permanent of n×n random unitary matrices as the function of the matrix size n.

Our implementation is four times faster than TheWalrus code executed on 24 threads of Intel Xeon Gold 6130 CPU processor. We also compared the numerical performance of the Piquasso Boost simulation framework to the benchmark of Ref. [3]. In this case our benchmark comes very close to the execution time achieved on a single node of the Tianhe-2 supercomputer consisting of an Intel Xeon E5 processor with 48 threads and three Xeon Phi 31S1P cards with 684 threads in total. Our results indicate that the Piquasso package provides a high performance simulation framework even on smaller hardware. Our recursive implementation scales well on shared memory architectures, however, its scalability over distributed computational resources is limited [L6].

To efficiently perform permanent calculations on large scale computing resources we need to come up with an alternative approach. Here we report on a data-flow engine (DFE) based permanent calculator device. We developed a full-fledged permanent calculator implementation on Xilinx Alveo U250 FPGA cards using high-level data-flow programming tools developed by Maxeler Technologies. The data-flow engines are driven by the CPU of the host machines providing a high scalability of our implementation over MPI communication protocol: it is possible to divide the overall computational problem into chunks and distribute them over the nodes of a supercomputer cluster just like in the case of the CPU implementation of [3].

Our FPGA-based DFEs have several advantages over CPU and GPU technologies. Probably the most important aspect of FPGA is the possibility to have hardware support for arithmetic operations exceeding the precision of 64-bit arithmetic units of CPU and GPU hardware. In practical situations the permanents of unitary matrices describing the photonic interferometers would be much smaller than the individual elements of the input matrix. In this situation one needs to increase the numerical precision in the calculations to obtain a result that can be trusted. In fact, the implementations of the walrus package and the Piquasso Boost library already use extended precision for floating point operations on the CPU side to calculate the permanent. On the DFE side we used 128-bit fixed point number representation to perform the calculations, providing the most accurate result among the benchmarked implementations.

Figure 1 also shows the performance of our DFE permanent calculator compared to the previously discussed CPU implementations. We observe that on DFE we can calculate the permanents of larger matrices much faster than by CPU based implementations with a numerical precision exceeding them. Note that at smaller matrices the execution time is dominated by the data transfer through the PCI slot.

In our future work we aim to use our highly optimized Piquasso simulations framework to address computational problems related to BS and examine the possibility of using BS quantum devices to solve real world computational problems. Our efficient method to evaluate the permanent would be valuable in other research works as well, making the evaluation of the amplitudes of many-body bosonic states faster and more reliable.

This project has received funding from the Ministry of Innovation and Technology and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary.

Links:

[L1] https://qi.nemzetilabor.hu/
[L2] https://kwz.me/h9H
[L3] https://kwz.me/h9E
[L4] https://www.maxeler.com/
[L5] https://kwz.me/h9K
[L6] https://arxiv.org/abs/2109.04528

References:

[1] S. Aaronson, A. Arkhipov: "The computational complexity of linear optics", in Proc. of STOC'11 https://dl.acm.org/doi/10.1145/1993636.1993682
[2] J. Wu, et al.: "A benchmark test of boson sampling on Tianhe-2 supercomputer", National Science Review, Volume 5, Issue 5, September 2018, Pages 715–720, https://doi.org/10.1093/nsr/nwy079

Please contact:

Peter Rakyta
Eötvös Loránd University, Hungary
peter.rakyta@ttk.elte.hu

Some Complexity Results Involving Quantum Computing

by Gábor Ivanyos, Attila Pereszlényi and Lajos Rónyai (ELKH SZTAKI, BME)

The Theory of Computing Research Group of the Informatics Laboratory at ELKH SZTAKI has studied quantum algorithms for various computational problems. We outline two projects in this area: one in computational algebra and one in machine learning.

Computing with black box groups

We are investigating possible applications of quantum computing to algorithmic problems from algebra and arithmetic. Examples of algorithms of this kind include Shor's famous quantum algorithms for factoring integers and computing discrete logarithms. The method for the discrete logarithm actually works in black box groups with unique encoding of elements.

The notion of black box groups was introduced by Babai and Szemerédi to study the complexity of problems related to the structure of matrix groups. The elements of a black box group are encoded (represented) by binary strings and the group operations are given by oracles (also called black boxes). In order to capture factor groups, they

allowed the same element to be represented by more than one string and they added a further oracle for testing equality with the identity element. In a recent manuscript [1], with our colleagues from France and Singapore, we studied the complexity of the discrete logarithm problem in an Abelian black box group with non-unique encoding of elements. We assumed encoding of the group elements by a covering group in a natural way. It turns out that in this setting the quantum query complexity becomes exponential.

The same holds for the closely related, possibly easier, computational Diffie-Hellman problem, which is the following: we are given three elements of the group: g, g^a and g^b, where the exponents a and b are hidden, compute g^{ab}. In the decision version, four group elements are given: g, g^a, g^b, and g^c and the task is to decide if $g^c=g^{ab}$. We showed that if the elements of a cyclic group of order p are encoded by the elements of a covering group of rank two, then, while the computational Diffie-Hellman problem is hard even for a quantum computer, the decisional version can be solved in polynomial time with a classical algorithm. Curiously, this difference disappears in higher ranks: even with encoding by a group of rank three both the decisional and computational problems have exponential quantum query complexity.

Quantum- and classical machine learning

Recent results in quantum machine learning show that many proposed quantum algorithms do not have much benefit over classical algorithms. We investigated the quantum classifier of Schuld and Petruccione [2] that selects a weak learner according to a distribution based on how good they are. It can be interpreted as a stochastic boosting algorithm on a finite set of learners where the learners are determined in advance. We simplified their algorithm to the point where it is intuitively easy to give an equivalent classical algorithm. We showed that a simple classical randomised method achieves the same result without changing the time complexity. We also gave an even simpler, constant time classical algorithm. We showed that this quantum ensemble method has no advantage over classical algorithms.

Independently from our work, Abbas, Schuld, and Petruccione also showed that the ensemble method can be turned into a classical algorithm. Our construction, however, is arguably simpler and more direct, especially the constant time algorithm.

We further developed the idea and, as the main contribution of our paper [3], we proposed classical methods that are inspired by combining the quantum ensemble method with adaptive boosting. We compute two types of weights in an alternating way: one on the samples, representing how difficult they are to learn and one on the learners, representing how well they perform on the chosen samples. We considered only the case of binary classification, but the methods can be extended. In our experiments we had different implementations of the above idea. We tested the algorithms on publicly available datasets and we found them comparable to the AdaBoost algorithm, which is one of the most efficient meta machine learning algorithms.

References:

[1] G. Ivanyos, A. Joux and M. Santha: "Discrete Logarithm and Diffie-Hellman Problems in Identity Black-box Groups", arXiv:1911.01662.

[2] M. Schuld, F. Petruccione: "Quantum ensembles of quantum classifiers", Sci Rep. 2018 Feb 9;8(1):2772. doi: 10.1038/s41598-018-20403-3.

[3] B. Daróczy, et al.: "Quantum Inspired Adaptive Boosting", arXiv:2102.00949

Please contact:

Gábor Ivanyos and Lajos Rónyai
ELKH SZTAKI, Hungary
gabor.ivanyos@sztaki.hu,
lajos.ronyai@sztaki.hu

Quantum Algorithms for Quantum and Classical Time-Dependent Partial Differential Equations

by François Fillion-Gourdeau (Institute for Quantum Computing and Infinite Potential Laboratories)

Quantum algorithms have been developed to solve two important classes of partial differential equations: the Dirac equation and linear symmetric hyperbolic systems of equations. These algorithms can be much more efficient than their classical counterparts.

Quantum computing is at the heart of a new revolution in information science where quantum effects and quantum states are leveraged to perform calculations. In recent decades, significant scientific resources have been devoted to this subject. A functional quantum computer could solve computational problems that are unsolvable on a classical computer, owing to quantum algorithms being much more efficient for a large class of problems, and giving a polynomial or an exponential speedup over classical ones.

Since the seminal works of R. Feynman in the 1980s, there has been growing evidence for the efficiency of quantum algorithms to simulate the time-evolution of quantum systems. However, this quantum advantage is not as clear for classical systems modelled by time-dependent partial differential equations (PDE). In 2015, we started to work on this topic to determine if some PDEs could be solved on existing quantum computer prototypes, even though the latter have limited resources.

Simulating the time-evolution of a physical system on a quantum computer requires two main ingredients: i) a mapping of the solution on the quantum register and ii) a mapping of the time-evolution operator on a set of quantum logic gates. The quantum register is