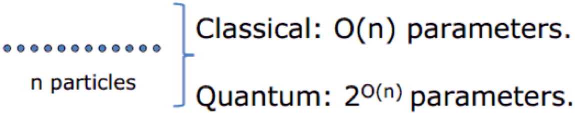
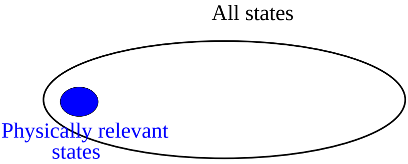
Local Algorithms for 1D Quantum Systems

I. Arad, Z. Landau, U. Vazirani, T. Vidick

Many-body physics



Are physical systems special?

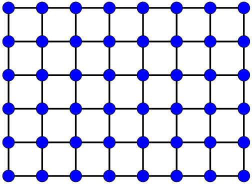


Ground States of Local Hamiltonians

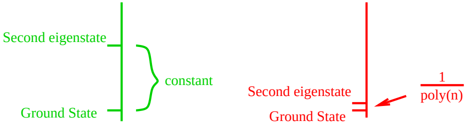
- Hilbert space of particles $\mathcal{H} = (\mathbb{C}^d)^{\otimes n}$,
- Local terms $0 \leq H_i \leq 1$,
- Interested in lowest eignevectors of $H = \sum_i H_i$.

Ground States of Local Hamiltonians

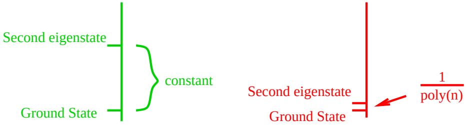
- Hilbert space of particles $\mathcal{H} = (\mathbb{C}^d)^{\otimes n}$,
- Local terms $0 \leq H_i \leq 1$,
- Interested in lowest eignevectors of $H = \sum_i H_i$.



1D Hamiltonians

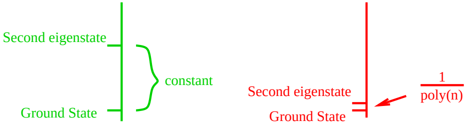


1D Hamiltonians



DMRG does well in practice ['92].

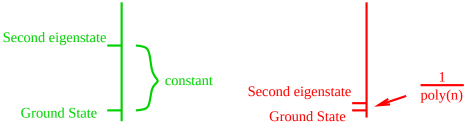
1D Hamiltonians



DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

1D Hamiltonians

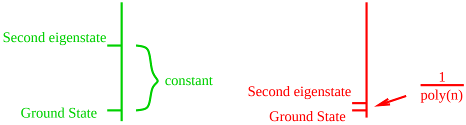


DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

Area law for 1D with unique ground state and constant gap['07].

1D Hamiltonians

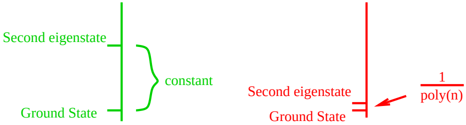


DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

1D Hamiltonians



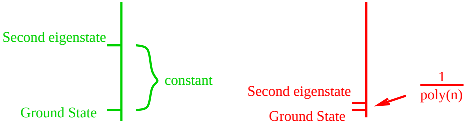
DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

What happens for critical systems (gap $\rightarrow 0$)?

1D Hamiltonians



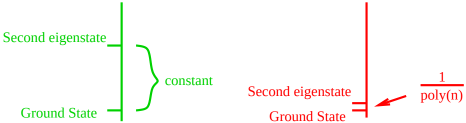
DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

What happens for critical systems (gap $\rightarrow 0$)?
Phase transition: believed "Log correction to area law"

1D Hamiltonians



DMRG does well in practice ['92].

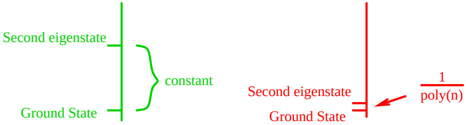
QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

What happens for critical systems (gap $\rightarrow 0$)?
Phase transition: believed "Log correction to area law"

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)

1D Hamiltonians



DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

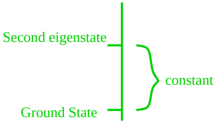
Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

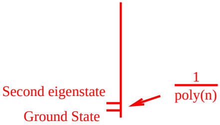
What happens for critical systems (gap $\rightarrow 0$)?
Phase transition: believed "Log correction to area law"

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)

Density of states assumptions $\rightarrow$ area laws

1D Hamiltonians





DMRG does well in practice ['92].

QMA-complete for 1D (gap $\frac{1}{poly(n)}$)['07]

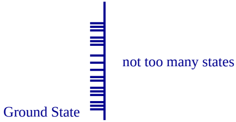
Area law for 1D with unique ground state and constant gap['07].
Polynomial time algorithm for finding unique ground state with constant gap['13].

What happens for critical systems (gap $\rightarrow 0$)?

Phase transition: believed "Log correction to area law"

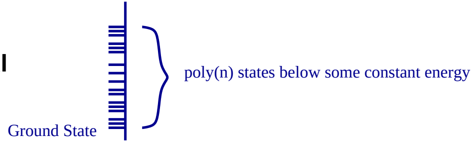
QMA-complete for 1D (gap $\frac{1}{poly(n)}$)

Density of states assumptions $\rightarrow$ area laws



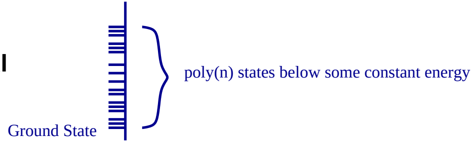
Result: Algorithm for critical systems

poly(n) eigenvalues in constant interval



Result: Algorithm for critical systems

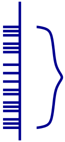
poly(n) eigenvalues in constant interval



- $2^{poly(\log n)}$ time algorithm for finding low energy states,
- MPS description with bond dimension $2^{poly(\log n)}$

Result: Algorithm for critical systems

poly(n) eigenvalues in constant interval



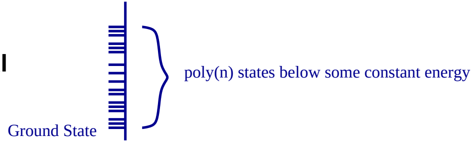
- $2^{poly(\log n)}$ time algorithm for finding low energy states,
- MPS description with bond dimension $2^{poly(\log n)}$

Notable features:

- More physically plausible: No convex optimization.

Result: Algorithm for critical systems

poly(n) eigenvalues in constant interval



- $2^{poly(\log n)}$ time algorithm for finding low energy states,
- MPS description with bond dimension $2^{poly(\log n)}$

Notable features:

- More physically plausible: No convex optimization.
- Hierarchical structure reminiscent of Renormalization Group.

Result: Algorithm for degenerate ground space

Degenerate ground space with gap:

Result: Algorithm for degenerate ground space

Degenerate ground space with gap:

Generalization of unique ground state case?

Result: Algorithm for degenerate ground space

Degenerate ground space with gap:

Generalization of unique ground state case?

Small perturbation would have $1/poly(n)$ gap

Result: Algorithm for degenerate ground space

Degenerate ground space with gap:

Generalization of unique ground state case?

Small perturbation would have $1/poly(n)$ gap

Area law and algorithm for constant degeneracy [Huang; Chubb, Flammia] .

Result: Algorithm for degenerate ground space

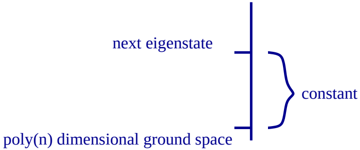
Degenerate ground space with gap:

Generalization of unique ground state case?

Small perturbation would have $1/poly(n)$ gap

Area law and algorithm for constant degeneracy [Huang; Chubb, Flammia] .

poly(n) degenerate ground space



Result: Algorithm for degenerate ground space

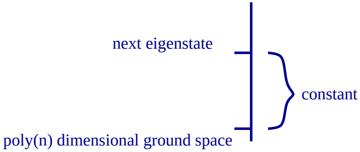
Degenerate ground space with gap:

Generalization of unique ground state case?

Small perturbation would have $1/poly(n)$ gap

Area law and algorithm for constant degeneracy [Huang; Chubb, Flammia] .

poly(n) degenerate ground space



- Polynomial time algorithm for finding the ground space.

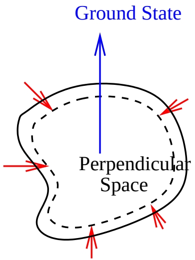
Key tool: Approximate Ground State Projection (AGSP)

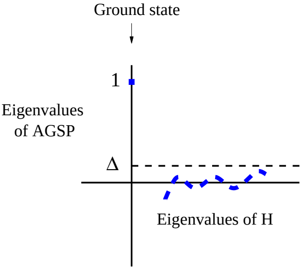
Key tool: Approximate Ground State Projection (AGSP)

- It approximately projects onto the ground state:

Key tool: Approximate Ground State Projection (AGSP)

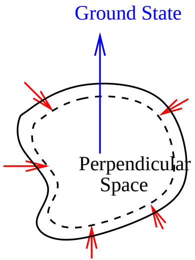
- It approximately projects onto the ground state:

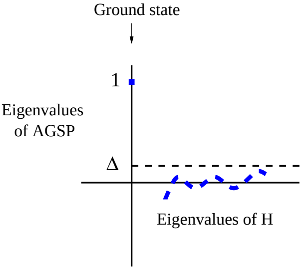




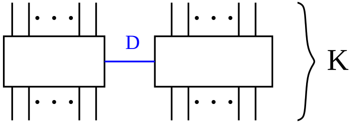
Key tool: Approximate Ground State Projection (AGSP)

- It approximately projects onto the ground state:



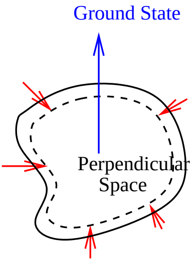


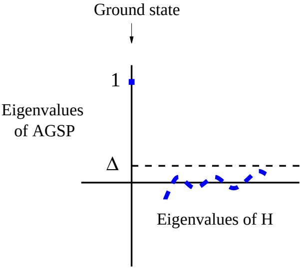
- It has small entanglement rank:



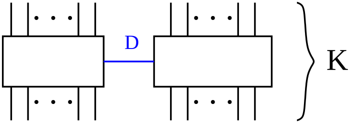
Key tool: Approximate Ground State Projection (AGSP)

- It approximately projects onto the ground state:





- It has small entanglement rank:

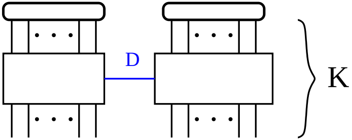


Critical threshold $D\Delta < 1$.

Steps towards an algorithm

A first pass.

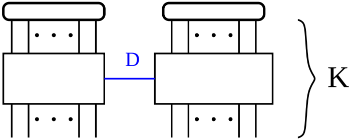
1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.

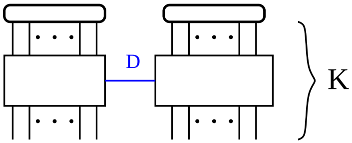


Each iteration moves closer $\rightarrow$ Area Law.

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



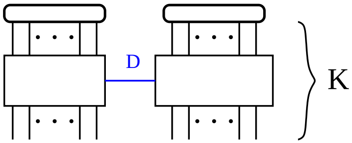
Each iteration moves closer $\rightarrow$ Area Law.

Algorithm?

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Each iteration moves closer $\rightarrow$ Area Law.

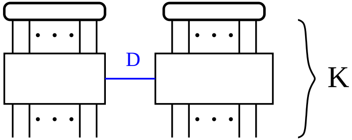
Algorithm?

- Will take $O(n)$ iterations to get close.

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Each iteration moves closer $\rightarrow$ Area Law.

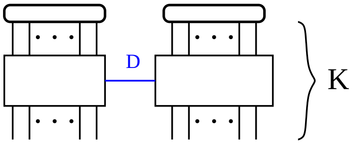
Algorithm?

- Will take $O(n)$ iterations to get close.
- Complexity OK at the cut but too complex on sides.

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Each iteration moves closer $\rightarrow$ Area Law.

Algorithm?

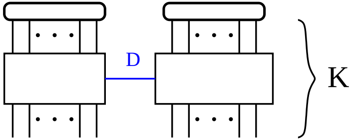
- Will take $O(n)$ iterations to get close.
- Complexity OK at the cut but too complex on sides.

Two directions:

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Each iteration moves closer → Area Law.

Algorithm?

- Will take $O(n)$ iterations to get close.
- Complexity OK at the cut but too complex on sides.

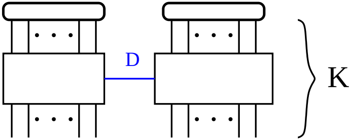
Two directions:

1. Reduce complexity → original algorithm.

Steps towards an algorithm

A first pass.

1. Start with unentangled state.
2. Apply AGSP. Randomly pick one Schmidt vector. Repeat.



Each iteration moves closer $\rightarrow$ Area Law.

Algorithm?

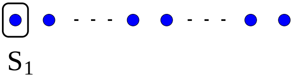
- Will take $O(n)$ iterations to get close.
- Complexity OK at the cut but too complex on sides.

Two directions:

1. Reduce complexity $\rightarrow$ original algorithm.
2. Reduce iterations $\rightarrow$ new results.

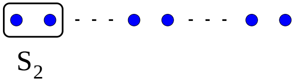
Original algorithm: bird's eye view

Original algorithm: bird's eye view



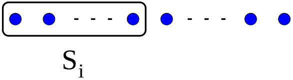
Viable set: subspace containing left Schmidt vectors of a good approximation to ground space

Original algorithm: bird's eye view



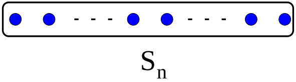
Viable set: subspace containing left Schmidt vectors of a good approximation to ground space

Original algorithm: bird's eye view



Viable set: subspace containing left Schmidt vectors of a good approximation to ground space

Original algorithm: bird's eye view



Viable set: subspace containing left Schmidt vectors of a good approximation to ground space

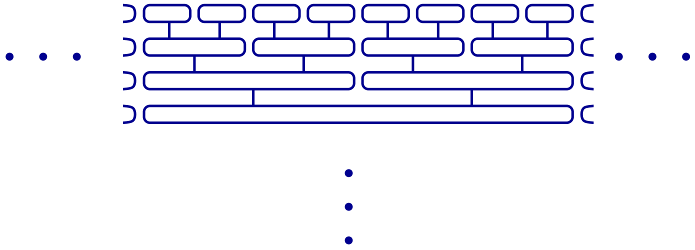
Reduce iterations

Move more quickly via simultaneous local movement?

Reduce iterations

Move more quickly via simultaneous local movement?

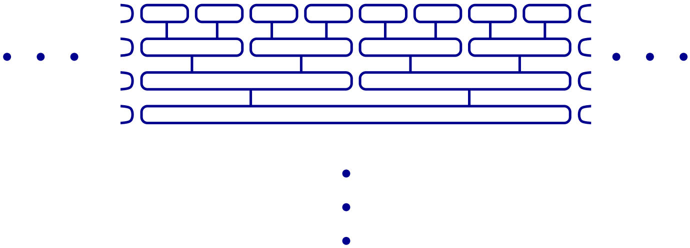
Tree Structure:



Reduce iterations

Move more quickly via simultaneous local movement?

Tree Structure:



Need: a way to merge two viable sets.

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

Merge Process

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

Merge Process

1. Merge: Tensor two neighboring viable sets $V = V_1 \otimes V_2$,

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

Merge Process

1. Merge: Tensor two neighboring viable sets $V = V_1 \otimes V_2$,
2. Size Reduction:

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

Merge Process

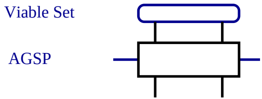
1. Merge: Tensor two neighboring viable sets $V = V_1 \otimes V_2$,
2. Size Reduction:
    1. Choose a small random subspace of $V' \subset V$,

New Algorithm Ingredients

- *Merge process*
- Modified AGSP
- Viable set for subspaces

Merge Process

1. Merge: Tensor two neighboring viable sets $V = V_1 \otimes V_2$,
2. Size Reduction:
   1. Choose a small random subspace of $V' \subset V$,
   2. "Apply AGSP"
      write $K = \sum_{i,j=1}^{D} A_i \otimes B_{i,j} \otimes C_j$, and choose $V'' = span\{B_{i,j}V'\}$.



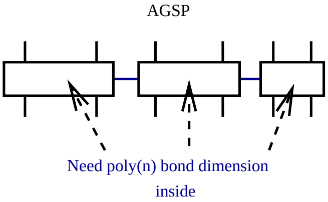
New Algorithm Ingredients

- Merge process
- *Modified AGSP*
- Viable set for subspaces

New Algorithm Ingredients

- Merge process
- *Modified AGSP*
- Viable set for subspaces

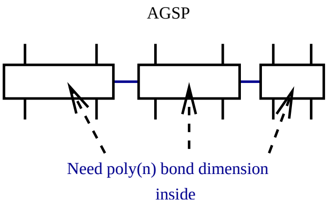
Modified AGSP



New Algorithm Ingredients

- Merge process
- *Modified AGSP*
- Viable set for subspaces

Modified AGSP



Requires soft truncation

- Uses [Molnar, Schuch, Verstraete, Cirac]: $e^{-\beta H}$ has small bond dimension MPO approximation.

New Algorithm Ingredients

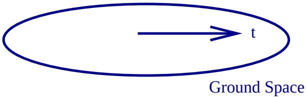
- Merge process
- Modified AGSP
- *Viable set for subspaces*

New Algorithm Ingredients

- Merge process
- Modified AGSP
- *Viable set for subspaces*

V **a viable set for subspace** T

For every element $t \in T$: a vector with left-Schmidt vectors in V that is near t **and projects** exactly in the direction t.

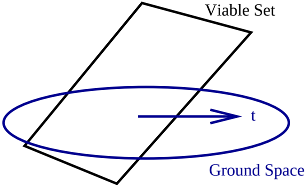


New Algorithm Ingredients

- Merge process
- Modified AGSP
- *Viable set for subspaces*

V **a viable set for subspace** T

For every element $t \in T$: a vector with left-Schmidt vectors in V that is near t **and projects** exactly in the direction t.

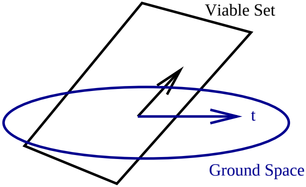


New Algorithm Ingredients

- Merge process
- Modified AGSP
- *Viable set for subspaces*

V a viable set for subspace T

For every element $t \in T$: a vector with left-Schmidt vectors in V that is near t **and projects** exactly in the direction t.

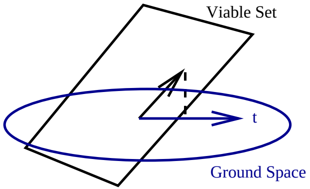


New Algorithm Ingredients

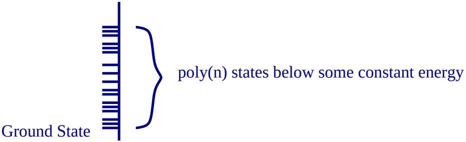
- Merge process
- Modified AGSP
- *Viable set for subspaces*

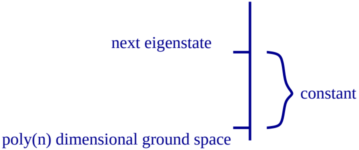
V **a viable set for subspace** T

For every element $t \in T$: a vector with left-Schmidt vectors in V that is near t **and projects** exactly in the direction t.



Discussion





- Physics viewpoint: when do density of states conditions occur? Examples when they don't? Proofs?
- Could this be a skeleton for a practical algorithm? Are there interesting questions in this regime to answer? What kinds of challenges exist?
- What kinds of questions/connections does the tree-like structure of the algorithm suggest?