Circuit Extraction for ZX-Diagrams Can Be #P-Hard

Niel de Beaudrap
University of Sussex, UK

Aleks Kissinger
University of Oxford, UK

John van de Wetering
Radboud University Nijmegen, The Netherlands
University of Oxford, UK

Abstract

The ZX-calculus is a graphical language for reasoning about quantum computation using ZX-diagrams, a certain flexible generalisation of quantum circuits that can be used to represent linear maps from m to n qubits for any $m, n \geq 0$. Some applications for the ZX-calculus, such as quantum circuit optimisation and synthesis, rely on being able to efficiently translate a ZX-diagram back into a quantum circuit of comparable size. While several sufficient conditions are known for describing families of ZX-diagrams that can be efficiently transformed back into circuits, it has previously been conjectured that the general problem of *circuit extraction* is hard. That is, that it should not be possible to efficiently convert an arbitrary ZX-diagram describing a unitary linear map into an equivalent quantum circuit. In this paper we prove this conjecture by showing that the circuit extraction problem is **#P**-hard, and so is itself at least as hard as strong simulation of quantum circuits. In addition to our main hardness result, which relies specifically on the circuit representation, we give a representation-agnostic hardness result. Namely, we show that any oracle that takes as input a ZX-diagram description of a unitary and produces samples of the output of the associated quantum computation enables efficient probabilistic solutions to NP-complete problems.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases ZX-calculus, circuit extraction, quantum circuits, #P

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.119

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Previous Version*: `https://arxiv.org/abs/2202.09194`

Funding *Aleks Kissinger*: Acknowledges support from AFOSR grant FA2386-18-1-4028.
John van de Wetering: Acknowledges support from an NWO Rubicon Personal Grant.

Acknowledgements The authors wish to thank the anonymous reviewers for their valuable suggestions regarding the presentation of this paper.

1 Introduction

Quantum circuit notation is widely used in the field of quantum computing to denote computations to be executed on a quantum computer. While quantum circuits are a useful tool for representing computations on a quantum computer, they are somewhat inconvenient for reasoning about computations (such as proving equalities or doing simplifications); and for representing computations in alternative models like the one-way model of measurement-based quantum computation (MBQC) [40], or surface code lattice surgery [31].

49th International Colloquium on Automata, Languages, and Programming (ICALP 2022).
Editors: Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff;
Article No. 119; pp. 119:1–119:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ZX-diagrams are an alternative, more general representation of quantum computations, which allow complex operations to be described using a few simple generating operators. ZX-diagrams come with an equational theory, called the *ZX-calculus* [11], which allows one to perform many useful calculations graphically, without resorting to concrete matrix computations. While ZX-diagrams can be seen as an extension of circuits [12], they also readily admit encodings of the one-way model [23] and lattice surgery [20], and allow one to reason more easily about such procedures. There are several known *complete* axiomatisations of the ZX-calculus [37, 49], where any true equality of linear maps can be proved graphically. For a review on the ZX-calculus we refer to [47].

The ZX-calculus has been used in a variety of areas. It was used to optimise T-count [34, 18], braided circuits [29] and MBQC [4]; to find a new normal form for Clifford circuits [22]; to do more effective classical simulation using stabiliser decompositions [35]; and to reason about surface codes [26, 27], mixed-state quantum computations [9], natural language processing [10], condensed matter systems [14], counting problems [21, 44] and spin-networks [24].

As a strict extension of quantum circuit language, ZX-diagrams may express operations in a form that do not correspond directly to a quantum circuit. This added flexibility makes it easier to find novel strategies to simplify quantum circuits, but it comes at a cost: given a ZX-diagram representing a unitary linear map, it might be non-trivial to transform it back into a circuit of comparable size. Such a translation might however be necessary if, for instance, we want to run the computation described by a ZX-diagram on a gate-based quantum computer.

We refer to the above problem, as the *circuit extraction* problem: given a ZX-diagram which denotes a unitary operator U, find a unitary circuit (*i.e.*, a quantum circuit without measurements) that implements U. In recent years, some progress has been made on this problem [22, 34, 4, 41, 33, 19]. However, all known methods for efficient extraction of circuits from ZX-diagrams rely on additional conditions, in particular requiring there to be some kind of *flow* on the diagram, a concept imported from MBQC [6]. Such conditions allow the diagram to be rewritten incrementally into a unitary circuit. Since many ZX-calculus rewrites preserve these conditions, it is possible to perform optimisation of quantum circuits using ZX-calculus rules and still recover circuits efficiently.

However, it is worth trying to generalise these conditions as much as possible, or even remove them. For instance, it was noted in [34] that a certain transformation of ZX-diagrams would decrease the T-count (an important metric for quantum circuit optimisation), but in the process broke the invariant (the existence of a gflow), preventing a circuit from being extracted efficiently using known techniques. Given all this it is then natural to wonder about the following question:

Is there some efficient procedure to translate any
unitary ZX-diagram into a quantum circuit?

In this paper we present strong evidence that there is no such efficient procedure, by showing that the circuit extraction problem is **#P**-hard in the worst case. The complexity class **#P** contains for instance the problem of strong simulation of quantum circuits, and counting the number of satisfying solutions to a Boolean formula, so **#P**-hard problems are expected to be intractable. We prove **#P**-hardness by giving an encoding of Boolean formulae into unitary ZX-diagrams in such a way that extracting a polysize circuit provides a solution to the associated **#SAT** instance. A consequence of our result is that if there were a polynomial time algorithm for circuit extraction, then $\mathbf{P} = \mathbf{NP}$.

Alternatively, since there is an evident translation from a ZX-diagram into a quantum circuit with postselection, this result can equivalently be seen as expressing the hardness of translating a postselected circuit that is promised to be proportional to a unitary into a circuit without postselection. While intuitively this seems likely to be hard, particularly in light of Aaronson's landmark result that **PostBQP** = **PP** [1], our hardness result seems to be quite different in nature due to the unitarity promise. In particular, the postselection does not seem to be the "source of power" in our proof: the measurement outcomes corresponding to the post-selections in our circuits occur with some bounded probability, independent of the problem size.

One could ask how much our hardness result is tied to the fact that we require a procedure that produces quantum circuits from ZX-diagrams. Especially, when considering that in most cases we are not interested in the circuit itself, but instead we simply want to sample the output of the quantum computation. Perhaps one could find some other procedure to "program" a quantum computer using a ZX-diagram describing a unitary and obtain samples of measurement outcomes. We show that an efficient such procedure is unlikely to exist for arbitrary ZX-diagrams, by finding that such a procedure allows you to probabilistically solve **NP**-hard problems. So if there were some way to generically translate unitary ZX-diagrams into procedures which could be realised in polynomial time on a quantum computer, it would follow that the entire polynomial hierarchy is in **BQP**, and in particular that **NP** ⊆ **BQP**.

The paper is structured as follows. We start by covering preliminaries on quantum circuits, ZX-diagrams and the necessary complexity theory in Section 2. Then in Section 3 we formally define the circuit extraction problem and prove it is hard. Section 4 considers several variations on circuit extraction, and in Section 5 we find some upper bounds on the hardness of circuit extraction. We end with some concluding remarks in Section 6.

2 Preliminaries

2.1 Quantum circuits

Since we wish to extract "a circuit" from a ZX-diagram, it will be helpful to first consider what we actually mean by a circuit.

In quantum computational theory, a "circuit" is a description of a computational process consisting of operations which may be decomposed as a sequence of primitive "gates" and "measurements", which act on one or more qubits to change the states of those qubits. The state-space of a qubit is identified with unit vectors of the finite-dimensional Hilbert space $\mathcal{H}_2 \cong \mathbb{C}^2$; the state of k qubits in parallel is described by the tensor product $\mathcal{H}_2^{\otimes k}$. A "gate" is an operation which is applied to one or more qubits and implements a unitary transformation $U : \mathcal{H}_2^{\otimes k} \to \mathcal{H}_2^{\otimes k}$ on the associated state space. A "measurement" is an operation which transforms a state $|\psi\rangle \in \mathcal{H}_2^{\otimes k}$ to some state $p_j^{-1/2}\,\Pi_j|\psi\rangle$ where $\{\Pi_1, \Pi_2, \ldots\}$ is a set of projections that sum up to the identity operator I, the p_j gives the probability of observing that particular measurement outcome and is given by $p_j = \langle\psi|\Pi_j|\psi\rangle$, and the index j provides the classical "outcome" indicating which transformation occurred. A gate or measurement acting on a small number of qubits can be applied to a larger set of qubits by taking the tensor product with an appropriate number of identity operators. A "circuit" is then a composition of such gates and measurements on some number of qubits, acting in sequence or in parallel, to describe more complex (and in general, non-deterministic and irreversible) transformations of a quantum state-space. To define a reasonable model of computational complexity using quantum circuits, one usually elaborates the above with a description of how one would specify a circuit as part of a family of unitary operators, acting on inputs of various sizes. For our purposes, it will suffice to require that the coefficients of the gates be efficiently computable, and in particular provided explicitly in some representation which suffices to approximate them to $O(\text{poly}(n))$ bits of precision in time $O(\text{poly}(n))$ for an n qubit circuit.

It will be convenient to refer to one specific such gate-set – an infinite set $\mathcal{B}$ of gates, consisting of the single-qubit gates Z_α for arbitrary angles α, the single-qubit Hadamard gate H and the two-qubit gate CNOT:

$$Z_\alpha \;=\; \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\alpha} \end{pmatrix} \qquad H \;=\; \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \mathrm{CNOT} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1}$$

This gate set forms a *universal* gate set, meaning that a unitary acting on any number of qubits can be written as a circuit consisting of these gates [38]. Other universal gate-sets exist, but so long as one considers gate sets whose parameters are efficiently computable from some input parameters and which act only on a bounded numbers of qubits (*e.g.*, at most two or three qubits), the size of a circuit to represent a given unitary operator can only vary by a constant factor, so that for the purpose of complexity theory, the details of the specific gate set chosen are not important.

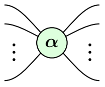
A circuit which contains no measurements, and therefore consists entirely of unitary gates, is called a “unitary circuit”. A unitary circuit is reversible, and “deterministic” in the sense that an idealised realisation of such a circuit will transform the state-space in the same way each time. As this is a convenient feature for the design and analysis of quantum algorithms, much of the literature on quantum algorithms concerns itself with unitary circuits, and much of the design of quantum computers is concerned with how to reliably implement unitary circuits.

2.2 ZX-diagrams

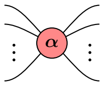
We provide a brief overview of ZX-diagrams. For a review see [47], and for a book-length introduction see Ref. [13].

ZX-diagrams form a diagrammatic language similar to the familiar quantum circuit notation. A *ZX-diagram* (or simply *diagram*) consists of *wires* and *spiders*. Wires entering the diagram from the left are *inputs*; wires exiting to the right are *outputs*. Given two diagrams we can compose them by joining the outputs of the first to the inputs of the second, or form their tensor product by simply stacking the two diagrams [11, 12].

Spiders are linear operations which can have any number of input or output wires. There are two varieties: Z-spiders depicted as green dots and X-spiders depicted as red dots:



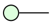
$$:= \; |0\cdots0\rangle\langle0\cdots0| \,+\, \mathrm{e}^{i\alpha}\,|1\cdots1\rangle\langle1\cdots1|$$

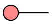


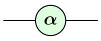
$$:= \; |{+}\cdots{+}\rangle\langle{+}\cdots{+}| \,+\, \mathrm{e}^{i\alpha}\,|{-}\cdots{-}\rangle\langle{-}\cdots{-}| \tag{2}$$

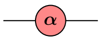
Here $|0\rangle$ and $|1\rangle$ represent the standard basis vectors of $\mathbb{C}^2$ which are the eigenvectors of the Pauli Z matrix; the states $|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$ are sometimes referred to as the “Hadamard basis”, and are eigenvectors of the Pauli X matrix. If you are reading this document in monochrome or otherwise have difficulty distinguishing green and red, Z spiders will appear lightly-shaded and X darkly-shaded. Note that here the number of inputs and outputs do not have to match. When $\alpha = 0$, we will not write the phase on the spider.

▶ **Example 2.1.** We can immediately write down some simple state preparations and unitaries in the ZX-calculus:

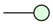
 $= \quad |0\rangle + |1\rangle \quad = \quad \sqrt{2}\,|+\rangle$

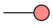
 $= \quad |+\rangle + |-\rangle \quad = \quad \sqrt{2}\,|0\rangle$

 $= \quad |0\rangle\langle 0| + e^{i\alpha}\,|1\rangle\langle 1| \quad =: \quad Z_\alpha$

 $= \quad |+\rangle\langle +| + e^{i\alpha}\,|-\rangle\langle -| \quad =: \quad X_\alpha$ (3)

We can also represent the effects that are dual to the states above using spiders:

 $= \quad \langle 0| + \langle 1| \quad = \quad \sqrt{2}\,\langle +|$

 $= \quad \langle +| + \langle -| \quad = \quad \sqrt{2}\,\langle 0|$ (4)

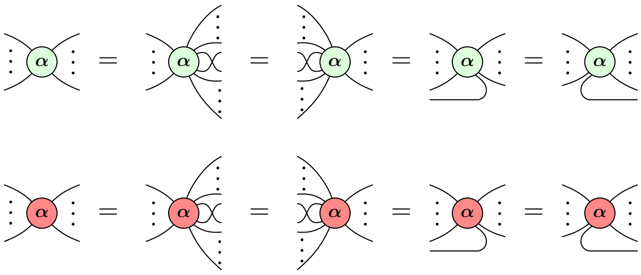
In the diagrams above we write explicit scalars to represent a proportionality constant. In this paper (non-zero) scalar factors will not be important. However, do note it is always possible to represent any scalar as an explicit ZX-diagram (of constant size). For this reason, our results will also apply to other proposed normalisations of the ZX generators, such as those in Refs. [17, 14, 24].

We can compose ZX-diagrams in two ways, either horizontally by connecting the wires together, which corresponds to the regular composition of linear maps, or vertically, which corresponds to the tensor product of linear maps. Any ZX-diagram is built by composing spiders (and permutations of wires) together in these ways.

On a formal level we consider a ZX-diagram D with n inputs and m outputs as a morphism $D : n \to m$ in a category. This category is the compact-closed *PROP* (symmetric monoidal category where the objects correspond to natural numbers and the tensor is addition) freely generated by the Z- and X-spider generators. The interpretation as a linear map is then a strongly monoidal functor into the category of Hilbert spaces, which is fully specified by the interpretation of the spiders (2). This level of formality won't be needed in this paper. The interested reader can look at for instance Refs. [11, 9, 8].

A more intuitive way to view ZX-diagrams is as tensor networks [39]: the spiders are the tensors, and a connection between spiders denotes a contracted index.

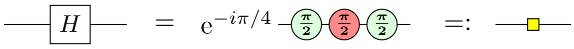
The Z- and X-spiders satisfy the following symmetries:



(5)

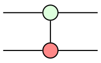
Here we are writing an equality of ZX-diagrams. This is to be understood as saying that the linear maps these ZX-diagrams represent are equal. In particular: in each equality the number of open wires extending to the left is the same, and similarly for the number of open wires extending to the right. Because of these symmetries we can treat ZX-diagrams as (labeled) undirected graphs: if we attach labeled nodes to the open wires at the input and output to distinguish their roles, arbitrary topological deformations of the diagram do not affect its interpretation as a linear map.

It is often convenient to introduce a symbol – a yellow square – for the Hadamard gate. This is defined by the equation:



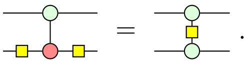
(6)

The CNOT gate also has a straightforward representation as a ZX-diagram:

$\mathrm{CNOT} \;=\; \sqrt{2}$

(7)

Here we are allowed to draw a horizontal wire as per the symmetries (5) whether this wire is an input or an output is irrelevant.

Seeing as we can represent Z_α, H and CNOT gates as ZX-diagrams, we can represent the gate set $\mathcal{B}$ of Eq. (1), and hence we can in fact represent any unitary as a ZX-diagram. The above demonstrates that ZX-diagrams can be used as an alternative representation for quantum circuits. However, ZX-diagrams are also more versatile than unitary circuits. Consider for example the following construction of the CZ gate as a ZX-diagram:

$\mathrm{CZ} \;\propto$


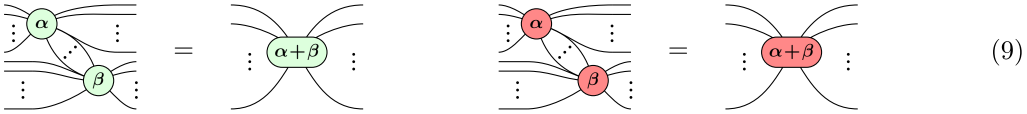
(8)

The right-hand-side demonstrates a different diagrammatic construction for CZ, that does not immediately look circuit-like, with the Hadamard-box representing some sort of interaction of two qubits rather than the evolution of a single qubit.

In fact, this versatility is reflected in the property that ZX-diagrams are universal for *all* linear maps between any number of qubits [11]. To see this, note that we can represent states as in Eq. (3). By composing tensor products of these states with some unitary we can write down any quantum state. By the map-state duality of quantum theory (i.e. the Choi-Jamiołkowski isomorphism), we can then also write every linear map, see for instance [47] for the details.

The universality of the gate-set $\mathcal{B}$ and of the ZX-calculus means that any unitary operator on some fixed number of qubits may be represented by some "gadget" in the ZX calculus, consisting of some fixed diagram of finite size – though as the example of CZ in Eq. (8) shows, there may also be "gadgets" which represent a unitary operator which do *not* consist of sequential and parallel composition of gates. Indeed, even the representation of the CNOT is by a simple "gadget" of two nodes, which is not describable as a composition of the other single-node "gadgets". In this respect, ZX-diagrams represent a more versatile notation than a conventional circuit notation. This raises the question of how, given a representation of some unitary U as a ZX-diagram, one might find another representation of U which consists of just compositions from the universal gate-set $\mathcal{B}$. This is the problem that this paper is concerned with.

ZX-diagrams are more than just a notation for unitary circuits (and non-unitary operators more generally): they may be used to perform computations. Specifically, ZX-diagrams come with a set of graphical rewrite rules, which may be used to find equivalent diagrams which represent the same state or operator, just as one might manipulate an algebraic expression. This rewrite system is *complete* [37, 49]: unlike other circuit diagrams, one may show that two equivalent ZX-diagrams are equivalent though transformations of diagrams alone. The possible advantage of this is that ZX-diagrams can often concisely represent operators which have a very large number of non-zero coefficients, and so that this reasoning can be done efficiently while it could not be done using the matrices directly. For instance, one of the rewrite rules we will use in this paper is *spider fusion*:



These rules say that we can fuse together adjacent spiders of the same colour.

While these rewrite rules are not immediately relevant to our results, the fact that it is possible to compute with ZX-diagrams is the motivation for considering this particular representation of unitary circuits, and also motivates the concept of considering different ZX-diagrams which represent the same unitary transformation. We refer the interested reader to [47] for an overview.

2.3 Circuit extraction

In the above section we saw that we can get ZX-diagrams directly from quantum circuits. We can also get ZX-diagrams from considering measurement patterns in the *one-way model* [40]. In the one-way model of quantum computation we start with a large *graph state*, on which we then do subsequent measurements, where the choice of measurement angle and axis may depend on previous measurement outcomes. This leads to another universal model of quantum computation. The one-way model can be straightforwardly represented in the ZX-calculus [23, 4].

An important property of a one-way computation is that we can perform a computation deterministically, so that we perform the same overall computation regardless of individual measurement outcomes. A sufficient property for ensuring that deterministic processes are possible on a given resource state is that its underlying graph has a property known as *gflow* [6]. This is an efficiently verifiable combinatorial condition on the entangled resource.

When we represent a one-way computation with gflow as a ZX-diagram, the gflow ensures that certain "local" parts of the diagram correspond to individual unitary gates, in a way which can be iteratively translated into an actual unitary circuit. In this case we can hence *extract* a unitary quantum circuit from the ZX-diagram that represents the one-way computation. See for instance [22, 4, 41] for several variations on this idea.

Measurement-based quantum computation like the one-way model is a type of non-unitary quantum computation. Another type of non-unitary model is given by doing *lattice surgery* in the surface code [30, 20]. A lattice surgery procedure can also be represented as a ZX-diagram [20]. Just as in the one-way model, there is a flow condition that ensures such a calculation is deterministic, and that the resulting ZX-diagram can be step-by-step rewritten into a unitary circuit [19].

We see that there are several quantum computational models that can be written in terms of ZX-diagrams, which can be rewritten into a unitary quantum circuit efficiently when they satisfy some condition. The type of flow condition required for these procedures ensures that the diagram can't get "too wild" in the middle, so that we can stepwise rewrite the diagram into something that looks more like a circuit. A natural question to ask then is how much we can weaken such additional conditions, and in particular if we can transform a ZX-diagram into a circuit efficiently in the most general setting, where the only condition we require of the ZX-diagram is that it is proportional to a unitary. The main result of this paper is that such a general efficient procedure most likely does not exist.

2.4 Background on computational complexity

Finally, we provide some background on computational complexity. We assume knowledge of **P**, the boolean satisfiability problem **SAT**, oracle machines, **NP** and nondeterministic Turing machines (NTMs) in general. Our results concern *Cook reductions* (in fact, usually Cook[1] reductions). A Cook reduction from a problem **X** to another problem **Z** is an algorithm for solving **X** using a deterministic Turing machine which halts in polynomial time, but which may query an oracle (in the case of a Cook[1] reduction, exactly once) for **Z**. This implies that, modulo some polynomial-time computation, the problem **Z** is at least as hard as **X**; and that if $\mathbf{Z} \in \mathbf{P}$, we also have $\mathbf{X} \in \mathbf{P}$. In symbols we may write $\mathbf{X} \in \mathbf{P}^{\mathbf{Z}}$. Our results will generally concern problems **Z** related to ZX-diagrams and problems **X** which are at least **NP**-hard (*i.e.*, they suffice to solve **SAT**).

Quantum circuits form a model of computation, which may be considered to generate random outcomes through measurement operations. Note that, just as with the study of boolean circuits as a model of computation, one often considers a quantum circuit to be described by some polynomial-time computable procedure (a sort of "effective blueprint"), which for a given $n \geq 0$ requires time $\mathrm{poly}(\log n)$ to produce a circuit taking inputs of size n. While this constraint is not essential when considering a single circuit on its own (the description of the circuit itself is a finite specification), this constraint prevents us from considering what might otherwise seem like "quantum algorithms" for uncomputable problems (in the same way that one must for boolean circuits). Additionally, to prevent unbounded computational power from being hidden elsewhere in the description of a quantum circuit, one often imposes constraints on the gates and measurements allowed in a circuit. For instance, requiring that gates only act on a small constant number of qubits, and that for parametrised gates like Z_α the parameter α is efficiently computable. The class **BQP** consists of decision problems which can be decided with bounded error (with error probability less than, say, $\frac{1}{3}$) by quantum circuit families satisfying these reasonable constraints. This class represents the decision problems that can be practically solved by an (idealised) quantum computer.

It is not expected that either of **NP** or **BQP** contain the other. So if we can reduce in polynomial time (by many-to-one or oracle reductions) an **NP**-complete problem to some problem **X**, then we expect **X** to be intractable for quantum computers. Certain modifications of the quantum computational model do allow for more difficult problems to be solved, however. For instance, **PostBQP** is the class of problems which may be solved with bounded error by a uniform quantum circuit family, *conditioned on some other measurement* yielding a specific outcome (which occurs with non-zero probability). This "conditioning" restriction is known as *postselection*, and appears to be operationally very powerful, as **PostBQP** coincides with the class **PP**, of decision problems for which a "yes" instance is accepted on more than half of the branches of some NTM halting in polynomial time.

The class **P** is closed under oracles: a deterministic Turing machine equipped with an oracle for some problem in **P** cannot decide more problems in polynomial time than a normal Turing machine, so that $\mathbf{P}^{\mathbf{P}} = \mathbf{P}$. The same is true for **BQP**: any decision problem solvable (with bounded error) by a uniform family of quantum circuits, can also be solved (with bounded error) by some other family of quantum circuits without oracle access, so that $\mathbf{BQP}^{\mathbf{BQP}} = \mathbf{BQP}$. The same is not true, however, for **NP**: it is not known whether $\mathbf{NP}^{\mathbf{NP}}$ (the class of decision problems, for which there is an NTM with an oracle for a problem in **NP**, halts in polynomial time and accepts in some branch precisely for "yes" instances) is equal to **NP**. It is widely conjectured that $\Sigma_2^{\mathrm{p}} := \mathbf{NP}^{\mathbf{NP}} \neq \mathbf{NP}$, and indeed that $\Sigma_3^{\mathrm{p}} := \mathbf{NP}^{\Sigma_2^{\mathrm{p}}} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \neq \mathbf{NP}^{\mathbf{NP}}$, and so forth. The union of $\Sigma_n^{\mathrm{p}} := \mathbf{NP}^{\Sigma_{n-1}^{\mathrm{p}}}$ for all $n > 1$, defines the class **PH**, called the *polynomial hierarchy* [42].

The hardness results which we are most concerned with involve problems in **#P**: the class of problems which may be reduced to counting the number of accepting branches of some NTM on a given input. In particular, we are interested in the problem **#SAT**, of counting the number of "solutions" $x \in \{0,1\}^n$ to an instance of **SAT**, presented as a formula for a function $f : \{0,1\}^n \to \{0,1\}$, where a "solution" satisfies $f(x) = 1$. The problem **#SAT** is **#P**-complete [46], as is tensor contraction over the natural numbers [15], and "strong simulation" (*i.e.*, precise estimation of explicit measurement probabilities) of uniform quantum circuit families [48]. The **#P**-completeness here means that a Cook reduction from any of these problems to some problem **Z**, establishes that there is a Cook reduction from *any* problem $\mathbf{X} \in$ **#P** to **Z**. in this case we say then that **Z** is "**#P**-hard". The computational power of **#P** is considered to be significantly greater than that of **NP**. In particular, Toda [43] showed that $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{\#P}}$.

3 Proof of hardness of Circuit extraction

We now present the central problem of our work.

CircuitExtraction
Input: A ZX-diagram D with n inputs and outputs and at most k wires and/or spiders, and a set $\mathcal{G}$ of unitary gates (each acting on at most $O(1)$ qubits).
Promise: The operator denoted by D is proportional to a unitary.
Output: Either **(a)** a poly(n,k)-size circuit C, expressed as a sequence of gates from $\mathcal{G}$ and expressing an n-qubit unitary that is proportional to the operator denoted by D, if such a circuit exists; or **(b)** a message that no such circuit exists, if that is the case.

Note that here we make no assumptions on the specific gate set $\mathcal{G}$, apart from the computability of the coefficients as described in Section 2.1, and that the number of qubits which is bounded by some constant. One might object to the requirement that the output list of gates must be polynomially related to the size of the input ZX-diagram: however, as we are interested in whether the extraction problem can be solved efficiently, the restriction on the size of C follows from the time required to represent it as a list of gates.

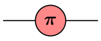
▶ Remark 3.1. For a finite gate set we can consider the gate set $\mathcal{G}$ as being supplied as concrete matrices, while for an infinite set we could consider it as being supplied as an efficient procedure for translating a "gate label" into a matrix. For concreteness sake we can take $\mathcal{G} = \mathcal{B}$, the gate set (1). This gate set contains the parametrised gate Z_α. Formally a circuit will then specify these phases α via some efficiently computable procedure.

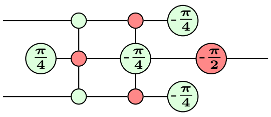
The above problem can of course also be stated for any related graphical language for quantum operations, such as the ZH-calculus [3] or the ZW-calculus [28]. Since such diagrams can be efficiently translated into one another, these problems are of equivalent hardness. There are some other reasonable variations we can consider of **CircuitExtraction** that we will discuss in the next section.

We will now show that **CircuitExtraction** is **#P**-hard. We do this by building a diagram that is proportional to a unitary based on a **SAT** instance, and showing that the resulting matrix the diagram represents is uniquely determined by the number of solutions of the instance.

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean formula with poly(n) terms. We say a bit string $x \in \{0,1\}^n$ is a *solution* to f when $f(x) = 1$. The first step will be to build a ZX-diagram that implements the linear map L_f that takes n qubits to 1 qubit by $L_f|x\rangle = |f(x)\rangle$. We can of course represent f as a tree of AND and NOT operations so that to construct L_f it suffices to find linear maps that implement AND and NOT on $|x\rangle$.

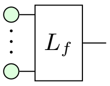
We may consider ZX diagrams for “quantum” versions of the boolean logical AND gate and NOT gate, *i.e.*, linear operators such that NOT$|0\rangle = |1\rangle$, NOT$|1\rangle = |0\rangle$, and AND$|x, y\rangle \mapsto |x \cdot y\rangle$. We could just appeal to the universality of ZX-diagrams to establish that there are diagrams that represent these operations, but for completeness sake let us give some concrete diagrams to realise these operations (up to a constant factor):

NOT $=$ 

AND $\propto$ 

The NOT gate is just an X_π gate, but the AND is more complicated. It is based on the representation of the CCZ gate from [35] that uses 4 T gates. Its correctness can be verified by inputting $|0\rangle$ and $|1\rangle$ on the inputs and seeing that it has the correct action on all these inputs.

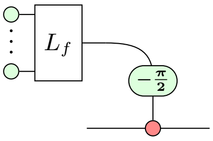
By combining these diagrammatic gadgets for NOT and AND we can build the operation L_f as a ZX-diagram using poly(n) spiders. Now, note that:



$$= \sum_x L_x|x\rangle = \sum_x |f(x)\rangle = \frac{N_0}{2^n}|0\rangle + \frac{N_1}{2^n}|1\rangle =: a_0|0\rangle + a_1|1\rangle \qquad (10)$$

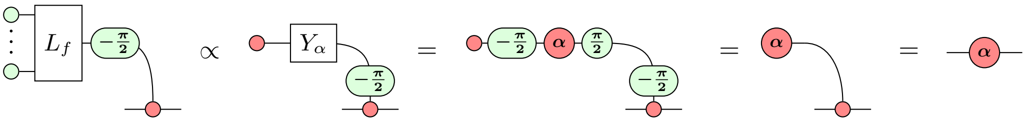
where N_1 is the number of solutions of f, $N_0 = 2^n - N_1$ is the number of “non-solutions” of f, and we set $a_0 = N_0/N$ and $a_1 = N_1/N$ for $N := 2^n = N_0 + N_1$. The resulting state is not normalised: to normalise it we should multiply both sides by $(a_0^2 + a_1^2)^{-1/2}$.

We use the “state” described in Eq. (10) as the input of a controlled operation. By choosing the controlled operation appropriately, we will be left with something proportional to a unitary. We may for instance consider the following diagram:

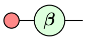
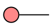
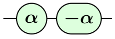
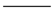


(11)

To see this is unitary first recall that a Y rotation over an angle α applied to $|0\rangle$ gives $Y_\alpha|0\rangle = \cos(\frac{\alpha}{2})|0\rangle + \sin(\frac{\alpha}{2})|1\rangle$. Hence the state of Eq. (10), when properly normalised, can be written as $Y_\alpha|0\rangle$ for $\alpha = 2\sin^{-1}\left(\frac{a_1}{\sqrt{a_0^2+a_1^2}}\right)$. We can then calculate:



(12)

In the above, we use the relation $Y_\alpha = Z_{-\frac{\pi}{2}} X_\alpha Z_{\frac{\pi}{2}}$, and some simple ZX-calculus rewrites (namely that  $=$  and  $=$ ). Hence, the diagram of Eq. (11) is proportional to an X_α rotation where α is uniquely determined by the number of solutions to f. Note that this operation can be easily represented (with at most three gates) using a gate-set such as $\{H, Z_\alpha, \text{CNOT}\}$, in which the set of values of allowed angles α include those that may arise in the diagram of Eq. (12) for some number of solutions N_1 to the formula f; such an operation will be representable using other gate-sets as well.[1]

[1] Note that the gate-set described here cannot be a single, finite gate set for all values of n. However, the angles α arising out of instances of satisfiability in this way can be specified in $O(n)$ bits, precisely by

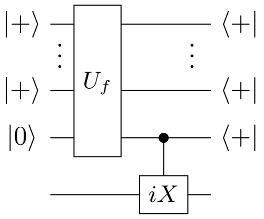
▶ **Theorem 3.2.** **CircuitExtraction** *is* **#P**-*hard.*

Proof. **#SAT** is a **#P**-complete problem, so it suffices to show that we can count the number of solutions to a Boolean formula using a call to an oracle which solves **CircuitExtraction**. Given a Boolean formula $f : \{0,1\}^n \to \{0,1\}$ with $\mathrm{poly}(n)$ terms, construct the diagram of Eq. (11). The diagram here for L_f uses $\mathrm{poly}(n)$ of the diagrammatic gadgets for NOT and AND, and hence the complete diagram consists of $\mathrm{poly}(n)$ spiders, each of which may be restricted to having at most 3 wires. We may apply the **CircuitExtraction** oracle on this diagram subject to a suitable gate set that can exactly generate the possible X-rotations X_α which may arise. As C is a single-qubit circuit with at most $\mathrm{poly}(n)$ gates, we can calculate the unitary it implements, up to any required precision $2^{-O(\mathrm{poly}(n))}$, in polynomial time. We know that the operation realised is of the form X_α, so to determine the value of N_1, it suffices to estimate the entries of the resulting X_α to within an error of $\frac{1}{2\sqrt{2}}$. Determining the value of $a_1/\sqrt{a_0^2 + a_1^2}$ to $2n$ bits of precision is sufficient to do this. ◀

▶ **Corollary 3.3.** *If there is a polynomial time algorithm for* **CircuitExtraction**, *then* $\mathbf{P} = \mathbf{P}^{\#\mathbf{P}}$. *In particular, the polynomial hierarchy collapses to the first level:* $\mathbf{P} = \mathbf{NP} = \mathbf{PH}$.

Proof. If **CircuitExtraction** can be done in polynomial time, then the above shows that we can solve **#SAT** in polynomial time, and hence $\mathbf{NP} \subseteq \mathbf{P}^{\#\mathbf{P}} = \mathbf{P}$. ◀

▶ Remark 3.4. Our construction of the diagram we use to prove our result might seem somewhat arbitrary. To motivate it some more, first realise that instead of the function L_f, we could have used the standard unitary quantum oracle for a Boolean function U_f which acts on $n+1$ qubits via $U_f|x,b\rangle = |x, b \oplus f(x)\rangle$. We can get L_f out of U_f by post-selecting the top n qubits to $\langle +|$. Using the language of post-selection, we may then present a circuit version of Eq. (11):



(13)

The top part is calculating the number of solutions, while the bottom part ensures that this information is fed into a qubit in such a way that the overall operation is proportional to a unitary. The choice of iX is for the sake of simplicity: any unitary U that satisfies $U = -U^\dagger$ would also suffice, such as iY or iZ.

▶ Remark 3.5. Even though we can view the diagram as a post-selected circuit, this does not seem to be where the power of the procedure comes from, as it is for instance in Aaronson's characterisation **PostBQP** = **PP** [1]. In our setting the probability of observing the "correct" outcome is bounded from below by a constant, and does not depend on n. This means in particular that by doing repeat-until-success we could with high probability implement the circuit Eq. (13) on a quantum computer. However, this does not allow you to solve **#SAT**, as adjacent possibilities of the rotation angle α are exponentially close. So rather, the power of the procedure comes from getting an explicit description of the circuit which allows us to exactly calculate the rotation angle.

characterising them in the way we have, by relating some integer ranging in $\{0, 1, \ldots, 2^n\}$ via inverse trigonometric functions. For remarks on what can be achieved with finite gate-sets, the reader may be interested in the related problem **ApproxCircuitExtraction**, in Section 4.

4 Variations on extraction

There are several variations on circuit extraction which we can consider, all of which also turn out to be hard.

The essential trick we used in our proof is that our resulting circuit has just *one* qubit, and hence a description of a unitary on it can easily be transformed into the actual unitary it implements by just multiplying all the resulting matrices. But of course the same statement remains true if we have slightly more than one qubit, say a logarithmic amount in the size of the **SAT** instance. We also see that it then doesn't matter if our circuit contains auxiliary qubits, measurements, or classically-controlled corrections. All of these can be efficiently calculated as long as the number of qubits is small enough. Therefore, let's define the following variant of circuit extraction.

AuxCircuitExtraction
Input: A ZX-diagram D with n inputs and outputs and at most k wires and/or spiders, and a set $\mathcal{G}$ of unitary gates (each acting on at most $O(1)$ qubits).
Promise: The operator denoted by D is proportional to a unitary.
Output: Either **(a)** a deterministic n-qubit circuit implementing the unitary of the input ZX-diagram, described as a $\mathrm{poly}(n,k)$ length list of gates, auxiliary qubit preparations, measurements, and classical corrections, with at most $O(\log k)$ auxiliary qubits; or **(b)** a message that no such circuit exists, if that is the case.

▶ **Theorem 4.1.** **AuxCircuitExtraction** *is* **#P***-hard.*

Proof. We construct the same diagram as in the proof of Theorem 3.2 to solve a **#SAT** instance, except that we can no longer assume that the final circuit will act only on a single qubit: instead it may act on up to $O(\log k)$ qubits, including the operations on the auxiliary qubits. The size of the matrices involved when trying to calculate the resulting unitary is $O(2^{\log \mathrm{poly}(k)}) = O(\mathrm{poly}(k))$, where here k is the size of the input diagram. We may then still multiply the matrices together in polynomial time to obtain sufficiently precise estimates of the coefficients. ◀

One might also object that requiring the output unitary to *exactly* represent the ZX-diagram is too strong – in particular, impossible in general even with an approximately universal, finite gate set – and wish for an approximate output instead. We say that a unitary operator $\tilde{U}$ is an *ε-approximation* of another unitary U for some $\varepsilon > 0$, if $\|\tilde{U} - e^{i\alpha}U\| < \varepsilon$ for some global phase α. Here, $\|M\|$ denotes the operator norm of M: the largest singular value of M.

ApproxCircuitExtraction
Input: A ZX-diagram D with n inputs and outputs and at most k wires and/or spiders, a set $\mathcal{G}$ of unitary gates (each acting on at most $O(1)$ qubits), and a precision parameter $\varepsilon > 0$.
Promise: The operator denoted by D is proportional to a unitary.
Output: Either **(a)** a $\mathrm{poly}(n,k,\log(1/\varepsilon))$-size circuit C, expressed as a sequence of gates from $\mathcal{G}$ and expressing an n-qubit unitary $\tilde{U}$ which is an ε-approximation to either the operator denoted by D, or some operator proportional to it; or **(b)** a message that no such circuit exists, if that is the case.

▶ **Theorem 4.2.** **ApproxCircuitExtraction** *is* **#P***-hard.*

Proof. For a given **SAT** instance $f : \{0,1\}^n \to \{0,1\}$ we again construct the same diagram as in the proof of Theorem 3.2 which denotes a unitary X_α , where α allows us to determine the number of solutions to f. This diagram has $\mathrm{poly}(n)$ spiders. Set $\varepsilon = 2^{-cn}$ for some large enough constant c. Then applying **ApproxCircuitExtraction** gives rise to a circuit, which has $\mathrm{poly}(\mathrm{poly}(n), \log(1/2^{-cn})) = \mathrm{poly}(n)$ gates. We can hence just multiply out the matrices in order to determine the unitary U it implements. This unitary U approximates X_α to degree 2^{-cn}. Since the top left entry of X_α is real, we can first multiply U by the appropriate global phase to ensure it is also real. If we have picked c large enough then the entries of U are then within $\frac{1}{2}2^{-n}$ of that of X_α so that we can determine α by rounding to the nearest allowed value. ◀

Note that, even for exponentially small angles α as might arise when f has few solutions, circuits of polynomial size do *exist* for X_α when $\mathcal{G}$ is an approximately universal gate-set: using the Solovay–Kitaev algorithm [36, 16] or any of its many refinements (see *e.g.* Ref. [5] and references therein), we may synthesise circuits approximating X_α to any precision ε in time scaling polynomially in $\log(1/\varepsilon)$. The difficulty of **ApproxCircuitExtraction** stems from determining *which* angle α to approximate. One might nevertheless want to consider a variation on **ApproxCircuitExtraction** with a polynomial dependence on $1/\varepsilon$ instead of $\log 1/\varepsilon$. We believe this variant will still be hard: see Remark 4.4.

Let us consider one final variation on extraction. One could argue that the reason that we end up with a hard problem in these instances, is because requiring the output to be some kind of circuit is too restrictive. The ultimate goal of circuit extraction is that we wish for the ZX-diagram to be run on a quantum computer in order to obtain some probability distribution over outcomes; but the complexity of **CircuitExtraction** and its variations seems to arise from the complexity of finding a precise description of the procedure to do so. Cutting out the middle-man, we may consider *any* process which takes as input a unitary ZX-diagram, and produces bit strings as output whose distribution conforms with the one we expect from the unitary.

UnitaryZXSampling
Input: A ZX-diagram D with n inputs and outputs and at most k wires and/or spiders.
Promise: The operator denoted by D is proportional to some unitary U.
Output: A sample $x \in \{0,1\}^n$ from a probability distribution, given by (or sufficiently close to) $|\langle x|U|0\cdots 0\rangle|^2$.

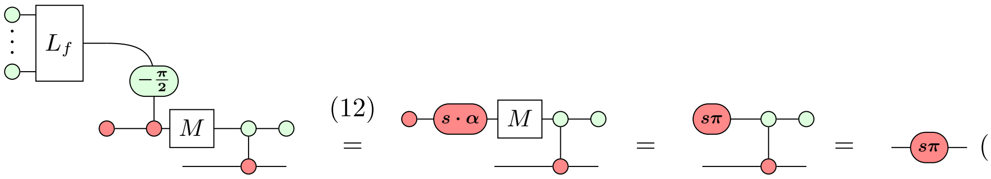
It is clear that **UnitaryZXSampling** is at least as hard as **BQP**: we could just input a ZX-diagram that directly represents a quantum circuit, in which case this problem is equivalent to simulating that circuit. The reason we write here that the probabilities just have to be "sufficiently close" is because the exact number doesn't matter for the following theorem. (For instance: we could allow the probability to additively deviate by 1/3 from the true value.)

▶ **Theorem 4.3.** *There is a randomised polynomial reduction from* **NP** *to* **UnitaryZXSampling**. *In other words: with access to a* **PromiseUnitaryZXSampling** *oracle – which produces the expected output if the input diagram is unitary and arbitrary output otherwise – we can with high probability solve* **NP***-complete problems.*

Proof. **SAT** is an **NP**-complete problem. To randomly reduce **NP** it however suffices to consider the problem **USAT** by the Valiant–Vazirani theorem [45]. **USAT** asks us to determine whether a Boolean formula is satisfiable, given the promise that it has at most one solution. Using the randomised reduction from **SAT** to **USAT**, we consider how to solve **USAT** using a **PromiseUnitaryZXSampling** oracle.

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean formula that has at most one solution. Construct the diagram Eq. (11) as in the previous proofs: as a unitary this implements the identity iff f is not satisfiable, and X_α for some fixed angle $\alpha > 0$ when f is satisfiable. In the latter case, the value of α is exponentially small, but known precisely, as f has exactly one solution in this case. So we can say the circuit implements $X_{s\cdot\alpha}$ where $s \in \{0,1\}$ encodes whether f is satisfiable or not.

Let M be the one-qubit (non-unitary) matrix that maps $|0\rangle \mapsto |0\rangle$ and $X_\alpha|0\rangle \mapsto |1\rangle$, so that in particular $MX_{s\cdot\alpha}|0\rangle = |s\rangle$. By universality of ZX-diagrams we can find some (constant sized) diagram to represent M. We can then calculate:



(14)

Here the last step is just spider fusion (9). We see then that the ZX-diagram on the left in Eq. (14) implements either the identity, or an X_π operation (that is to say, a NOT operation), depending on whether f is satisfiable. When we feed this ZX-diagram to an oracle for **PromiseUnitaryZXSampling**, we get either the output `0` or `1`, where a `0` indicates with high probability that the circuit is the identity, and a `1` indicates that the circuit is a NOT operation. We can repeatedly call the oracle to get additional samples to increase our confidence in the result.

Now suppose f is a general instance of **SAT**, which may have more than one solution. Using the Valiant–Vazirani reduction multiple times we probabilistically produce different Boolean formulae $f_1, \ldots, f_m$. If f is not satisfiable, then none of the f_j will be satisfiable either and this is what the **PromiseUnitaryZXSampling** will tell us as well. If f *is* satisfiable, then a significant fraction of the f_j will have a unique solution, so that our oracle tells us they are satisfiable. For the other f_j the oracle will return some arbitrary output. So by picking m large enough there will with high probability be some f_j that will be uniquely satisfiable, and so we can conclude that f is satisfiable as well.

Hence, we can determine with arbitrary high probability whether a **SAT** instance is satisfiable using enough calls to **PromiseUnitaryZXSampling**. ◀

▶ Remark 4.4. For **ApproxCircuitExtraction** we allowed a polynomial dependence on $\log 1/\varepsilon$ for the circuit size. We believe this is reasonable as the Solovay–Kitaev algorithm allows you to find a poly$(\log 1/\varepsilon)$-sized circuit when approximating a unitary. However, one can also consider the hardness of the problem when we allow the circuit size to depend on poly$(1/\varepsilon)$, or even when the circuit size does not depend at all on the error. One might suspect that this could change the hardness of the problem. (As an analogy: one may compute the permanent of an $n \times n$ matrix with positive entries to within a multiplicative error of ε in time poly$(n, 1/\varepsilon)$ [32], despite the exact problem being **#P**-complete.) However, note that if we can approximately do circuit extraction up to some *constant* error (say $\varepsilon = 1/10$), we can feed the resulting circuit to a quantum computer in order to solve **UnitaryZXSampling**. Hence, even if we were to relax **ApproxCircuitExtraction** to allow an output circuit of size poly$(1/\varepsilon)$, an efficient algorithm for **ApproxCircuitExtraction** would imply that **NP** $\subseteq$ **BQP** by Theorem 4.3.

▶ Remark 4.5. If we knew that the number of solutions to the **SAT** instance was some other fixed number, then we could pick a different matrix M' to boost the state up to X_π gate as in the proof of Theorem 4.3. If we pick M' "slightly wrong", then the resulting diagram will just be close to X_π. One might think that we could use such a procedure to try and determine the number of solutions to f by doing binary search on the number of solutions, and so boost the power of **UnitaryZXSampling** to **#P**. However, the problem with this is that the resulting diagrams are not proportional to a unitary most of the time. There might be some way around this issue, so that **UnitaryZXSampling** is still **#P**-hard: we leave this as an open problem.

▶ Remark 4.6. Note that if we were to consider a version of **UnitaryZXSampling**, without the promise of unitarity, such an oracle would be as powerful as **PostBQP**, since we can represent any ZX-diagram as a post-selected quantum circuit (and conversely). In our case, the power again comes not so much from postselection, as being able to take advantage of the versatility of ZX-diagrams to gain access, in some way, to extract very precise information regarding a **#P** problem.

5 Upper bounding the complexity of CircuitExtraction

Given that **CircuitExtraction** is **#P**-hard, one might ask whether or not the problem is **#P**-complete (or more precisely: $\mathbf{FP}^{\mathbf{\#P}}$-complete since we are not making a decision but rather outputting a circuit), in the sense that a Turing machine with access to a **#P** oracle would be able to solve it, for some given polynomial upper bound on circuit size and some given gate-set (perhaps with suitable restrictions), in polynomial time. We have not managed to prove such a completeness result. We will however present the following upper bounds on decision problem versions of circuit extraction, relying on techniques from [2] that relate calculating amplitudes of quantum circuits to counting complexity problems.

First, consider the following decision problem: given a ZX-diagram, and a circuit, determine whether the circuit implements a unitary which is proportional to that represented by the ZX-diagram (whether by a factor of $e^{i\theta}$ for some angle θ, or a more general complex number). This problem is in $\mathbf{coNP}^{\mathbf{\#P}}$. To sketch why this is, consider a circuit C representing a unitary U, and a ZX diagram D representing an operator V. If a_0 and a_1 are two non-zero coefficients from U, and b_0 and b_1 are the corresponding (non-zero) pair of coefficients from the matrix V represented by D, then $U \propto V$ only if $a_0/b_0 = a_1/b_1$ for all possible such pairs. We also require that for any coefficient a in U which is zero, the corresponding coefficient b of V is also zero. Taken together, this implies that for all corresponding pairs of coefficients of U and V we should have $a_0 b_1 = a_1 b_0$. This is also sufficient for $U \propto V$ to hold. Now, a **#P** oracle allows one to calculate coefficients[2] of ZX-diagrams and circuits. Hence, if $U \not\propto V$, an NTM with access to a **#P** oracle can non-deterministically find a witness that these two operators are not in fact proportional to one another. Thus, determining whether a circuit does *not* represent a unitary which is denoted (up to scalar factors) by a ZX-diagram, is in $\mathbf{NP}^{\mathbf{\#P}}$.

The above result has a simple corollary: determining whether a ZX-diagram is proportional to a unitary itself belongs to $\mathbf{coNP}^{\mathbf{\#P}}$. We may see this by the fact that a ZX-diagram denoting an operator V, which is proportional to a unitary, satisfies $VV^\dagger \propto I$. We may

[2] In this case, it is not necessary to compute complete information about a_0, a_1, b_0, and b_1: it suffices to compute information about individual components of the products $a_0 b_1$ and $a_1 b_0$ when considering these as numbers in some number field over $\mathbb{Q}$. See Ref. [2] for the details.

represent $VV^\dagger$ by composing the diagram D with its adjoint (which is the left-to-right mirror image of D, with all phase angles negated). This composite diagram may easily be computed, at which point we may ask whether the operator it represents is proportional to the identity by a non-zero scalar factor. As we note above, this problem is in $\mathbf{coNP}^{\mathbf{\#P}}$.

Finally, using these ideas, we may consider the decision problem of determining for a ZX-diagram D denoting an operator V and some gate set $\mathcal{G}$ and polynomial length bound N, whether *there exists* a circuit of at most N gates over $\mathcal{G}$ which implements V (up to a scalar). This problem is in $\mathbf{NP}^{\mathbf{NP}^{\mathbf{\#P}}}$: for an NTM with access to an $\mathbf{NP}^{\mathbf{\#P}}$ oracle, it suffices to make a nondeterministic guess at a circuit of length N (where each gate may be the identity operator, or some gate $G \in \mathcal{G}$ acting on a non-deterministically chosen set of qubits) and then query the oracle to determine whether the circuit realises V. A deterministic Turing machine, with access to an oracle for this problem, could then solve **CircuitExtraction** in polynomial time using standard techniques, using the oracle to facilitate a search for a circuit to realise D.

These observations represent the most straightforward approach to determining an upper bound for the circuit extraction problem, and seem to place it at a level of complexity significantly higher than $\mathbf{P}^{\mathbf{\#P}}$. If we conceive of **#P** as broadly representing the complexity of evaluating a tensor network, a superficial analogy between **CircuitExtraction** and boolean circuit minimisation [25, 7] would seem to suggest that **CircuitExtraction** is likely to be hard for some complexity class higher than $\mathbf{P}^{\mathbf{\#P}}$ (barring some collapse of complexity classes).

6 Conclusion

In this paper we studied the problem of extracting a quantum circuit description from a unitary ZX-diagram. We've shown that this problem is **#P**-hard by reducing **#SAT** to an application of circuit extraction. We've also studied some variations where we allow auxiliary qubits, classical control, and/or approximate synthesis of the desired unitary, and have shown that these problems are also **#P**-hard. In addition, we studied the hardness of a machine that takes in a unitary ZX-diagram and outputs measurement samples from that ZX-diagram, and have shown that such a machine allows one to probabilistically solve **NP**-hard problems.

A conclusion to be drawn from our results is that if we want some efficient procedure to transform a unitary ZX-diagram into a quantum circuit, then we will have to have some additional information about the structure of the ZX-diagram. In the known procedures for efficient circuit extraction [4, 41, 19], this additional information takes the form of a kind of "flow" on the diagram that prevents parts of the diagram from becoming too unwieldy. An immediate question then is if there are other types of, more general, promises on the structure of the diagram which then allow you to extract a circuit from it.

Aaronson showed that sampling from a post-selected quantum circuit is hard [1]. Our results imply that some other tasks surrounding *unitary* post-selected circuits (that is, circuits which perform a unitary transformation conditioned on some post-selection) are hard. However, this hardness seems to stem not from the post-selection itself, as the post-selections can be simulated with high probability in our case. Rather, the hardness seems to stem from a hypothetical ability to find an equivalent, deterministic way to realise the same operation – which implies an ability to extract difficult-to-access information about the input diagram.

A question related to circuit extraction from ZX-diagrams is circuit extraction from deterministic measurement patterns (in for instance the one-way model or lattice surgery). When we have a deterministic measurement pattern, we can represent each branch of the computation by a ZX-diagram denoting a unitary. Our hardness proof does however not immediately translate to this setting, as it might be that the fact that all of these ZX-diagrams are branches of the same measurement pattern forces some kind of structure on the diagrams that might make it easier to rewrite them into circuits. The diagrams we used to show hardness of circuit extraction are as far as we are aware not representable as branches of some deterministic measurement pattern, so that we can't use the same proof. We leave it for future work to determine the hardness of extracting unitary circuits from deterministic measurement patterns.

References

1 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005. doi:10.1098/rspa.2005.1546.

2 Leornard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26:1524–1540, 1997. doi:10.1137/S0097539795293639.

3 Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 18–34. Open Publishing Association, 2019. doi:10.4204/EPTCS.287.2.

4 Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, March 2021. doi:10.22331/q-2021-03-25-421.

5 Adam Bouland and Tudor Giurgica-Tiron. Efficient universal quantum compilation: An inverse-free Solovay–Kitaev algorithm. *arXiv preprint*, 2021. arXiv:2112.02040.

6 Daniel E. Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9(8):250, 2007. doi:10.1088/1367-2630/9/8/250.

7 David Buchfuhrer and Christopher Umans. The complexity of Boolean formula minimization. *Journal of Computer and System Sciences*, 77(1):142–153, 2011. Celebrating Karp's Kyoto Prize. doi:10.1016/j.jcss.2010.06.011.

8 Titouan Carette. *Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations.* PhD thesis, Loria, Université de Lorraine, 2021. URL: https://hal.archives-ouvertes.fr/tel-03468027/document.

9 Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of Graphical Languages for Mixed States Quantum Mechanics. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 108:1–108:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.108.

10 Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. Foundations for Near-Term Quantum Natural Language Processing. *arXiv preprint*, 2020. arXiv:2012.03755.

11 Bob Coecke and Ross Duncan. Interacting quantum observables. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, 2008. doi:10.1007/978-3-540-70583-3_25.

12 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13:043016, 2011. doi:10.1088/1367-2630/13/4/043016.

13 Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes.* Cambridge University Press, 2017. doi:10.1017/9781316219317.

14 Richard D. P. East, John van de Wetering, Nicholas Chancellor, and Adolfo G. Grushin. AKLT-states as ZX-diagrams: diagrammatic reasoning for quantum states. *PRX Quantum*, 3:010302, January 2022. doi:10.1103/PRXQuantum.3.010302.

15 Carsten Damm, Markus Holzer, and Pierre McKenzie. The complexity of tensor calculus. *Computational Complexity*, 11(1):54–89, 2002. doi:10.1007/s00037-000-0170-4.

16 Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm. *arXiv preprint*, 2005. arXiv:0505030.

17 Niel de Beaudrap. Well-tempered ZX and ZH Calculi. In Benoît Valiron, Shane Mansfield, Pablo Arrighi, and Prakash Panangaden, editors, *Proceedings 17th International Conference on Quantum Physics and Logic, Paris, France, June 2 - 6, 2020*, volume 340 of *Electronic Proceedings in Theoretical Computer Science*, pages 13–45. Open Publishing Association, 2021. doi:10.4204/EPTCS.340.2.

18 Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to Reduce $\pi/4$-Parity-Phase Circuits, Motivated by the ZX Calculus. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 131–149. Open Publishing Association, 2020. doi:10.4204/EPTCS.318.9.

19 Niel de Beaudrap, Ross Duncan, Dominic Horsman, and Simon Perdrix. Pauli Fusion: a Computational Model to Realise Quantum Transformations from ZX Terms. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 85–105. Open Publishing Association, 2020. doi:10.4204/EPTCS.318.6.

20 Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4, 2020. doi:10.22331/q-2020-01-09-218.

21 Niel de Beaudrap, Aleks Kissinger, and Konstantinos Meichanetzidis. Tensor Network Rewriting Strategies for Satisfiability and Counting. In Benoît Valiron, Shane Mansfield, Pablo Arrighi, and Prakash Panangaden, editors, *Proceedings 17th International Conference on Quantum Physics and Logic, Paris, France, June 2 - 6, 2020*, volume 340 of *Electronic Proceedings in Theoretical Computer Science*, pages 46–59. Open Publishing Association, 2021. doi:10.4204/EPTCS.340.3.

22 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020. doi:10.22331/q-2020-06-04-279.

23 Ross Duncan and Simon Perdrix. Rewriting Measurement-Based Quantum Computations with Generalised Flow. In *Proceedings of ICALP*, Lecture Notes in Computer Science, pages 285–296. Springer, 2010. doi:10.1007/978-3-642-14162-1_24.

24 Richard D. P. East, Pierre Martin-Dussaud, and John van de Wetering. Spin-networks in the ZX-calculus. *arXiv preprint*, 2021. arXiv:2111.03114.

25 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

26 Craig Gidney and Austin G. Fowler. Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation. *Quantum*, 3:135, April 2019. doi:10.22331/q-2019-04-30-135.

27 Craig Gidney and Austin G. Fowler. Flexible layout of surface code computations using AutoCCZ states. *arXiv preprint*, 2019. arXiv:1905.08916.

28 Amar Hadzihasanovic. A diagrammatic axiomatisation for qubit entanglement. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 573–584. IEEE, 2015. doi:10.1109/LICS.2015.59.

29 Michael Hanks, Marta P. Estarellas, William J. Munro, and Kae Nemoto. Effective Compression of Quantum Braided Circuits Aided by ZX-Calculus. *Physical Review X*, 10:041030, 2020. doi:10.1103/PhysRevX.10.041030.

30 C. Horsman. Quantum picturalism for topological cluster-state computing. *New Journal of Physics*, 13(9):095011, 2011. doi:10.1088/1367-2630/13/9/095011.

31 C. Horsman, Austin G. Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14:123011, 2012. doi:10.1088/1367-2630/14/12/123011.

32 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries. *J. ACM*, 51(4):671–697, July 2004. doi:10.1145/1008731.1008738.

33 Aleks Kissinger and Arianne Meijer-van de Griend. CNOT circuit extraction for topologically-constrained quantum memories. *Quantum Information and Computation*, 20:581–596, 2020. doi:10.26421/QIC20.7-8.

34 Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Physical Review A*, 102:022406, August 2020. doi:10.1103/PhysRevA.102.022406.

35 Aleks Kissinger and John van de Wetering. Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 2022. doi:10.1088/2058-9565/ac5d20.

36 Alexei Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

37 Kang Feng Ng and Quanlong Wang. A universal completion of the ZX-calculus. Preprint, 2017. arXiv:1706.09877.

38 M. A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. doi:10.1119/1.1463744.

39 Roger Penrose. Applications of negative dimensional tensors. In *Combinatorial Mathematics and its Applications*, pages 221–244. Academic Press, 1971.

40 Robert Raussendorf and Hans J. Briegel. A One-Way Quantum Computer. *Physical Review Letters*, 86:5188–5191, May 2001. doi:10.1103/PhysRevLett.86.5188.

41 Will Simmons. Relating Measurement Patterns to Circuits via Pauli Flow. In Chris Heunen and Miriam Backens, editors, *Proceedings 18th International Conference on Quantum Physics and Logic, Gdansk, Poland, and online, 7-11 June 2021*, volume 343 of *Electronic Proceedings in Theoretical Computer Science*, pages 50–101. Open Publishing Association, 2021. doi:10.4204/EPTCS.343.4.

42 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977. doi:10.1016/0304-3975(76)90061-X.

43 Seinosuke Toda. PP is as hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, pages 865–877, October 1991. doi:10.1137/0220053.

44 Alex Townsend-Teague and Konstantinos Meichanetzidis. Classifying Complexity with the ZX-Calculus: Jones Polynomials and Potts Partition Functions. *arXiv preprint*, 2021. arXiv:2103.06914.

45 L G Valiant and V V Vazirani. NP is as Easy as Detecting Unique Solutions. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 458–463, New York, NY, USA, 1985. Association for Computing Machinery. doi:10.1145/22145.22196.

46 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.

47 John van de Wetering. ZX-calculus for the working quantum computer scientist. *arXiv preprint*, 2020. arXiv:2012.13966.

48 Maarten Van Den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation*, 10(3):258–271, 2010. doi:10.5555/2011350.2011356.

49 Renaud Vilmart. A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, 2019. doi:10.1109/LICS.2019.8785765.