Statement of Research

JinGuo Liu
Department of Physics, Harvard University

November 10, 2022

I am a Post-Doctoral fellow in Mikhail Lukin's group at the department of physics at Harvard University with an interest in understanding the connection between computing and physics. The majority of my current research is about understanding the computational hardness and its relation to quantum physics, which I believe is the key to understanding the nature of computation. In the following, I will explain my works in the past two years mainly from two aspects, one is understanding the computational hardness from the solution space properties, and another is embedding computational-hard problems in a physical system.

1 Current Works

1.1 Solution space properties of hard combinatorial optimization problems

In the past two years, I have been most dedicated to creating a unified framework to solve the *solution space properties* [?] of a class of hard combinatorial optimization problems. Here, the solution space property refers to a class of quantities that not only include the maximum or minimum set size, but also include the number of sets at a given size, enumeration of all sets at a given size, and direct sampling of such sets when they are too large to be fit into memory; the class of problem includes but is not limited to the independent set problem, the maximum cut problem, the vertex coloring problem, the maximal clique problem, the dominating set problem, and the satisfiability problem, among others. The framework I and my collaborators created is called generic tensor network, where the word "generic" comes from generic programming in computer science. While the relationship between a counting problem and a tensor network is well known, these works did not catch much attention of computational scientists due to their limited use cases. We take a different view of a tensor network and show how different solution space properties can be computed with the same program by elevating the tensor element algebra to commutative semiring algebras. We show the real algebra is related to the counting of all solutions, the (extended) tropical algebra is related to the largest solution size(s), the polynomial algebra is related to the graph polynomials, the truncated polynomial algebra is related to the degeneracy of solutions with largest or smallest several sizes, the bit string algebra is related to finding one best solution, and the set algebra is related to solution enumeration.

I created an open-source package GenericTensorNetworks (will be public on Github soon) that might benefit people in the field of computational complexity and statistic physics. During the development of this package, I also contributed a lot to the open-source community. I and Chris Elrod wrote the package TropicalGEMM for fast tropical matrix multiplication, which has a very close to the theoretical optimal speed, i.e. half the speed of floating-point number; I rewrote the tensor permutation in Julia base and CUDA to speed up the high-rank tensor manipulation; I wrote the package OMEinsumContractionOrders for the state of the art `einsum` contraction order optimization.

This project is highly motivated by another experimental project that I participated in, where we use variational quantum algorithms to solve the maximum independent set (MIS) problem on a diagonal-coupled unit-disk grid graph (DUGG) by embedding a problem instance into a Rydberg atom array Hamiltonian [?]. In this experiment, it is crucially important to explain why our quantum algorithm works better than classical simulated annealing in some graph instances but not in others. The results found by using the generic tensor network deepened our understanding of the MIS experiment. By checking the degeneracies of independent set solutions at different sizes, we find that the degeneracy ratio can be a good indicator of the classical hardness of a problem instance. By inspecting the configuration space connectivity, we find the absence of the overlap-gap-properties [?] in most of the target problems. Here, the overlap gap property is an important feature of a solution space geometry, the presence of which indicates a no-go for a local-search-based algorithm. These solution space properties that helped understand quantum and classical algorithms are not feasible in the previous frameworks that mainly targeted on best solutions. Our method fills the gap between finding one solution for a hard problem and understanding a hard problem.

1.2 Solving the maximum independent set problem with Rydberg atom arrays

I and my collaborators proposed a scheme to reduce the MIS problem on a general graph to one on a DUGG and showed this reduction is very likely to be optimal [?]. The proposed method only introduce an overhead that is linear to the pathwidth of the source graph, where the pathwidth is a graph characteristic that is upper bounded by the number of vertices. This result may have impacts on both fields of quantum computing and computational complexity.

In the quantum computing field, it is a crucial step toward solving the MIS problem on a general graph with quantum algorithms implemented on Rydberg atom arrays that have highly constrained two-dimensional geometry. In the previous MIS experiment paper, we studied the MIS problem on a DUGG. We showed although DUGGs have highly constrained topology, finding their MISs is nevertheless NP-complete. However the reduction in that paper from an MIS problem on a general graph to an MIS problem on DUGG has an overhead of n^8. If one wants to solve an MIS problem on a general graph with 100 vertices, which can be solved in a classical computer in milliseconds, the required number of qubits is $\sim 10^{16}$, which is a number not feasible in the near term. Our new mapping scheme resolved these concerns.

In the computational complexity field, it can bridge some important results about the MIS problem. For example, a lemma of our reduction is: the MIS size of a unit-disk graph is hard to approximate with an error smaller than $\sqrt{n}$ because the MIS size of a general graph is hard to approximate within $n^{1-\epsilon}$ [?].

The paper for this work as well as an open source software will be released soon.

2 Future Works

My wildest dream is to understand the nature of computation from a physicists perspective. So I propose the following two research directions for the next 5 years.

2.1 Energy efficient computational frameworks

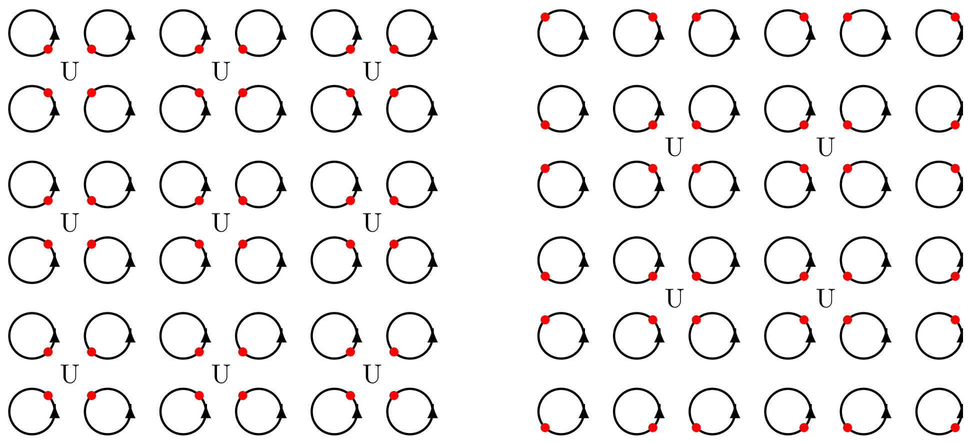
One of the most important goals of quantum physics is building new computational frameworks that can be much better than traditional CMOS based technologies. It can be better in reducing the energy consumption, making use of quantum entanglement to solve hard problems or both. These two goals are often not very compatible in traditional circuit based quantum computing due to the requirement of external classical controls. If one wants to apply a quantum gate with precision ϵ, the energy consumption is expected to scale as $1/\epsilon$. I want to build an energy efficient computational framework without external classical control. In the following, I will sketch a possible approach based on quantum cellular automata for energy efficient quantum computing.

2.1.1 Reversible Cellular Automata on Rydberg atom arrays

Using physics like computing model based on block cellular automata does not require an external classical control and is potentially much more energy efficient. So I propose implementing quantum cellular automata in Rydberg atom arrays as a practical approach towards universal quantum computing for the following reasons

1. Cellular automata potentially can avoid using local pulses, which is costly in Rydberg atom arrays,
2. Cellular automata does not require runtime atom moving, which is costly in Rydberg atom arrays,

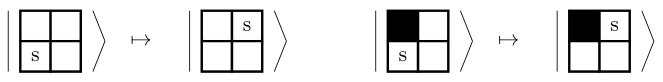
3. Cellular automata can parallelize computations easily.



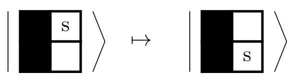
(a) even time step (b) odd time step

Figure 1: Time dependent optical traps to drive atoms (red dots) moving in a circle.

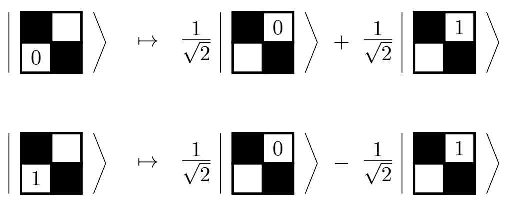
To build a block quantum cellular automata (BQCM), ① we first prepare two piles of Rydberg atoms in hyperfine states $|0\rangle$ and $|1\rangle$. ② Then initialize the configuration by moving atoms to a bi-layer square lattice. We map two atoms at the same x, y-coordinate to a qudit, and each qudit has 4 possible states $|00\rangle = 0$, $|01\rangle = 1$, $|10\rangle = \square$ and $|11\rangle = \blacksquare$. We use these qudits to encodes both the program and the input data. ③ Drive the atoms on the square lattice with periodic optical traps/tweezers as shown in Fig. **??** and make them oscillate with period T. Each time four atoms move closest to each other, the state will be updated by U. Where U is a unitary gate that implements the rules specified in Fig. **??**. After certain steps m, the results can be read out from the final configuration.



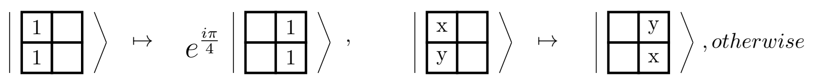
(a) Signal propagation



(b) Bouncing



(c) Hadamard gate



(d) CPhase($\frac{\pi}{4}$)

Figure 2: Building blocks of quantum cellular automata in Ref. [**?**]. These rules are rotational symmetric.

There is a remaining challenge, that is the global pulses alone do not have the ability to implement the desired 4-qudit gate, it can only implement certain gates. In order to avoid local addressing, we need to either schedule

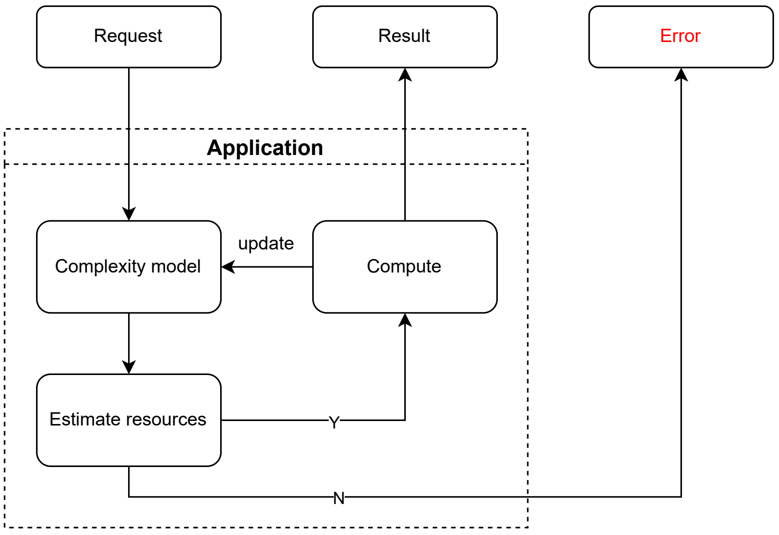


Figure 3: Algorithm Hub for scientific computing.

the atoms moving cleverly so that we can apply single qubit gates partially or design a Rydberg friendly quantum cellular automata model.

2.2 A unified framework for resource estimation for computing hard problems

It is estimated that 5-25 percent of all energy consumption in the world goes to running computing devices. A significant amount of them goes to scientific computing, for example, both training a neural network for generative language modeling [?] and classical simulation [?] and benchmarking [?] quantum algorithms can cost energy that worth million dollars. However, for many of them, the energy consumption can be significantly reduced by calling the correct algorithm and its best implementation. For example, with the state of the art tensor network contraction order finding algorithm, classical algorithms can beat google supremacy circuit with much lower energy cost [?].

I want to build an algorithm hub to help researchers and industry users to choose the best algorithms. As illustrated in Fig. **??**, the algorithm hub is a web application that containing both the algorithm implementations and their complexity models to estimate computational resources from inputs. For example the DMRG algorithm can be specified as an matrix product operator with bond dimension c, the maximum bond dimension D, the size of physical degree of freedoms d, the length of chain L and the number of sweeps m. The developer can input the time complexity as $const. \times L \times m \times D^3 \times d^2 \times c$ and space complexity $const. \times L \times d \times D^2$. The application server does not know whether this complexity model is correct or not, so the best thing it can do is trust the developer and keep improving it automatically.

To use the service, a user posts a request with the input arguments to the server. Then the server estimate the required resource from the input arguments and does the computing if the required resource is available. Computing time and space will be collected for improving the complexity model, e.g. tuning the constant factor, additive overhead and the exponents.

To summarize, building an algorithm hub can

- save energy resources for human beings,
- save time for researchers and industry users,
- benchmark and polish the complexity model for developers,
- reward open source scientific computing software developers properly.

I am the right one for building an algorithm hub because I have been developing several popular open source packages with good reputations, and I know the need and the best approach to achieve this goal. Technically, developers can release their application through a GitHub continuous integration, which provides a proper version control. Julia language can be the host language for algorithms, because it is open source, high performance language designed for scientific computing. Also, the network base algorithm model solves the problem of Julia's just in time compiling overhead since the web service can reuse the same Julia session.

3 Summary

To summarize, at this point in my career, my primary interests are computational-hard problems and near-term quantum algorithms. In the future, I want to understand the nature of computation and propose energy efficient computational frameworks. Except for these research interests, I want to devote half of my time to the help build up the open-source scientific software ecosystem. I believe programming is a proper way to formalize some part of human knowledge, and open-source software development is an integrable and collaborative practice in this direction.