Optimize Symmetric Infinite Projected Entangle-Pair State with Automatic Differentiation

Xingyu Zhang

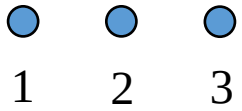
Ghent university

https://github.com/XingyuZhang2018/2025-Lumi-Hackathon

Background

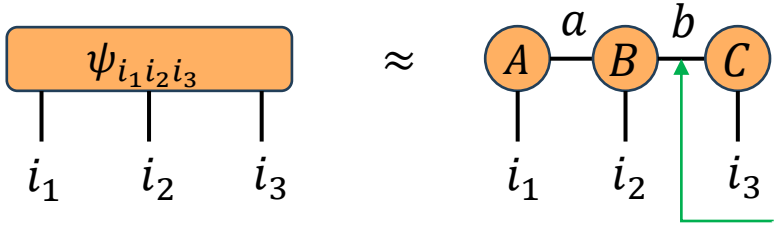
Tensor network ansatz for wave function

- lattice



$\{|\uparrow\rangle, |\downarrow\rangle\}$ for each site

$$|\psi\rangle = \sum_{i_1 i_2 i_3} \psi_{i_1 i_2 i_3} |i_1 \otimes i_2 \otimes i_3\rangle$$



Matrix product state (MPS)

$$\psi_{i_1 i_2 i_3} \approx \sum_{ab} A^a_{i_1} B^{ab}_{i_2} C^b_{i_3}$$

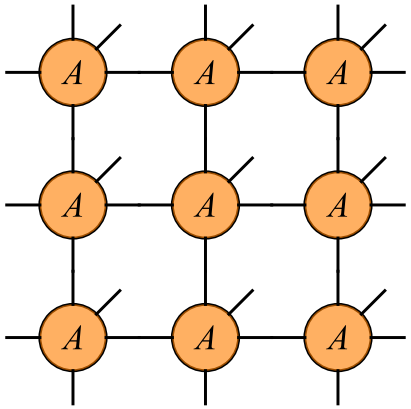
Coefficients: 2^3 vs $2D + 2D^2 + 2D$

d^N vs $\sim NdD^2$

Polynomial representation for exponential space

S.R. White, PRL 69, 2863(1992)

2D ansatz: IPEPS

$|\Psi\rangle =$ 

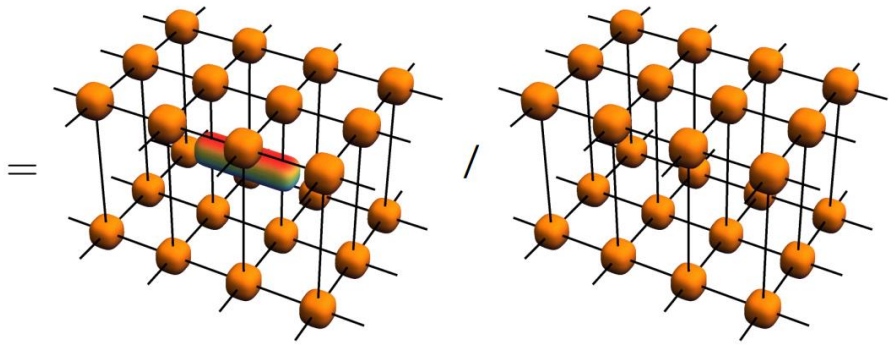
$$E = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle}$$

How to optimize and contract?

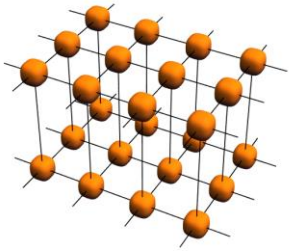
AD-VUMPS

- Optimization Energy

$$E(A) = \langle \Psi(A) | H | \Psi(\bar{A}) \rangle / \langle \Psi(A) \mid \Psi(\bar{A}) \rangle$$

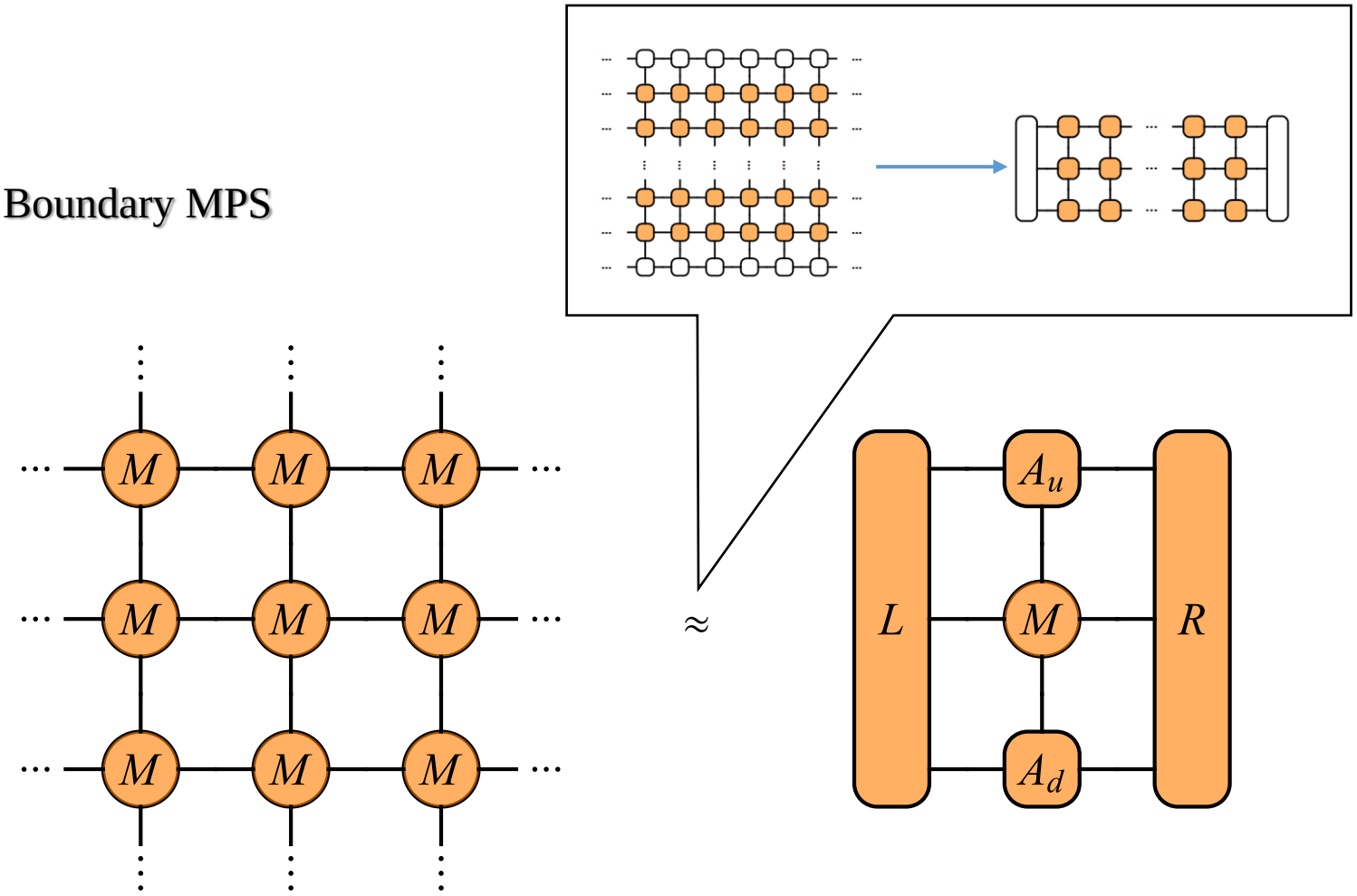


- VUMPS for tensor contraction



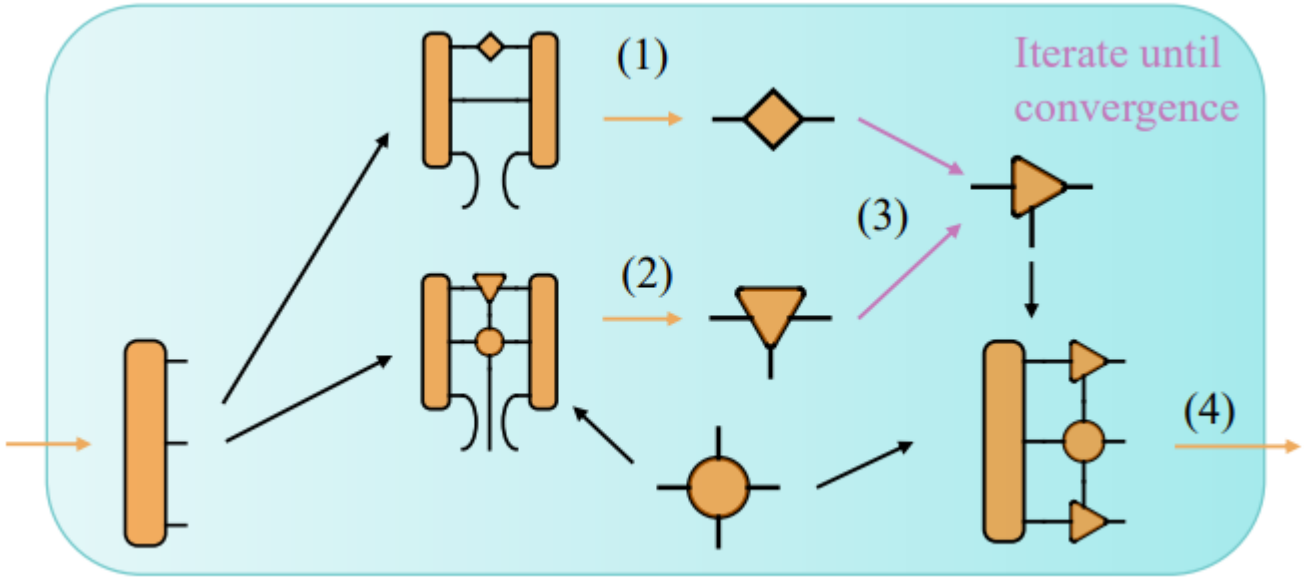
Contraction: VUMPS

Boundary MPS



Computation Graphs

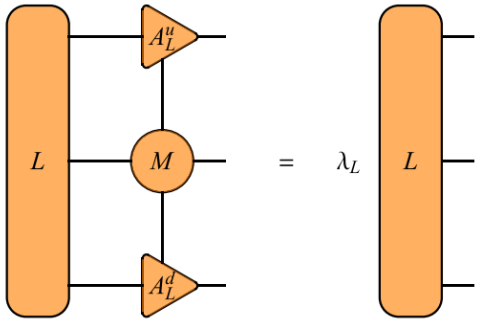
AD for gradient



Project

Project 1: symmetry tensor multiplication

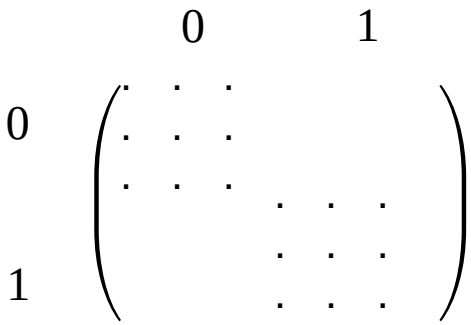
- Largest computation cost
  - dominant eigsolve
  - The largest matrix multiplication size is

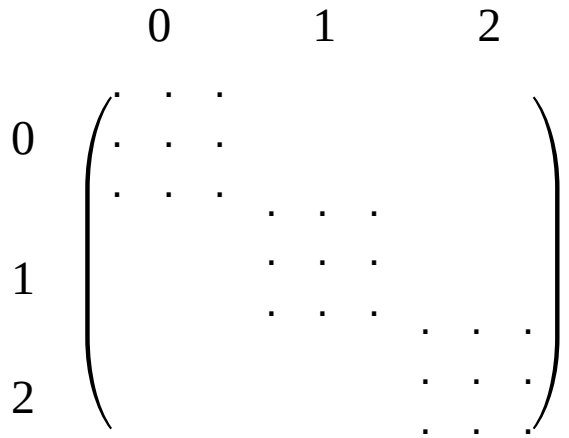


Order 1: $\chi^2 D^2 \times D^2 \times D^2 d \to O(\chi^2 D^6 d) \to O(D^{10})$

Order 2: $\chi^2 \times D^4 \times D^4 \to O(\chi^2 D^8) \to O(D^{12})$

- $Z2 \to U1$





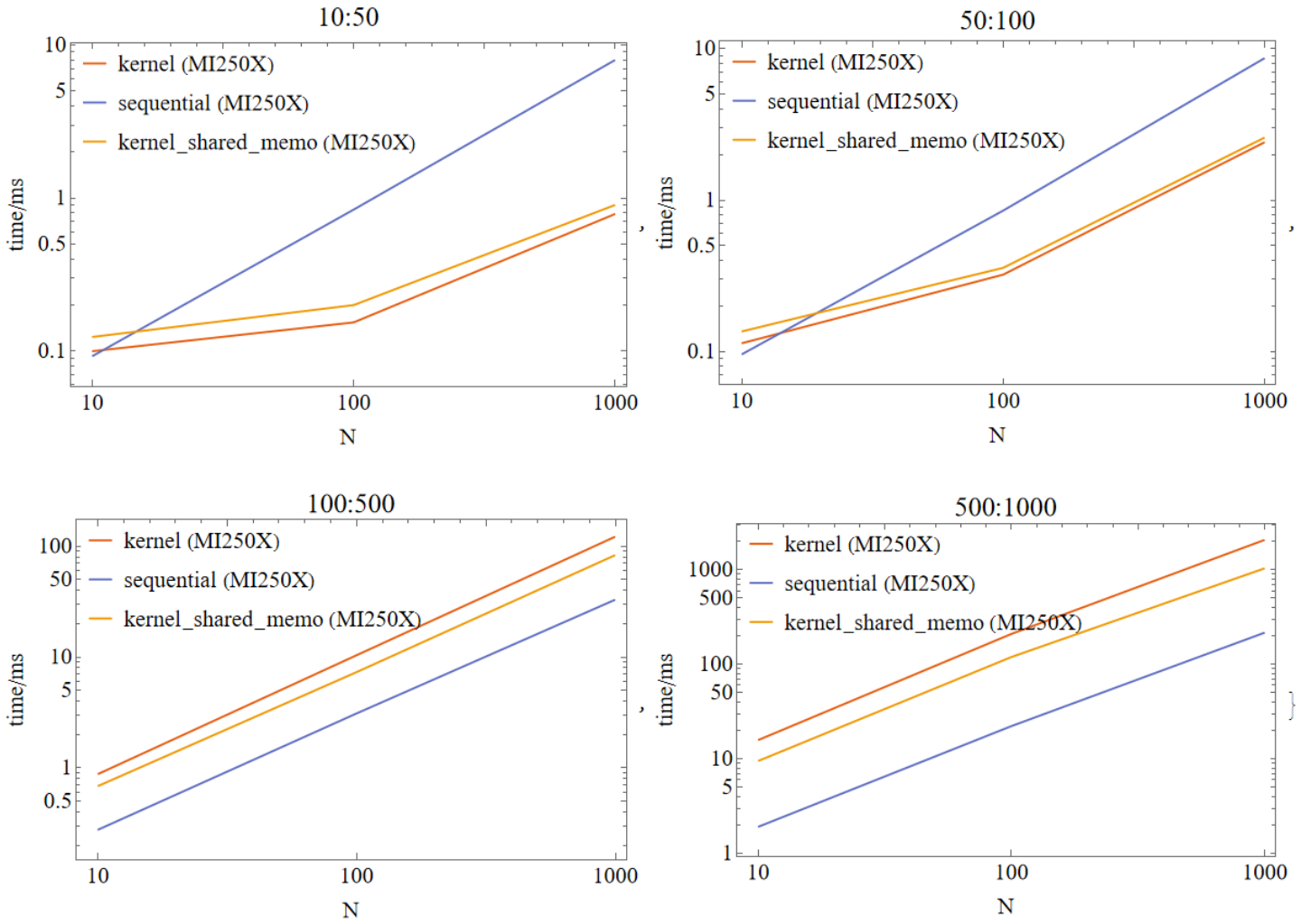
- Tensor contraction → multiple small matrix multiplications

Setup

- $C_{ij}^{n} = \sum_k A_{ik}^{n} B_{kj}^{n} \qquad 1 \le n \le N$
  - size $1 \le i \le M_n^i$
- A, B, C are stored as large vectors

Benchmark: Kernel vs Sequential

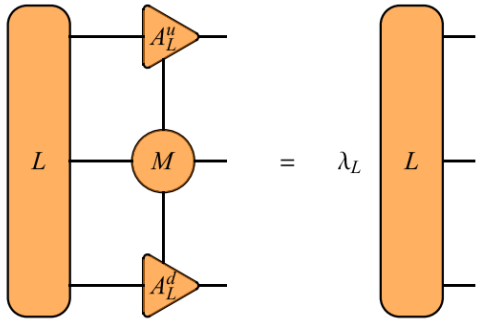
Kernel programming: AMDGPU.jl and KernelAbstractions.jl



Practical Lmap

- $\chi = 160, D = 9, d = 4$

```
χ = Rep[U₁](0=>40, 1=>30, -1=>30, 2=>20, -2=>20, 3=>10, -3=>10)
D = Rep[U₁](0=>3, 1=>2, -1=>2, 2=>1, -2=>1)
d = Rep[U₁](0=>2, 1=>1, -1=>1)
```



- size pairs

Order 1: ((2200, 40, 2200),(2020, 30, 2020),(2020, 30, 2020),(1560, 20, 1560),(1560, 20, 1560),(1000, 10, 1000),(1000, 10, 1000),(70, 19, 291600),(61, 16, 274800),(61, 16, 274800),(40, 10, 229700),(40, 10, 229700),(19, 4, 169600),(19, 4, 169600),(6, 1, 109800),(6, 1, 109800),(19, 70, 291600),(16, 61, 274800),(16, 61, 274800),(10, 40, 229700),(10, 40, 229700),(4, 19, 169600),(4, 19, 169600),(1, 6, 109800),(1, 6, 109800),(40, 2200, 2200),(30, 2020, 2020),(30, 2020, 2020),(20, 1560, 1560),(20, 1560, 1560),(10, 1000, 1000),(10, 1000, 1000))
Order 2: ((2200, 40, 2200),(2020, 30, 2020),(2020, 30, 2020),(1560, 20, 1560),(1560, 20, 1560),(1000, 10, 1000),(1000, 10, 1000),(1107, 1107, 4400),(1016, 1016, 4000),(1016, 1016, 4000),(784, 784, 3100),(784, 784, 3100),(504, 504, 2000),(504, 504, 2000),(266, 266, 1000),(266, 266, 1000),(112, 112, 400),(112, 112, 400),(36, 36, 100),(36, 36, 100),(40, 2200, 2200),(30, 2020, 2020),(30, 2020, 2020),(20, 1560, 1560),(20, 1560, 1560),(10, 1000, 1000),(10, 1000, 1000))

Time/ms	Sequential	Kernel	Shared memo
Order 1	8.873	16.419	20.643
Order 2	10.208	105.641	55.902

Project 2: Matrix multiplication with multiple GPUs

- The memory is limited for one GPU card
- Parallel strategy
  - High level: limited by several cards
    - Up and down boundary MPS
    - Large unit cell
  - Low level: data transfer between different cards
    - Large matrix multiplication

Project 3: Krylov space method in GPU

- KrylovKit.jl: Slow for GPU
  - Array of Array Inner is slow
  - Increase and decrease krylov space

```
@testset "eigsolve $atype" for atype in [Array]
    Random.seed!(100)
    N = 10^3
    A = atype(rand(ComplexF64, N, N))
    v0 = atype(rand(ComplexF64, N))
    linearmap(v) = A * v
    @btime dot($v0, $v0)
    @btime $linearmap($v0)
    @btime λs, vs = eigsolve(v -> $linearmap(v), $v0, 1, :LM)
end | 1-element Vector{Any}:
```

```
  98.097 ns (0 allocations: 0 bytes)
  51.900 μs (3 allocations: 15.69 KiB)
  2.491 ms (350 allocations: 1.06 MiB)
Test Summary:   | Total   Time
eigsolve Array |     0  19.0s
```

```
  32.152 μs (6 allocations: 112 bytes)
  51.370 μs (36 allocations: 768 bytes)
  39.419 ms (8585 allocations: 357.25 KiB)
Test Summary:     | Total   Time
eigsolve ROCArray |     0  26.8s
```

Thank you for Listening!

Q&A