Quantum Dynamic Programming

Jeongrak Son, Marek Gluza, Ryuji Takagi, Nelly Ng

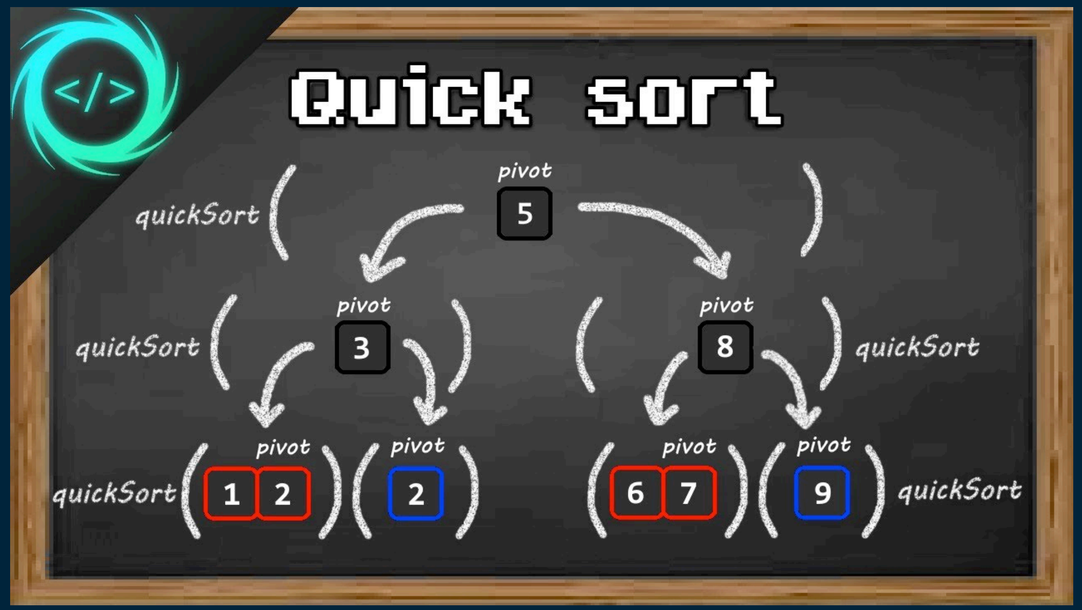
Nanyang Tech Uni, Singapore Uni Tokyo, Japan

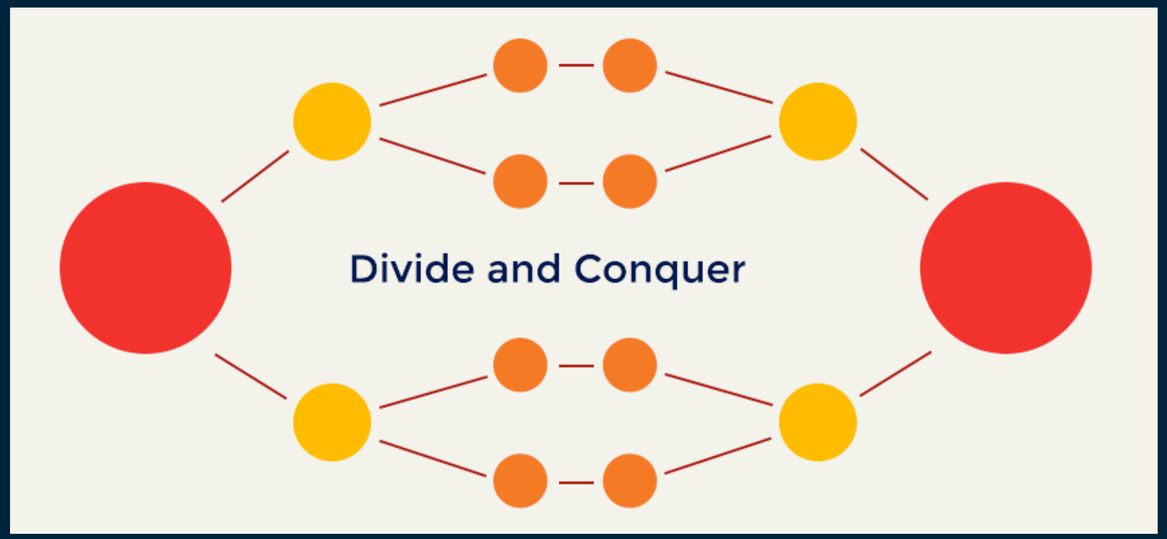
arXiv:2403.09187

Outline

- Classical dynamic programming and their role in algorithms
- The challenge of quantum recursions
- QDP as a resolution to quantum recursions
- Applications and outlook

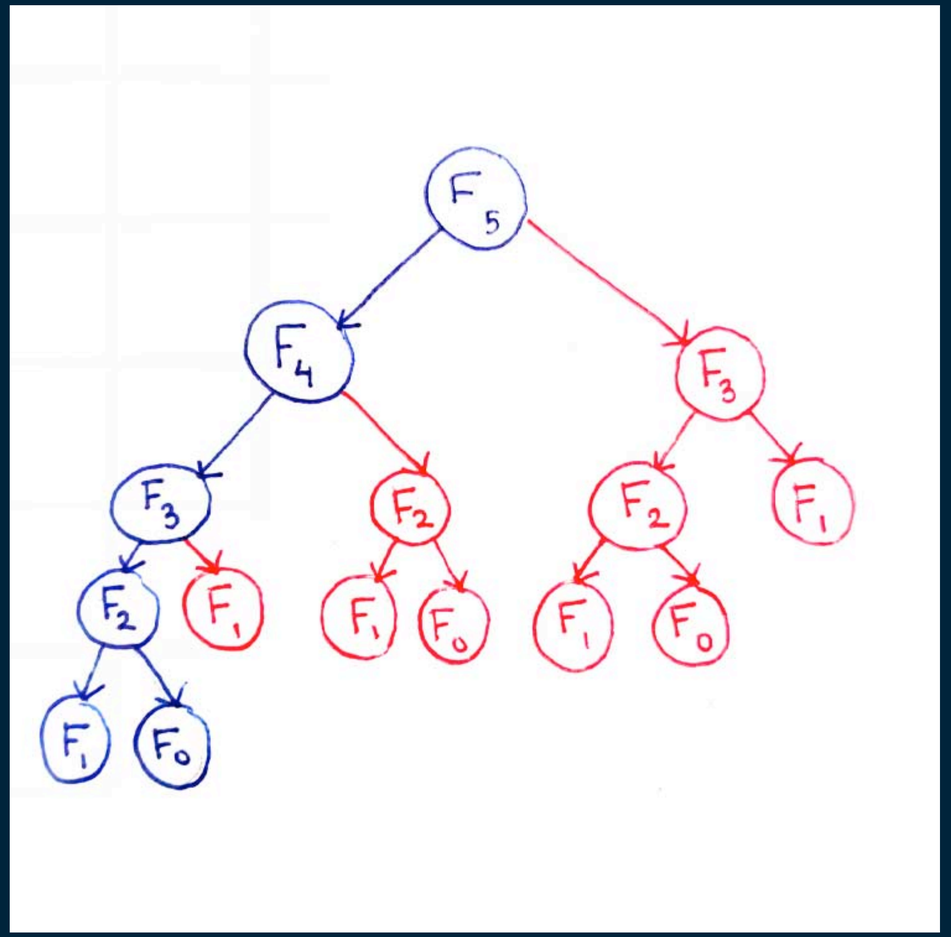
Examples of classical recursive algorithms





Classical Dynamic Programming

- The breaking down of a complex problem into **subproblems**, and **storing the corresponding solutions** to increase computational efficiency
- Ex: calculate $F(n)$, the n-th number in the Fibonacci sequence, defined as $F(n) = F(n-1) + F(n-2)$, with $F(0) = F(1) = 1$
  - Naive computation (without memory) requires the computation of $F(n-1)$ and $F(n-2)$, every time a Fibonacci number is called (both blue and red nodes need to be computed upon use)
  - Dynamic computation (with memory) computes each $F(n)$ only once, stores in an internal memo, and calls it the next time it is needed.

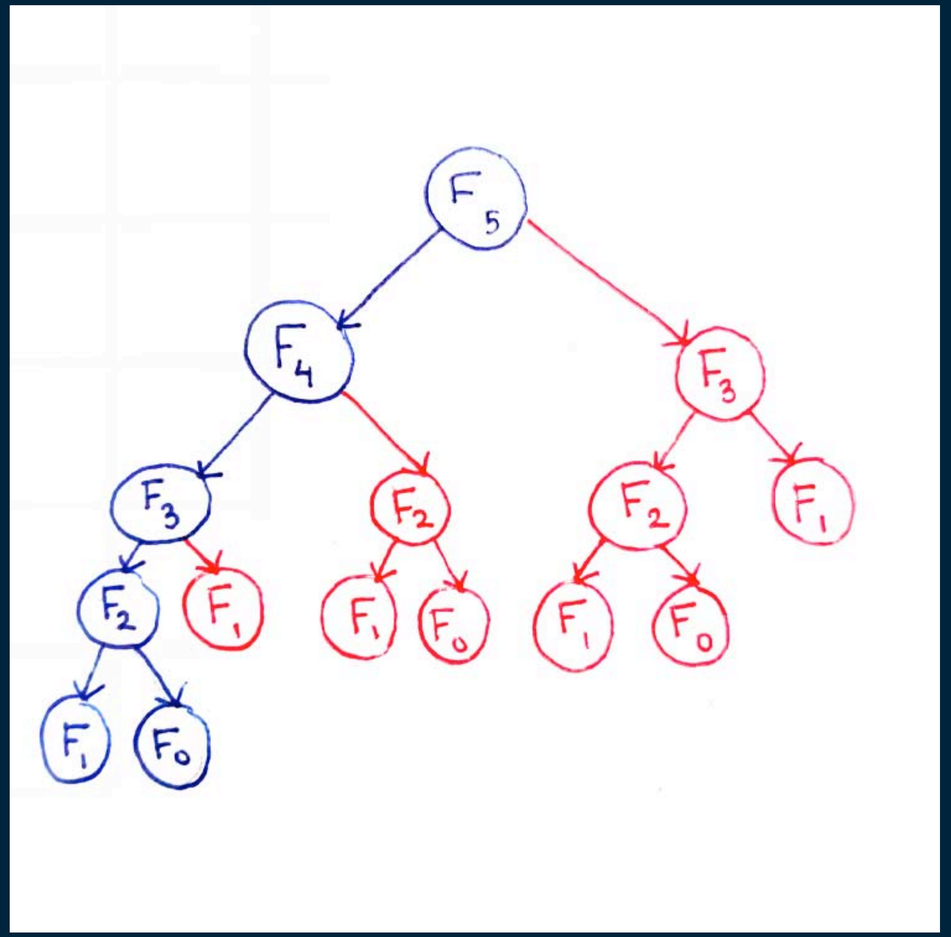


Classical Dynamic Programming

- The breaking down of a complex problem into **subproblems**, and **storing the corresponding solutions** to increase computational efficiency
- Ex: calculate $F(n)$, the n-th number in the Fibonacci sequence, defined as $F(n) = F(n-1) + F(n-2)$, with $F(0) = F(1) = 1$
  - Naive computation (without memory) requires the computation of $F(n-1)$ and $F(n-2)$ every time a Fibonacci numb… need to be comp…
  - Dynamic computation (with memory) computes each $F(n)$ only once, …it the next time it is…

Computational steps required : $O(2^n)$

Computational steps required : $O(n)$!
At the cost of $O(n)$ memory…



Quantum recursions

$$\rho_n = \hat{U}^{(\rho_{n-1})} \rho_{n-1} (\hat{U}^{(\rho_{n-1})})^\dagger$$

Assumptions:

- The n_{th} recursive step $\hat{U}^{(\rho_{n-1})}$ is **unitary** and depends on the **previous result** ρ_{n-1}.
- Initial state ρ_0 and $\hat{U}^{(\rho_0)}$ are given (not necessarily known)
- The exact form of $\hat{U}^{(\rho)}$ as a function of ρ is known

Quantum recursions

$$\rho_n = \hat{U}^{(\rho_{n-1})} \rho_{n-1} (\hat{U}^{(\rho_{n-1})})^\dagger$$

Assumptions:

- The n_{th} recursive step $\hat{U}^{(\rho_{n-1})}$ is **unitary** and depends on the **previous result** ρ_{n-1}.
- Initial state ρ_0 and $\hat{U}^{(\rho_0)}$ are given (
- The exact form of $\hat{U}^{(\rho)}$ as a functio

Can we even perform a quantum computation on this naively? We need $\hat{U}^{(\rho_1)}$, $\hat{U}^{(\rho_2)}$, … all the way up to $\hat{U}^{(\rho_{n-1})}$, but we don't really know the intermediate states!

Examples of quantum recursions

1. Grover search $|\psi\rangle \mapsto e^{i\beta_L \psi} e^{i\alpha_L \tau} \cdots e^{i\beta_1 \psi} e^{i\alpha_1 \tau} |\psi\rangle \sim |\tau\rangle$

   Nested formulation:

$$|\psi_0\rangle \mapsto |\psi_1\rangle = e^{i\beta_L^{(0)} \psi_0} e^{i\alpha_L^{(0)} \tau} \cdots e^{i\beta_1^{(0)} \psi_0} e^{i\alpha_1^{(0)} \tau} |\psi_0\rangle$$

$$|\psi_1\rangle \mapsto |\psi_2\rangle = e^{i\beta_L^{(1)} \psi_1} e^{i\alpha_L^{(1)} \tau} \cdots e^{i\beta_1^{(1)} \psi_1} e^{i\alpha_1^{(1)} \tau} |\psi_1\rangle$$

$$\bullet \bullet \bullet \quad |\tau\rangle$$

   with equivalent circuit depth as above,
   if the recursion is done via unfolding

*Yoder, Low, and Chuang, PRL **113**, 210501 (2014)*

Examples of quantum recursions

2. Double braket iterations, inspired by Wegner flows

   General technique to iteratively diagonalize an unknown state ρ in the energy eigenbasis, useful in eigenstate preparation tasks

D, ρ_i : unitary channel induced by $e^{s[\hat{D},\rho_i]}$ for some diagonal matrix D, can be done via unfolding

ρ_∞ converges to a diagonal state for small s

Gluza, Quantum ***8****, 1316 (2024)*

Quantum recursions

$$\rho_n = \hat{U}^{(\rho_{n-1})} \rho_{n-1} (\hat{U}^{(\rho_{n-1})})^\dagger$$

Assumptions:

- The n_{th} recursive step $\hat{U}^{(\rho_{n-1})}$ is **unitary** and depends on the **previous result** ρ_{n-1}.
- Initial state ρ_0 and $\hat{U}^{(\rho_0)}$ are given (not necessarily known)
- The exact form of $\hat{U}^{(\rho)}$ as a function of ρ is known

Quantum recursions

$$\rho_n = \hat{U}^{(\rho_{n-1})} \rho_{n-1} (\hat{U}^{(\rho_{n-1})})^\dagger$$

Assumptions:

- The n_{th} recursive step $\hat{U}^{(\rho_{n-1})}$ is **un** **s result** ρ_{n-1}.
- Initial state ρ_0 and $\hat{U}^{(\rho_0)}$ are given (
- The exact form of $\hat{U}^{(\rho)}$ as a functio

What is this unfolding business?

Can we even perform a quantum computation on this naively? We need $\hat{U}^{(\rho_1)}$, $\hat{U}^{(\rho_2)}$, … all the way up to $\hat{U}^{(\rho_{n-1})}$, but we don't really know the intermediate states!

Implementing the next recursion step

This is indeed possible sometimes. Let's look at the particular example when

$$\hat{U}^{(\rho)} = e^{is\rho}$$

Remember: we assume we are given ρ_0 and $\hat{U}^{(\rho_0)}$, our goal is to implement $\hat{U}^{(\rho_1)}$

- Observe that since we do know ρ_0, and furthermore $\rho_1 = \hat{U}^{(\rho_0)}\ \rho_0\ (\hat{U}^{(\rho_0)})^\dagger$, we also have that $e^{is\rho_1} = \hat{U}^{(\rho_0)} e^{is\rho_0} (\hat{U}^{(\rho_0)})^\dagger$, and $\hat{U}(\rho_1)$ is possible to implement without knowing ρ_1 — we just need to call the unitary $\hat{U}^{(\rho_0)}$ and its inverse.
- This example is trivial on its own….
- However, $\hat{U}^{(\rho)} = V_1 \cdot e^{is\rho} \cdot V_2$ is not so trivial, and yet we still have $e^{is\rho_1} = \hat{U}^{(\rho_0)} \cdot e^{is\rho_0} \cdot \hat{U}^{(\rho_0)^\dagger}$.

Implementing the next recursion step

- For a more general unitary $\hat{U}^{(\rho)} = \hat{V}_L e^{is_L\rho}\hat{V}_{L-1}\cdots\hat{V}_1 e^{is_1\rho}\hat{V}_0$, we still can do this: $\hat{U}^{(\rho_1)}$ is possible if we call the unitary $\hat{U}^{(\rho_0)}$ a number of $2L$ times.

How about $\hat{U}^{(\rho_2)}$, $\hat{U}^{(\rho_3)}$, … $\hat{U}^{(\rho_{n-1})}$?

Sticking with the more general example of $\hat{U}^{(\rho)} = \hat{V}_L e^{is_L\rho}\hat{V}_{L-1}\cdots\hat{V}_1 e^{is_1\rho}\hat{V}_0$, to execute $\hat{U}^{(\rho_2)}$,

- we call $\hat{U}^{(\rho_1)}$ a number of $2L$ times,
- each call for $\hat{U}^{(\rho_1)}$ is done by calling $\hat{U}^{(\rho_0)}$ for $2L$ times,
- Hence, the implementation of $\hat{U}^{(\rho_2)}$ requires $4L^2$ calls of $\hat{U}^{(\rho_0)}$. Similarly, $\hat{U}^{(\rho_3)}$ requires $8L^3$ calls of $\hat{U}^{(\rho_0)}$…
  - In other words, $e^{is\rho_n} = \hat{U}^{(\rho_{n-1})}\ldots\hat{U}^{(\rho_0)}e^{is\rho_0}(\hat{U}^{(\rho_0)})^\dagger\ldots(\hat{U}^{(\rho_{n-1})})^\dagger$
- $O((2L)^N)$ steps (i.e. circuit depth) is required for $\hat{U}^{(\rho_{N-1})}$

unfolding, reminiscent of classical naive methods without memory

How about $\hat{U}^{(\rho_2)}$, $\hat{U}^{(\rho_3)}$, … $\hat{U}^{(\rho_{n-1})}$?

Sticking with the more general example of $\hat{U}^{(\rho)} = \hat{V}_L e^{is_L\rho}\hat{V}_{L-1}\cdots\hat{V}_1 e^{is_1\rho}\hat{V}_0$, to execute $\hat{U}^{(\rho_2)}$,

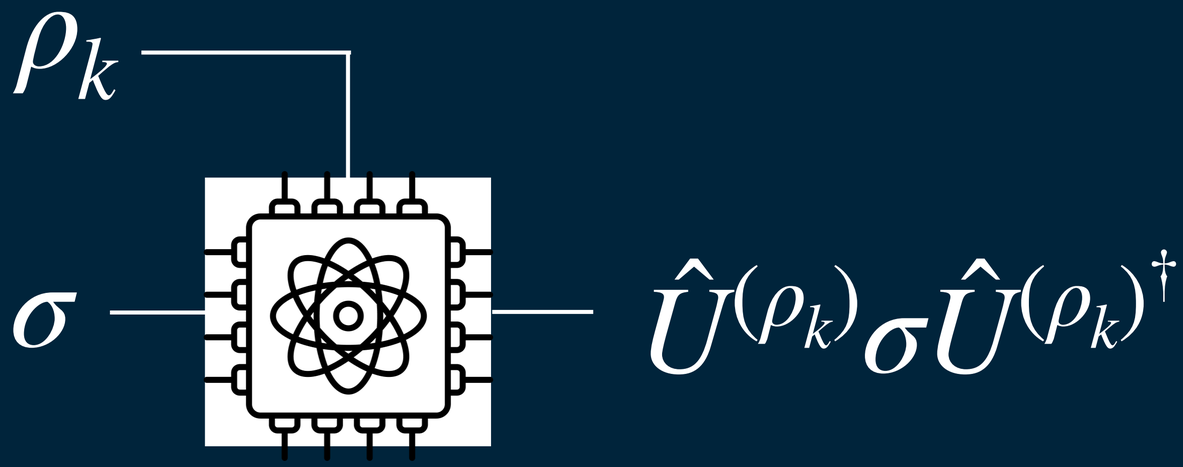
- we call $\hat{U}^{(\rho_1)}$ a number of $2L$ times,
- each call for $\hat{U}^{(\rho_1)}$ is done by calling $\hat{U}^{(\rho_0)}$ for $2L$ times,
- Hence, the implementation of $\hat{U}^{(\rho_2)}$ requires $4L^2$ calls of $\hat{U}^{(\rho_0)}$. Similarly, $\hat{U}^{(\rho_3)}$ requires $8L^3$ calls of $\hat{U}^{(\rho_0)}$…
  - In other words, $e^{is\rho_n} = \hat{U}^{(\rho_{n-1})}\ldots\hat{U}^{(\rho_0)}e^{is\rho_0}(\hat{U}^{(\rho_0)})^\dagger\ldots(\hat{U}^{(\rho_{n-1})})^\dagger$
- $O((2L)^N)$ steps (i.e. circuit depth) is req

unfolding, reminiscent of classical naive methods without memory

Surely it'll be faster to compute this if we have a dynamic version of quantum computation?

Can we program quantum dynamically?

To do so, we need to consider circuits that allow us to take in **instructions encoded in the form of quantum states** ρ_k,



Surely there are better ways to use quantum-mechanically encoded instructions?

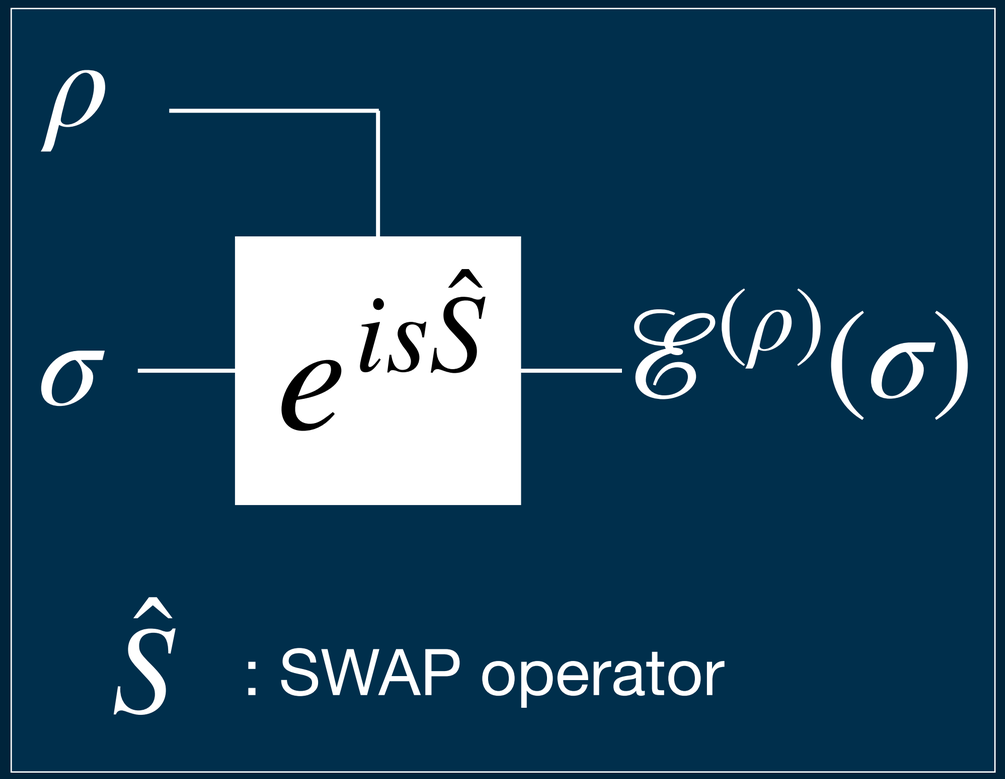
Naive method: learn ρ e.g. tomography, and then compile $\hat{U}^{(\rho)}$ via Solovay-Kitaev.

Surely it'll be faster to compute this if we have a dynamic version of quantum computation?

Yes, DME is a better way!

Suppose we want $\hat{U}^{(\rho)}\sigma\hat{U}^{(\rho)\dagger}$ where $\hat{U}^{(\rho)} = e^{is\rho}$

Density Matrix Exponentiation



Let A $= \cos(s)$ and B $= \cos(s)$. Then

$$\mathscr{E}^{(\rho)}(\sigma) = \mathrm{Tr}_1[e^{-is\hat{S}}(\rho \otimes \sigma)e^{is\hat{S}}]$$
$$= \mathrm{A}^2\sigma\mathrm{Tr}[\rho] - i\mathrm{AB}[\rho,\sigma] + O(s^2)$$

On the other hand, we also know that

$$e^{-is\rho}\sigma e^{is\rho} = \sigma - is[\rho,\sigma] + O(s^2)$$

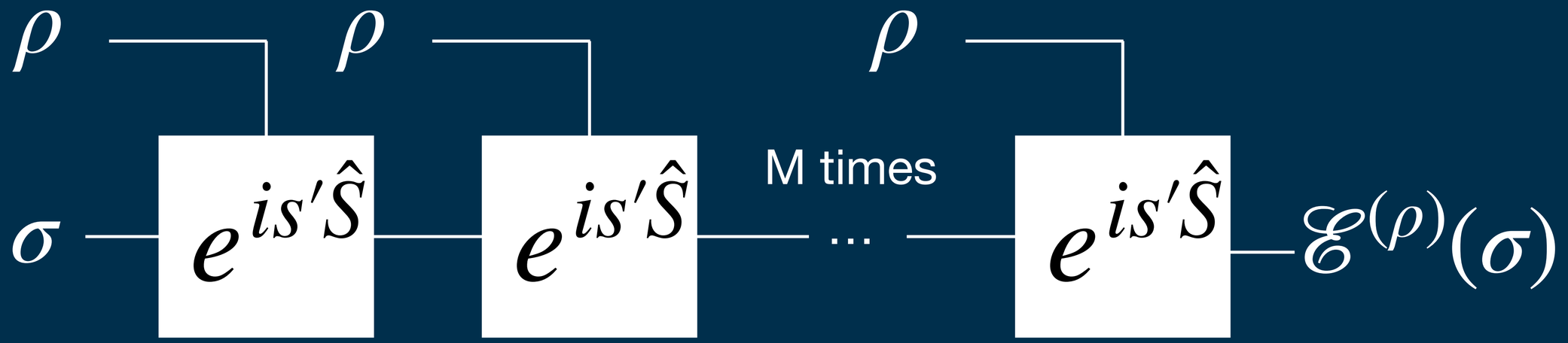
In other words,

$$\| \mathscr{E}_s^{(\rho)}(\bullet) - e^{-is\rho}(\bullet)e^{is\rho} \| = O(s^2)$$

Lloyd, Mohseni, and Rebentrost, Nat. Phys. ***10****, 631 (2014)*

Yes, DME is a better way!

To achieve a higher accuracy, one can use more copies, e.g. M copies of ρ, with smaller values of $s' = s/M$:



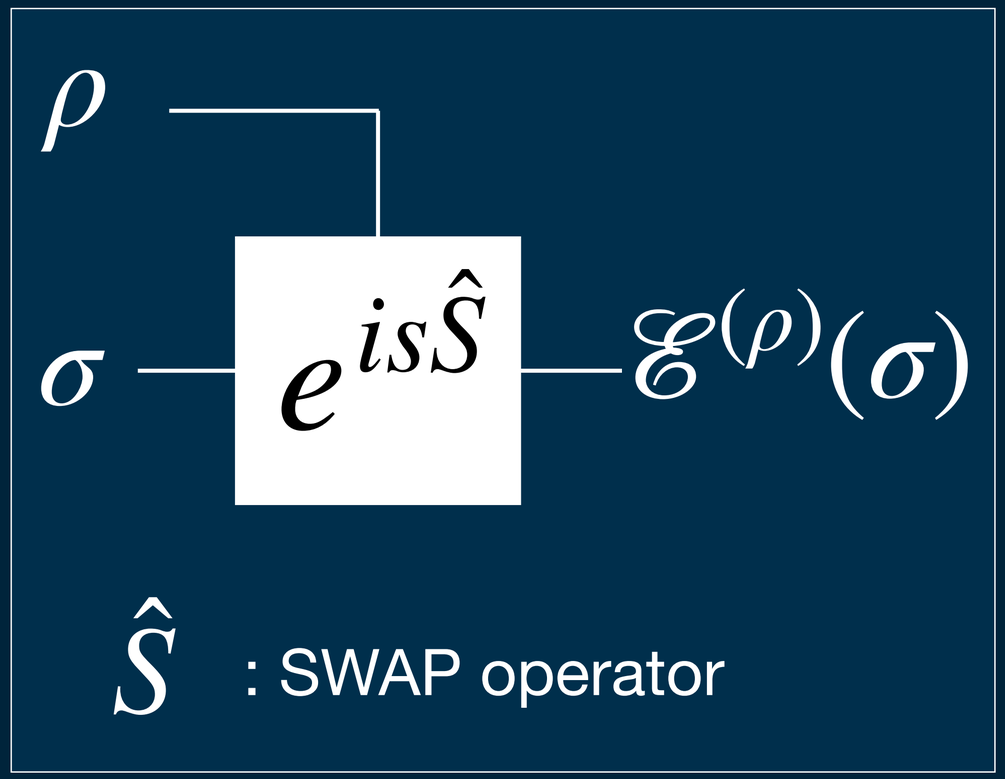
$$\| \mathscr{E}^{(\rho)}_{s/M} \circ \cdots \circ \mathscr{E}^{(\rho)}_{s/M}(\bullet) - e^{-is\rho}(\bullet)e^{is\rho} \| = O(s^2/M) \text{ , or } \quad \epsilon \propto M^{-1}$$

Cost: circuit width (preparation of states is still necessary, although we bypass explicit learning)

Yes, DME is a better way!

Suppose we want $\hat{U}^{(\rho)}\sigma\hat{U}^{(\rho)\dagger}$ where $\hat{U}^{(\rho)} = e^{is\rho}$

Density Matrix Exponentiation



In other words,

$$\| \mathscr{E}_s^{(\rho)}(\bullet) - e^{-is\rho}(\bullet)e^{is\rho} \| = O(s^2)$$

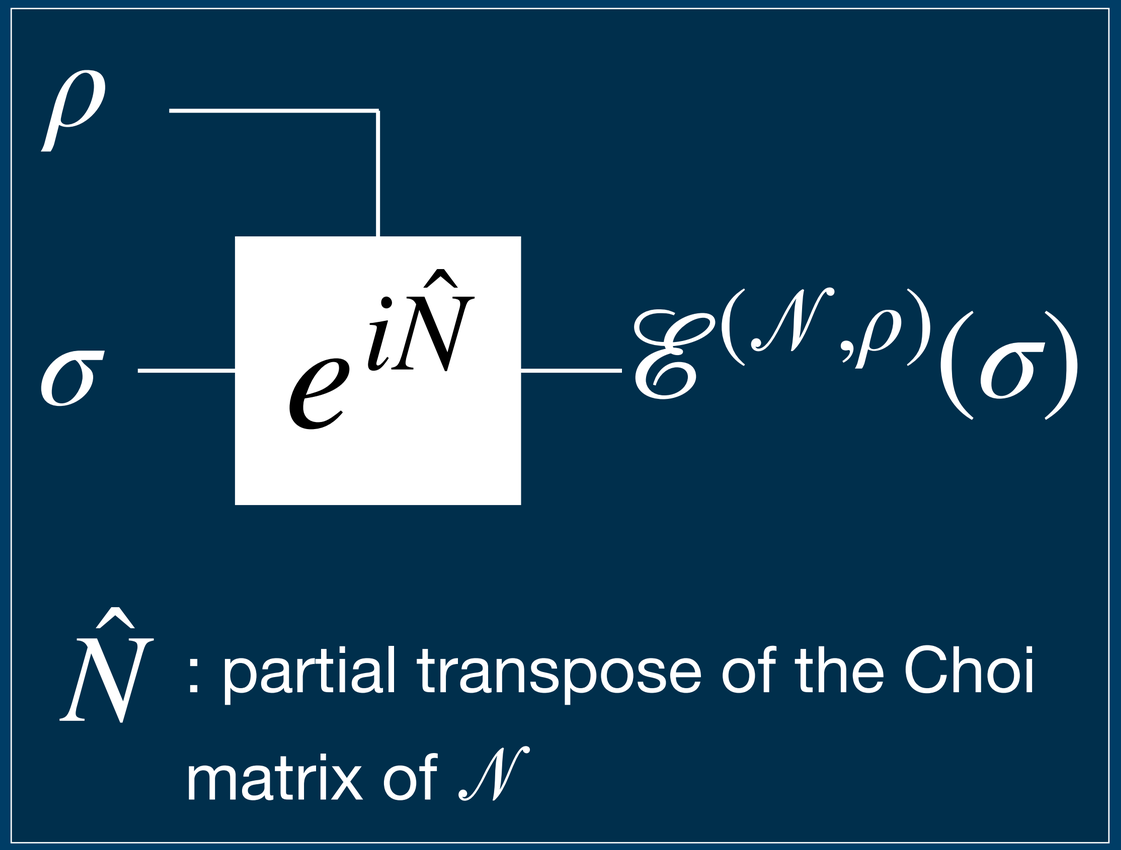
Optimality of DME routine shown for sample complexity in Hamiltonian simulation, in comparison to tomographic methods

Kimmel et al., npj QI ***3****, 13 (2017)*

Lloyd, Mohseni, and Rebentrost, Nat. Phys. ***10****, 631 (2014)*

Generalizations of DME also exist!

Hermitian-preserving Map Exponentiation



Suppose we want to implement the unitary channel induced by $e^{i\mathcal{N}(\rho)}$,

Then $\mathscr{E}^{(\mathcal{N},\rho)}(\sigma)$ is a good approximation.

In other words,

$$\|\mathscr{E}^{(\mathcal{N},\rho)}(\bullet) - e^{-i\mathcal{N}(\rho)}(\bullet)e^{i\mathcal{N}(\rho)}\| = O(\|\hat{N}\|_\infty^2)$$

Supress errors again by using multiple copies

Wei. et al, arXiv:2308.07956 (2023)

Let's recall our recursive problem…

- With unfolding, in order to implement the n-th recursion unitary $\hat{U}^{(\rho_{N-1})}$ alone, $O((2L)^N)$ steps (i.e. exponential circuit depth) is required.
- With quantum instructions, if the recursion unitary has the form of

$$\hat{U}^{(\rho)} = \hat{V}_L e^{-i\mathcal{N}_L(\rho)} \hat{V}_{L-1} \cdots \hat{V}_1 e^{-i\mathcal{N}_1(\rho)} \hat{V}_0$$

where each $\mathcal{N}_i$ are general Hermiticity-preserving maps, then implementing $\hat{U}^{(\rho_N)}$ requires $O(\epsilon^{-1}L)$ copies of ρ_N and circuit of depth $O(\epsilon^{-1}L)$

- preparing multiple copies of ρ_{N-1} still required $O((2L)^{N-1})$ circuit depth, so the replacement of the last step only reduces the overall depth by $1/2L$

Let's recall our recursive problem…

- With unfolding, in order to implement the n-th recursion unitary $\hat{U}^{(\rho_{N-1})}$ alone, $O((2L)^N)$ steps (i.e. exponential circuit depth) is required.
- With quantum instructions, if the recursion unitary has the form of

$$\hat{U}^{(\rho)} = \hat{V}_L e^{-i\mathcal{N}_L(\rho)} \hat{V}_{L-1} \cdots \hat{V}_1 e^{-i\mathcal{N}_1(\rho)} \hat{V}_0$$

where each $\mathcal{N}_i$ are general Hermiticity-preserving maps, then implementing $\hat{U}^{(\rho_N)}$ requires $O(\epsilon^{-1}L)$ copies of ρ_N and

- preparing multiple copies of ρ_{N-1} still re depth, so the replacement of the last step only reduces the overall depth by $1/2L$

But surely we can now apply this to the preparation of ρ_{N-1}!

Now we have a quantum version of dynamic programming, decreasing circuit depth (computational time) exponentially at the cost of width (memory)!

Not so fast!
There are caveats to this…

Errors propagate

- Recall that in order to drastically reduce circuit depth, we implement $\hat{U}^{(\rho_{N-1})}$ ***approximately*** via DME/HME. To do that, we need to prepare multiple copies of ρ_{N-1} efficiently, by implementing $\hat{U}^{(\rho_{N-2})}$ ***approximately***, ……
- In other words, we prepared $\tilde{\rho}_{N-1} \approx \rho_{N-1}$ in every step due to implementation error, which has the potential to accumulate!
  - The same exponential blow-up happens when unfolding unitary operations are not perfect

Oh no! What can we do about this?

Resolution to errors: 1) they don't always blow-up

Theorem (high level description)

- If the recursion unitary $\hat{U}^{(\rho)}$ fulfills a stability criteria,
- Then QDP can implement N recursions with final error ϵ and total circuit of depth $O(N^2\epsilon^{-1})$ — no exponential blowing up of errors
- If your initial state (and therefore target final state) is furthermore pure, then depth $O(N\epsilon^{-1})$ width $e^{O(N)}\epsilon^{-N}$ suffices

- *Stability criteria: given any ρ_0, the sequence of states $\{\rho_i\}_i$ generated by the recursion $\rho_i = U_{i-1}\rho_{i-1}U_{i-1}^\dagger$ has a unique fixed-point τ, such that the distance of ρ_i to τ is contracting at some finite speed*

Resolution to errors: 1) they don't always blow-up

Theorem (high level description)

- If the recursion unitary
- Then QDP can implement
  of depth $O(N^2\epsilon^{-1})$ —
- If your initial state (and
  then depth $O(N\epsilon^{-1})$ wi
- *Stability criteria: given any ρ*
  $\rho_i = U_{i-1}\rho_{i-1}U_{i-1}^\dagger$ *has a un*
  contracting at some finite s

What we need:

For any ρ, s.t. $\mathrm{spec}(\rho) = \mathrm{spec}(\tau)$

$\|\tau - \hat{U}^{(\rho)}(\rho)\| \leq h(\|\tau - \rho\|) < \|\tau - \rho\|$ and

$h(\delta + \epsilon) \leq h(\delta) + r\epsilon$, where $r < 1$ for $\delta < \delta^\star, \epsilon < \epsilon^\star$

Proof idea:

- **unitary errors get suppressed by a factor r after each iteration**
- **non-unitary errors accumulate linearly, use subroutines to supress them**

*Cirac, Ekert, and Macchiavello, PRL **82**, 4344 (1999)*

Resolution to errors: 2) QDP offers a hybrid approach

Suppose a quantum processor with specifications of circuit depth and volume.

- Using unfolding only, there is a maximum achievable recursion ρ_{N_1} due to depth limitations. Using QDP only, there is a maximum achievable recursion ρ_{N_2} due to width limitations
- A hybrid approach allows for:
  - Implementing N_1 iterations with unfolding — $e^{O(N_2)}$ such circuits are run in parallel; depth scales as $e^{O(N_1)}$
  - N_2 iterations are subsequently implemented with QDP
  - Obtains ρ_N, with $N \approx N_1 + N_2$ recursive steps with total circuit depth: $\mathrm{poly}(N_2)e^{O(N_1)}$

Take-home message

- Quantum recursions are expensive…… you pay either with circuit depth (unfolding), or width.
- QDP gives us an additional tool to make full use of a quantum processor, trading depth at the cost of width.
- Will this really be useful? I don’t know, but we have plans to find out…
  - Marek is outlining a roadmap on systematic usage of double-brackets
    - Integrate DB & QDP onto QIBO (open source middleware for quantum computing)
    - Plans for whitepaper on DB
- Singapore has long-term ambitious plans in building quantum computers