Community Detection Meets Parallelism

A Scalable Approach to Quantum Circuit Simulation

Presented By:

Muhammad Usman
Abdul Wahab
Ahmed Ali Zaidi

WHY SIMULATE QUANTUM CIRCUITS CLASSICALLY ?

- In the current era of quantum computing (NISQ), devices are noisy and have limited qubit counts.
- Classical simulators validate quantum algorithms and testing circuit behavior.
- However, simulation becomes exponentially more difficult as the number of qubits increases.
- Quantum hardware is not yet scalable, we rely on simulations to explore quantum solutions. (e.g., 50 qubits ≈ 9 PB)

TENSOR NETWORKS OVERVIEW

Tensor networks efficiently represent quantum circuits where tensors are nodes and qubit interactions are edges.

Nodes = quantum gates (tensors)

Edges = qubits (indices)

Simulation involves contracting these tensors, similar to multiplying matrices

Merge tensors pairwise to compute amplitudes

Choosing the right contraction order can drastically reduce cost but finding it is NP-hard.

PROPOSED SOLUTION

The authors proposed to accelerate quantum circuit simulation using tensor networks. Each of these techniques tackles a different performance bottleneck —either in computation speed, memory usage, or both.

1. ComPar
2. ComPar_gpu
3. ParSli
4. GN_gpu

COMPAR
COMMUNITY-BASED PARALLEL ALGORITHM

Divides tensor network into communities using Girvan–Newman algorithm.

Three stages:

1. Community Detection
2. Parallel contraction of communities (CPU)
3. Final contraction (CPU/GPU)

Low interconnection = lower memory + faster simulation
Suitable for high-entanglement circuits (e.g., QFT, RQC)

COMPAR_GPU
HYBRID PARALLEL ALGORITHM

Same as ComPar but stage 3 uses GPU

Dual parallelism:

1. Stage 2: CPU contracts communities in parallel
2. Stage 3: GPU accelerates final tensor contraction

Achieved 44× speedup on large RQC circuits

PARALLEL SLICING ALGORITHM

- Slicing technique: splits large tensor networks by cutting indices
- Each contracted in parallel using MPI processes
- Best for memory-intensive or distributed setups

GN_GPU

- Applies Girvan–Newman to get contraction order
- Simple and fast for medium-sized circuits
- It’s simple to implement.

COMPARISON OF ALGORITHMS

	QFT30		RQC-6-6-32	
	Contraction	Total	Contraction	Total
GN_gpu	**5.33**	1.1	1.1	1.13
ComPar_cpu	3.17	1.36	3.66	2.83
ComPar_gpu	2.45	**1.94**	**44.73**	**12.68**
ParSli	1.97	1.01	5.93	1.06

The algorithm achieving the highest speedup in each column are shown in bold

LIMITATIONS

High Spatial Cost

- Memory consumption bottleneck(e.g., QFT, RQC).
- Large intermediate tensors.
- This limits the maximum circuit size, with practical caps around 41 qubits

Limited Parallelism in Final Stage

- The final contraction stage involves a small number of tensors, reducing opportunities for parallel execution.
- dominant performance bottleneck in both time and memory.
- Limits overall scalability and simulation speed.

OVERHEAD OF GIRVAN-NEWMAN

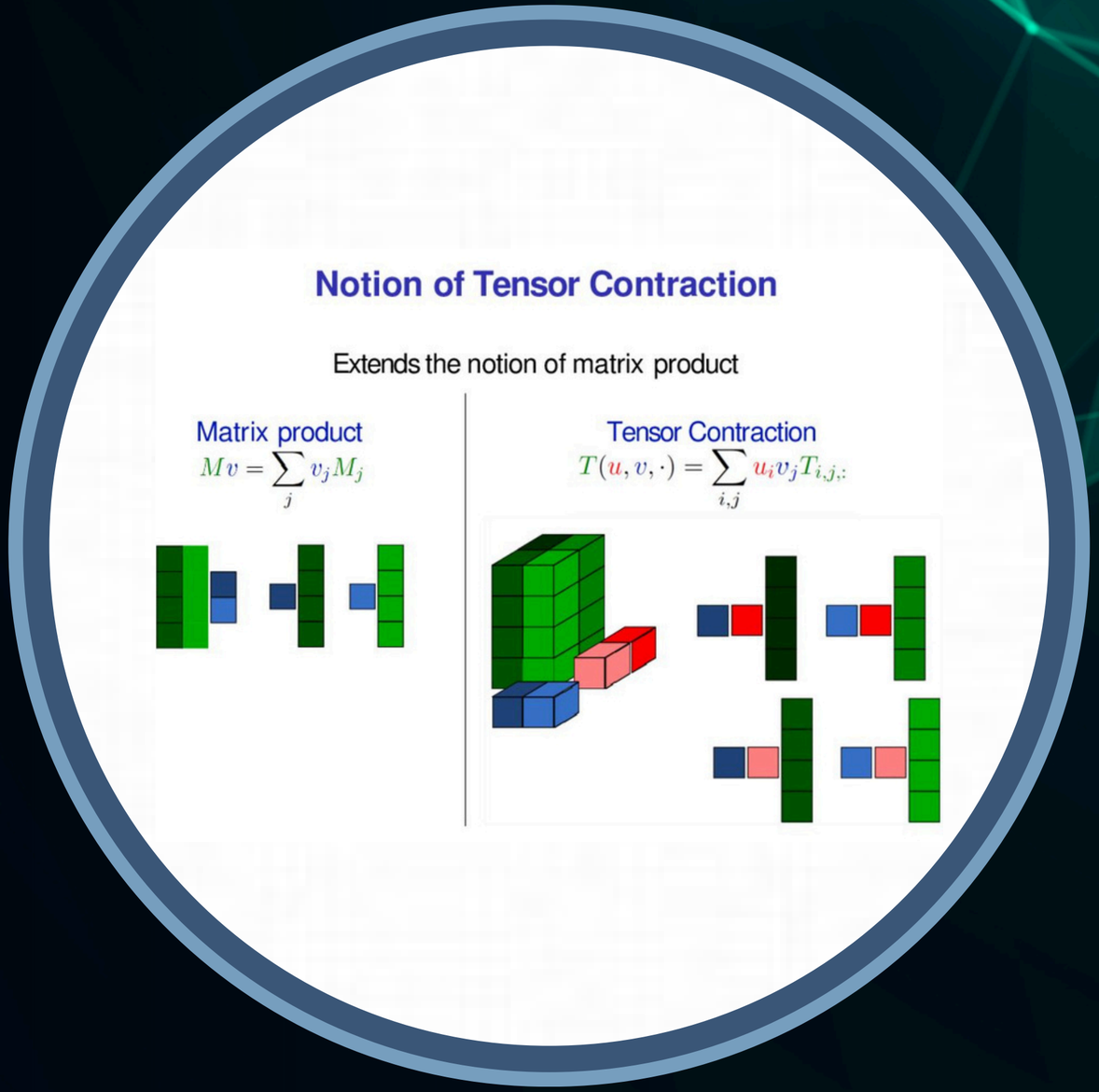
- Recomputes edge betweenness after each removal
- Not scalable for large tensor networks

EXCESS SLICING

- Tensor slicing is used to reduce memory footprint, but introduces computational overhead and management complexity.
- Can result in a large number of sub-networks, reducing overall performance.

TENSOR CONTRACTION

- Combines two tensors by summing over shared indices (like matrix multiplication).
- Core operation in tensor networks for simulating quantum circuits.
- Contraction order heavily impacts time and memory cost.



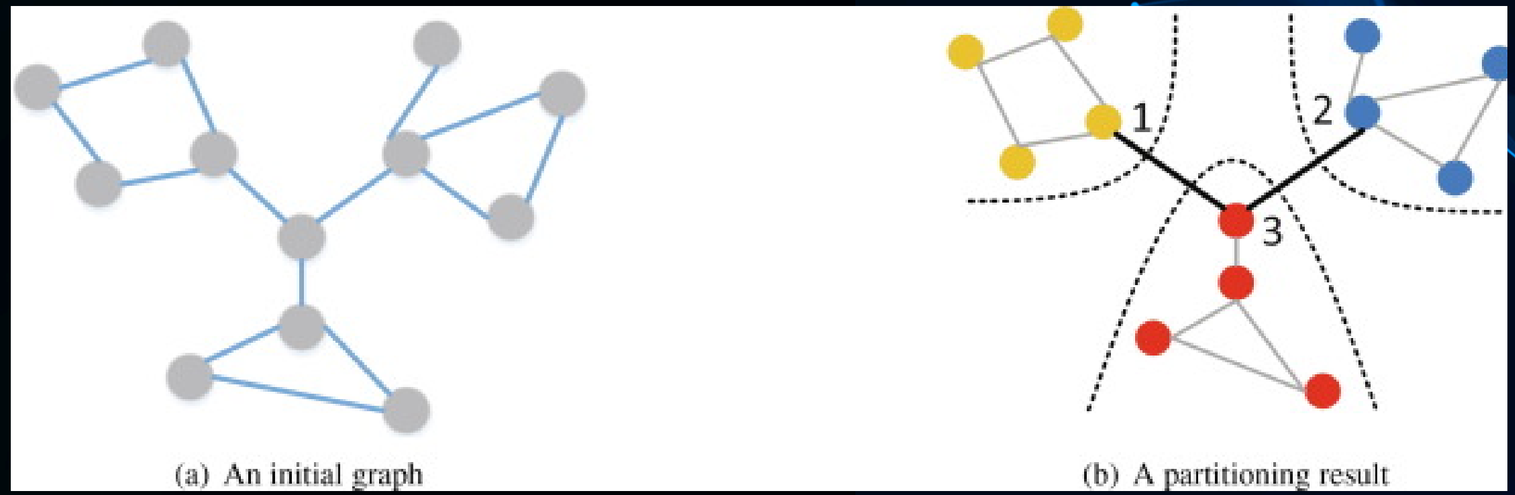
Why METIS ?

Advantages Over Girvan–Newman

01 **Faster for Larger Graphs.**

02 **Scalable for Distributed Systems.**

03 **Minimizes Edge Cuts**



(a) An initial graph

(b) A partitioning result

Challenges & Solutions

Challenges:

- Load imbalance
- MPI communication overhead
- GPU memory limits

Solutions:

- METIS balancing constraints
- Non-blocking MPI
- Optimized OpenCL kernels

CONCLUSION

Graph partitioning → splits tensor network into communities

- **MPI** → across node
- **OpenMP / OpenCL** → within nodes

✅Faster execution

✅Lower memory usage

✅Less communication overhead

ANY QUESTIONS

THANK YOU!

FOR YOUR ATTENTION