties of the quantum walk operator, i.e. its eigenvalues and eigenvectors, are tied to eigenvalues of the Dirac operator, which allows to compute Betti numbers.

In this **Qubitization** process, the quantum walk operator is constructed from the block-encoded, sparse **Dirac operator** V and a **reflection operator** P which inverts the phase of certain components of a quantum state:

$$W = RV$$

Qubiterate operator or quantum walk operator W

The **reflection operator** R is used to construct the quantum walk operator, where $|0\rangle$ is a specific quantum state, often computational basis state with all qubits in state 0. I **is identity operator**. The reflection operator R acts on a quantum state by identifying the component of the state that overlaps with the state $|0\rangle$. Then it inverts the phase (multiplying by -1) of this component, and leaves all other components unchanged:

$$R = 2|0\rangle\langle 0| - I$$

Reflection operator R

In quantum TDA, a **specific projection operator** P has been included, which ensures that the reflection operation acts only on relevant subspaces of the Hilbert space, determined by topological properties. The qubitization and subsequent quantum walk operate within correct subspaces, which leads to an efficient and accurate estimation of the Betti numbers:

$$R = i(2|0\rangle\langle 0| \otimes P - I)$$

Reflection operator R in Quantum TDA

The reflection helps to **amplify the desired components** of the quantum state, which leads to a more efficient and accurate estimation of the Betti numbers. This brings us to the complete formula for the quantum walk W:

$$W = i\,(2|0\rangle\langle 0| \otimes P - I)\,V$$

Qubiterate operator or quantum walk operator W

The projection operator P ensures that the state remains in the relevant subspace and V "qubitizes" the sparse matrix and applies the operator efficiently in a quantum circuit. The eigenvalues of this quantum walk operator are related to the eigenvalues of the original Dirac operator.

- **Ancillary Qubit** is introduced to $|0\rangle$
- **PREP Operator** creates a superposition over the ancillary qubit, which prepares the system for qubitization.
- **SELECT Operator** conditionally applies the block-encoded unitary U, effectively performing a **quantum walk** on the system.
- **Rotation-based Quantum Walk** is applied during qubitization based on state of the ancillary qubit. The rotations correspond to time evolution under Hamiltonian H, resulting in unitary operation e^{-iHt}.

The eigenvalues of this quantum walk operator W are directly related to the eigenvalues of the original Dirac operator B. If $|\psi\rangle$ is an eigenstate of B with eigenvalue E, then the corresponding eigenvalues of W are:

$$\pm e^{\pm i \arcsin(E_k/\lambda)}$$

Unitary operation after rotations corresponding to time evolution under Hamiltonian H

The quantum walk is applied repeatedly which helps to efficiently simulate unitary time evolution e^{-iHt} of the original system as governed by Dirac operator. It contains the eigenvalues that we want to estimate, e.g. using quantum phase estimation or Chebychev polynomials. The quantum walk focuses on simulating relevant dynamics within a carefully chosen subspace (Kernel) that replicates key behaviors of original system.

The quantum walk operator *W* is applied to the initial state $|0\rangle|\psi\rangle$ resulting in a superposition of two states. The first state is the original state with a phase proportional to the energy E. The second state is an orthogonal state, representing a “failure” in the sense that it lies outside the desired subspace (Kernel). The amplitudes of these two states depend on the ratio of the energy E to the normalization factor λ.

$$W|0\rangle|\psi_k\rangle = i\frac{E_k}{\lambda}|0\rangle|\psi_k\rangle + \sqrt{1-\left(\frac{E_k}{\lambda}\right)^2}|0\rangle|\psi_k^{\perp}\rangle$$

- **W** is the qubiterate operator (the quantum walk operator).
- $|0\rangle$ is the initial state of the ancilla qubit used in block encoding.
- $|\psi\rangle$ is an eigenstate of the Hamiltonian (in this case, the Dirac operator B) with energy E.
- λ is a normalization factor associated with the block encoding.
- $|\psi\perp\rangle$ is a state orthogonal to $|0\rangle$ on the ancilla qubit and also orthogonal to the desired subspace (defined by the projector P) on the system register.

The spectral properties of quantum walk operator *W* are directly tied to eigenvalues and eigenvectors of the Dirac operator *B (*Hamiltonian *E)* through a trigonometric relationship. The transformed eigenvalues encode important properties about the connectivity in simplicial complexes. By estimating the eigenvalues of the quantum walk, we can infer information about the kernel of Dirac operator used to compute Betti numbers.

This article is part of my series on quantum topological data analysis.

Disclaimer: *The views and opinions expressed in this article are my own and do not*

Quantum Computing ce Topological Data Analysis

Follow

Written by Alexander Del Toro Barba (PhD)

124 Followers · 10 Following

Machine Learning and Quantum Computing at Google Cloud | Google Scholar: https://scholar.google.com/citations?user=fddyK-wAAAAJ&hl=de

No responses yet

Write a response

What are your thoughts?

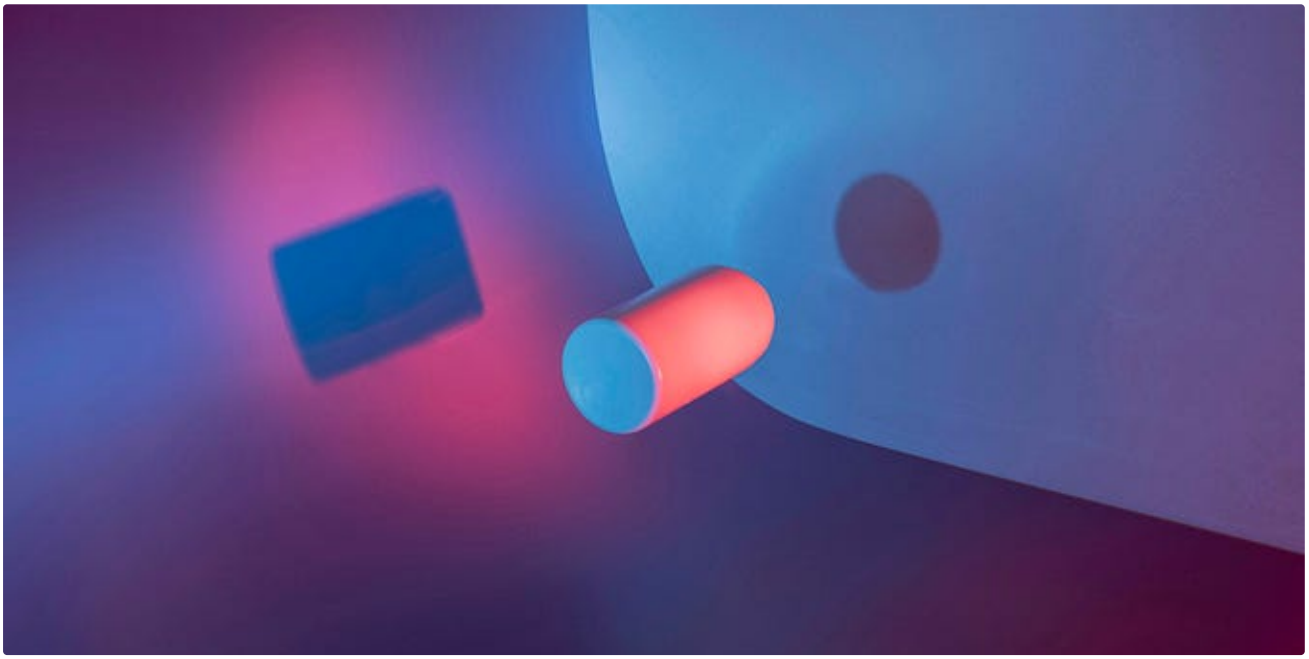
More from Alexander Del Toro Barba (PhD)

Alexander Del Toro Barba (PhD)

Topological Data Analysis with Persistent Homology

Quantum TDA is a very promising quantum algorithm that resists dequantization. In part 1 we learn how to identify topological invariants in

Aug 7, 2024 4



Alexander Del Toro Barba (PhD)

Can Quantum Computing accelerate Generative AI?

Alexander Del Toro Barba (PhD)

The Many Worlds of Quantum-Inspired

Quantum-inspired algorithms run on classical hardware, but use methods to imitate quantum-mechanical effects such as superposition and...

Feb 5 3

Alexander Del Toro Barba (PhD)

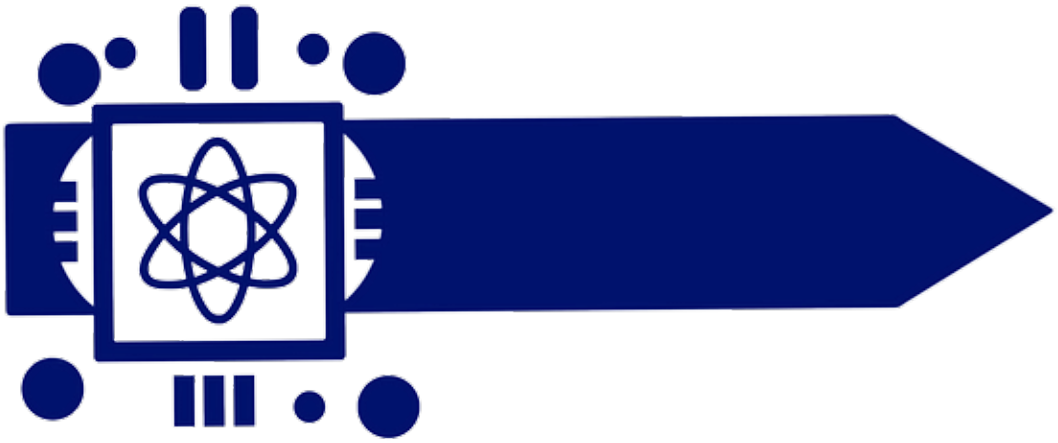
Topological Data Analysis with Spectral Analysis of Combinatorial Laplacians

Quantum TDA is a very promising quantum algorithm that resists dequantization. In this part we learn how to identify topological...

Aug 9, 2024 3

See all from Alexander Del Toro Barba (PhD)

Recommended from Medium



Shubhransh Rai

Quantum Diary #1—How I'm planning to self learn Quantum Physics and Computing in 1 year

Yes, two of the most difficult subjects in the entire world and my overconfident ass decides "Hm, wouldn't it be cool if i learnt it, all...

Jan 7 26 1

Alexander Del Toro Barba (PhD)

Block Encoding in Quantum Computing

The Dirac operator connects different simplices (e.g., vertices, edges, triangles) in the simplicial complex, and the goal is to isolate...

Oct 5, 2024 1

Lists

Staff picks

821 stories · 1642 saves

Stories to Help You Level-Up at Work

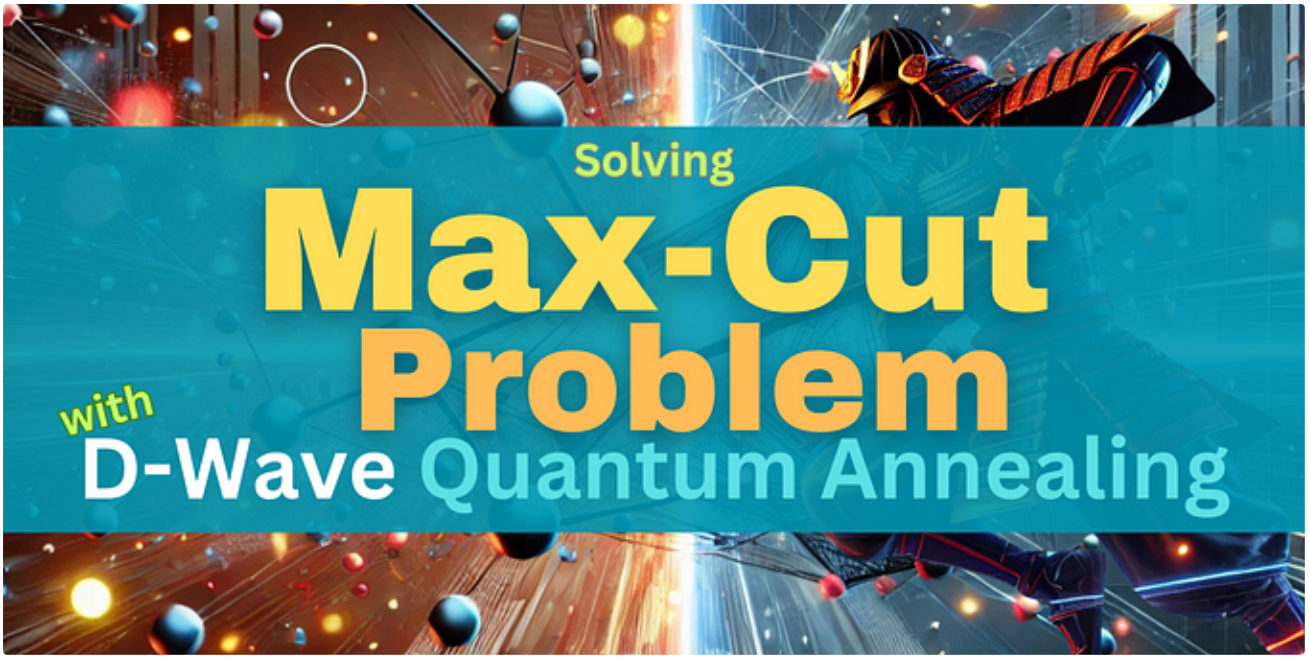
19 stories · 945 saves

Self-Improvement 101

20 stories · 3336 saves

Productivity 101

20 stories · 2803 saves



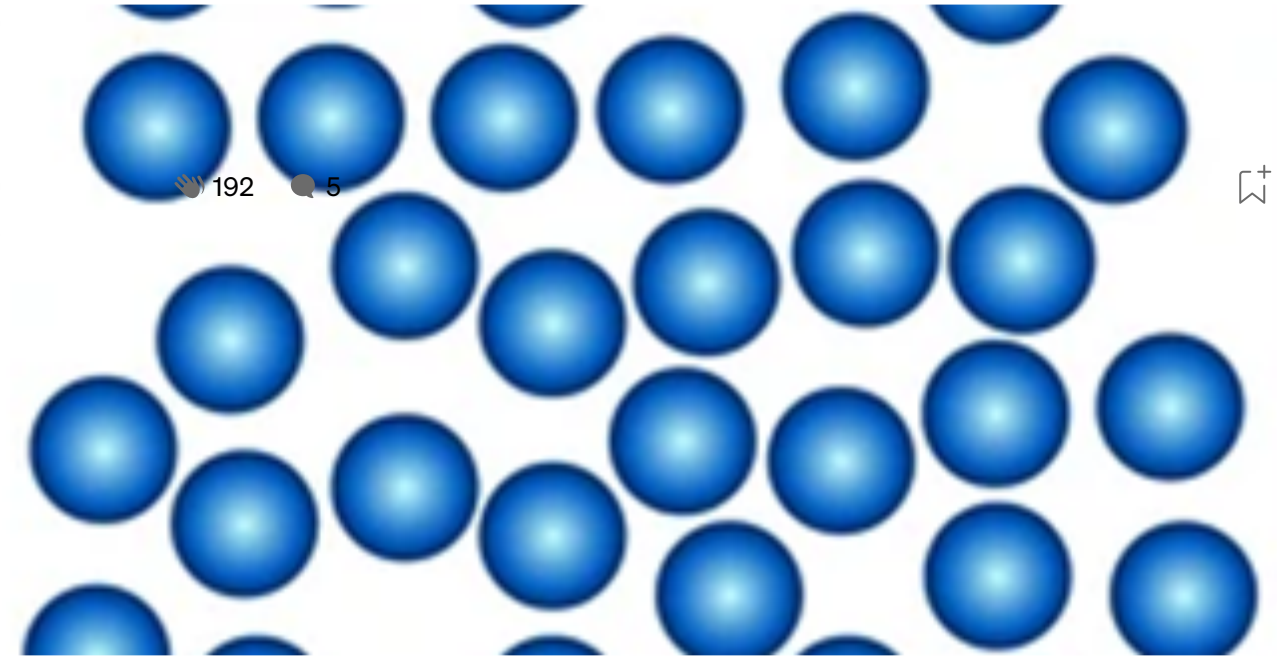
Naoki

Solving Max-Cut Problems with D-Wave Quantum Annealing

Split the Network for Maximum Gain!

Nov 24, 2024

John Watrous

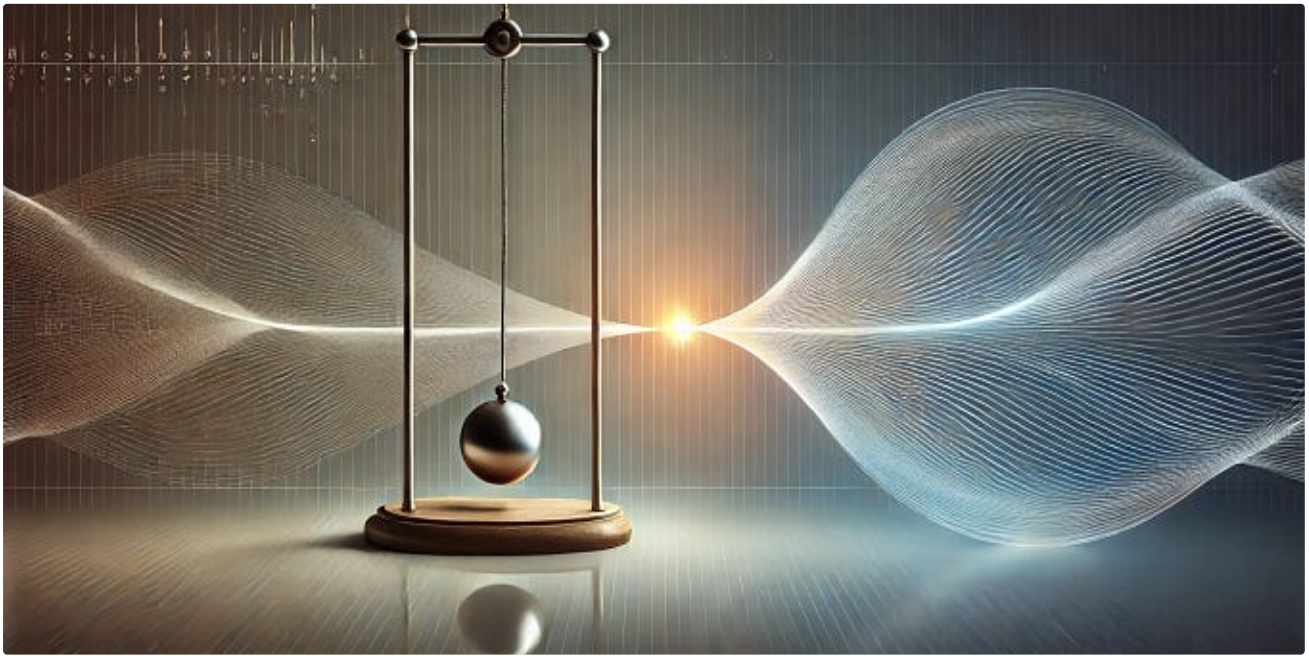


Aurelien Pelissier

It Took Me 10 Years to Understand Entropy, Here is What I Learned.

From the Big bang to the Heat death of the universe

Apr 17, 2022 1.8K 24



Arun

The Evolution of the Hamiltonian: From Newton to Quantum

Imagine standing at the edge of a vast ocean, where the waves are made not of water, but of energy, particles, and probabilities. Beneath...

Jan 25 51 1 See more recommendations