Exploring the use of parameterised complexity in quantum computing

Alexis Shaw[1*], Michael Bremner[1†], Ryan Mann[1], Luke Mathieson[2], and Ashley Montanaro[3]
1) Centre for Quantum Software and Information, Centre for Quantum Computation and Communication Technology, University of Technology Sydney | 2) Faculty of Engineering and IT, University of Technology Sydney | 3) School of Mathematics, University of Bristol
† michael.bremner@uts.edu.au
* alexis.t.shaw@student.uts.edu.au

CENTRE FOR QUANTUM COMPUTATION & COMMUNICATION TECHNOLOGY
AUSTRALIAN RESEARCH COUNCIL CENTRE OF EXCELLENCE
Australian Government
Australian Research Council
UTS:QSI
CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

Classically assisted quantum computing

Quantum Computers are usually envisioned as a specialist accelerator that runs alongside and under the control of a higher clocked classical machine. Normally we can ignore the power of this classical computer because quantum computers can simulate classical behaviour. In the NISQ (Near Term Intermediate-scale Quantum computing) regime however, due to the non-availability of quantum error correction, this does not necessarily hold, because quantum circuits are not able to scale to polynomial gate counts. In order to formally understand the benefits of NISQ quantum accelerators we need a new approach. One way to understand these combined systems is by the use of oracles.

In an oracle model we say that one complexity class (e.g. $\boldsymbol{BPP}$) is augmented to be able to query results from another class (e.g. $\boldsymbol{QNC_0}$) as an instantaneous action. We call this combination

$$\boldsymbol{BPP^{QNC_0}}$$

This work examines the computational power if we replace $\boldsymbol{BPP}$ and $\boldsymbol{QNC_0}$ above with other classes. Particularly allowing the base computer to be given limited exponential power as in the classical analysis of Parameterised Complexity.

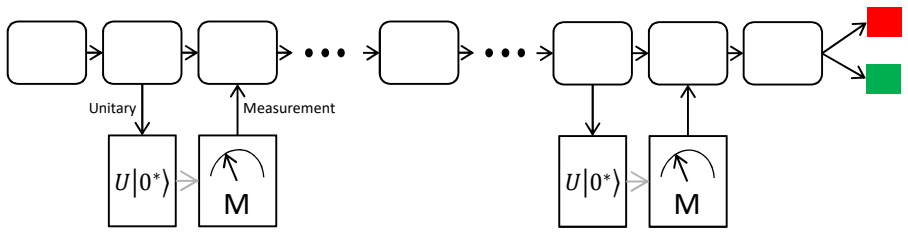


Figure 1: The flow of execution in a quantum oracle machine

Why combine parameterised complexity and quantum computing?

In the NISQ era of quantum computing, and beyond classical computers associated with quantum computers will be able to do significant amounts of computation, both due to the intrinsic high expense of quantum hardware (especially deep cryogenic refrigeration), but also due to the high classical computation costs required for implementing QEC (Quantum Error Correction). However with this amount of classical computation available, it is not strictly true that only problems in $\boldsymbol{BPP}$ can be tackled by the classical systems. Limited amounts of exponential computation is likely to be tolerable, both due to the relatively slow speed of the quantum computer, and due to the relative cost difference between classical and quantum circuits.

One of the best models available to theoretical computer science as to what can be solved on such a system is parameterised complexity. If a problem is $\boldsymbol{FPT}$ it is likely that a classical computer will be able to solve it in a reasonable amount of resources. However it is unlikely that a classical machine will be able to solve every problem in $\boldsymbol{NP}$.

There are two ways in which one may envisage the use of $\boldsymbol{FPT}$ classical resources could be used to augment a NISQ quantum machine. In one, we have both a quantum oracle $\boldsymbol{Q}$ (e.g. an $\boldsymbol{NC_0}$ oracle) and an $\boldsymbol{FPT}$ oracle being used by a probabilistic machine. i.e.

$$\boldsymbol{BPP^{FPT+Q}}$$

In this scenario only polynomial queries to the quantum machine are allowed, whilst exponential calculations are carried out in parallel. This is likely a good model for NISQ computation. The other scenario is more forward looking, where we have an $\boldsymbol{FPT}$ controller making queries to a (fast) Quantum machine. i.e.

$$\boldsymbol{PFPT^{Q}}$$

In this case we are able to make exponentially many queries on the quantum state. This may allow some additional power than above, as state estimation is no-longer limited to polynomial numbers of queries. Whether this has any practical application is yet to be seen.

It may also be possible to apply kernelization style techniques to reduce the number of qubits required to encode some hard problems onto a quantum machine, for example for accelerating a classical algorithm using Grover-like techniques. However it seems that this may have little near-term applicability.

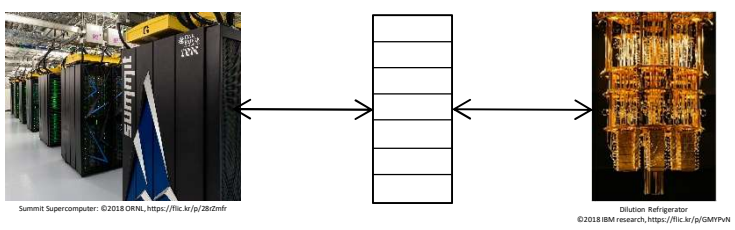


Figure 2: Can NISQ computation leverage classical supercomputing resources?

Parameterized complexity

In the early 1990s it was discovered that many $\boldsymbol{NP}$−hard do not permit a polynomial time approximation scheme, this includes SAT, Independent set, and the traveling salesman problem. These results meant that other approaches to finding solutions to hard optimization problems were required. One approach due largely to Robert Downey and Michael Fellows was to try to limit the exponential growth of the algorithm to something other than the size of the input. This is called Parameterized Complexity.

Parameterized problems

The first concept to be formalized was that of a *Parameterized Decision Problem*. This is a problem where the description gives some number in addition to the structure under investigation. The existence of such a number is often natural in the conversion of an optimisation problem into a decision problem. Formally this is done as usual using the concept of a Parameterized Language.

Definition: Parameterised Language
An element (X, k) of a parameterized language $L \subseteq \Sigma^\star \times \mathbb{N}$ is a tuple consisting of a string X and an integer parameter k (often defined to be represented in unary).

Kernelizations

The next observations was that in many cases heuristics worked very well, when they can be applied. This observation was formalized into the concept of kernelization defined as follows:

Definition: Kernelization
A kernelization $K : L \to L$ of a parameterised language $L \subseteq \Sigma^\star \times \mathbb{N}$ is an algorithm that maps $(x, k) \to (x', k')$ such that:

- $(x, k) \in L$ if and only if $(x', k') \in L$
- $|x'| \in O(f(k))$ for some fixed function f only defined in terms of k
- k' is bounded by some function in k

If f is a polynomial then K is called a polynomial kernel, if $K \in P$ then it is a polynomial time kernelization. Unless stated otherwise a kernelization is assumed to be polynomial time, as any problem in $\boldsymbol{FPT}$ has a trivial $\boldsymbol{FPT}$-time kernelization

Fixed Parameter Tractability (FPT)

More generally than Kernelizations, we can ask what can be done if we only allow non-polynomial growth to occur in terms of k, with only a polynomial term in $|n|$ being allowed. The complexity class $\boldsymbol{FPT}$ is defined as

$$\boldsymbol{DTIME}(O(f(k)poly(|x|)).$$

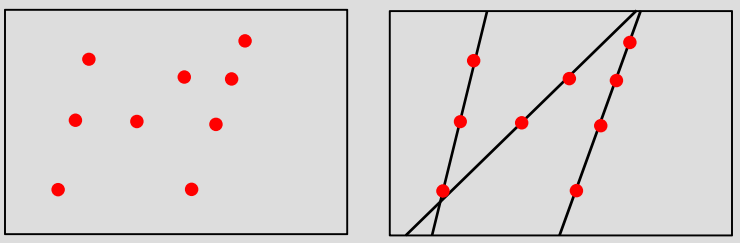
Where $f(k)$ is allowed to be any computable function. Some NP-hard problems like *Steiner Tree* and *Multicut* have a reasonable $\boldsymbol{FPT}$ algorithm, but do not admit a polynomial-time polynomial kernel, others like *vertex cover* have a known $\boldsymbol{FPT}$ algorithm that is (much) faster than just an application of their kernelization. Yet more problems like *Dominating Set* are believed to not have any $\boldsymbol{FPT}$ solution.

Example: Point-Line Cover

Input
- A set of points S taken from the plane.
- An integer k

Question
Is there a collection L of at most k lines such that L covers the points in S?

Kernelization
1. Let $k' = k, S' = S$
2. For every pair $X_1, X_2 \in S'$
3. Count the number n of points that lie on the line l through X_1 and X_2
4. If $n \geq k'$ remove all the points covered by l from S' and reduce k' by 1
5. If $S' \geq (k')^2$ or $k' < 0$ then return a trivial no-instance as a kernel, otherwise (S', k') is a kernel for (S, k).



References

Nielsen, Michael A., and Isaac Chuang. "Quantum computation and quantum information." (2002): 558-559.
Downey, Rodney G., and Michael R. Fellows. *Fundamentals of parameterized complexity*. Vol. 4. London: Springer, 2013.
Aaronson, Scott, Greg Kuperberg, and Christopher Granade, Vincent Russo. "The complexity zoo.", University of Waterloo, 2005 – 2019, https://complexityzoo.uwaterloo.ca/Complexity_Zoo
Kratsch, Stefan, Geevarghese Philip, and Saurabh Ray. "Point line cover: the easy kernel is essentially tight." ACM Transactions on Algorithms (TALG) 12.3 (2016): 40.