On the Verge to Improve Technique of T-count Reduction via Spider Nest Identities

Witalis Domitrz
Supervisors: Aleks Kissinger, Quanlong Wang
Linacre College
University of Oxford

A dissertation submitted for the degree of
Master of Science in Mathematics and Foundations of Computer Science
Trinity 2021

Acknowledgements

I would like to thank Aleks Kissinger for introducing me to diagrammatic reasoning, ZX-calculus, and quantum computing, Quanlong Wang, who helped me keep the right direction and for the regular supervision, and Xiaoning Bian for help in using the `stomp-code` [9].

Approximate, universal quantum computation is most commonly described by circuits consisting of Clifford+T gates. The T gate, except being crucial for the universality, is the one which is, in real-world implementations, the most resource-intensive, and the least fault-tolerant of all operations. Because of that, a natural question emerges – how to reduce the number of T gates (or T-count) in a given circuit. The known methods used to reduce the T-count to an optimal value have an exponential running time [24, 2], which motivates the search for an efficient heuristic algorithm. The importance of this problem, along with proposed solutions were discussed in multiple publications including [6, 18, 16, 15, 7, 23].

Spider nest identities, introduced in [16], together with the decomposition of the circuit inspired by [18], were combined in [15] to provide an effective algorithm for T-count reduction. This approach was tested and compared with previous results, and, in some cases, it resulted in a significant improvement of the state of the art. Another advantage of this algorithm was, in contrast to the other results, a significant improvement of the execution time. In this dissertation, we discuss several modifications of the algorithm in order to gain partial outperformance without a significant increase of the running time, which include analysis of the influence of the order of applying the identities, efficient usage of identities generated from small and big spider nests, combining the results obtained in [25] with spider nest identities, and others. Some of these modifications improve or achieve the current state of the art for various circuits. We also provide a quantitative comparison of results, verified using [3] – a tool created alongside [4], on numerous benchmark circuits obtained from [9, 26, 3]. Moreover, we present abstract formulations of considered problem, pose a question of completeness of spider nest identities with respect to T-count reduction, and present different, possibly more elegant, way of phrasing the spider nest identities, as dense spider nest identities.

Contents

Chapter 1

Introduction

The goal of realizing universal quantum computation is currently, approached by approximations that use both Clifford and non-Clifford gates. The usage of Clifford gates is a natural consequence of their relative stability and efficiency [10], but unfortunately, as we can efficiently simulate all circuits consisting of only Clifford operations on classical computer [1], to achieve the efficient, approximate universality, there is a need for usage of the non-Clifford gates. The canonical choice of such an operation is the T gate (a phase rotation by $\frac{\pi}{4}$), which is sufficient to realize the approximate universal quantum computation [28]. The drawback that comes with the T gate, is its significantly lower fault-tolerance, or when implemented in a fault-tolerant way, significantly higher resource requirements [10]. Such circumstances make us consider a natural question – how to modify a circuit such that its semantics remains the same, but it uses as few T gates as possible – or in other words, how to reduce the T-count of a given circuit.

In this dissertation, we focus on the question of efficient T-count reduction. This problem was already considered in various publications including [6, 17, 7, 23, 18, 16, 15]. We focus our work on improving the newest of mentioned algorithms [15]. A reoccurring approach in the T-count optimization problem is analyzing the non-Clifford diagonal circuits. In our work, we follow the approach from [18], by using the improved version presented in [15], in order to synthesize a single subcircuit composed of phase gadgets [23] and containing all T gates. Additionally, we utilize the results of [25, 29, 11] in order to present a modified way of transforming Galois Field multipliers circuits, which combined with the considered algorithm yields new state-of-the-art results.

In order to process the diagonal non-Clifford part of the circuits, following [15], we use, introduced in [16], spider nest identities. This particular family of circuits was proven to be highly effective in both [15, 16]. We discuss various approaches to this challenge – by analyzing the problem and the identities in abstract formal terms, as well as by practical modifications of phase gadget elimination tactics (or PHAGE Tactics) [15, 16].

1.1 Content of the dissertation and originality

This dissertation is composed of the two following parts, the content of which we describe below. Additionally, for each chapter, we clearly state its original contributions.

1.1.1 Preliminaries

Chapter 2 covers preliminaries, foundations, and background for the problems considered in the dissertation. All the concepts discussed in this chapter are not novel work. Here we will briefly discuss the content of each of sections 2.1 to 2.5.

ZX-calculus

In section 2.1, we present a short introduction to the ZX-calculus, based on [14]. We cover basic rules and give various proofs of some of the properties in order to familiarize the reader with the notation. The presented there approach is meant to be purely axiomatic, not to intimidate the reader with the complexity of the underlying concepts. It is also not meant to be a full introduction to a ZX-calculus. As such, we recommend reading [14]. All proofs and concepts considered in this section were introduced, used, known, and considered previously.

Phase gadgets

In section 2.2 we introduce gadgets, and more specific phase gadgets, along with their interesting properties. All proofs and concepts considered in this section were introduced, used, known, and considered previously.

Spider nest identities

Section 2.3 is is an introduction to spider nest identities as described in [15, 16]. We start from a description of the spider nest identity on 4 wires and extend it to multiple wires using the method presented in [15]. Finally, we discuss the T-count of spider nest identities and present the composite spider nest identities. All proofs and concepts considered in this section were introduced, used, known, and considered previously.

PHAGE Tactics

In section 2.4, we present PHAGE Tactics as described in [15], as well as more specific PHAGE4 and PHAGE5 tactics. We discuss problems related to PHAGE Tactics and present a small example illustrating the advantage of utilizing the composite spider nest identities in PHAGE Tactics. All proofs and concepts considered in this section, except

the small example for the advantage of composite spider nest identities, were introduced, used, known, and considered previously.

Algorithm

In the final section of the preliminaries– section 2.5, we present a detailed recap of the algorithm presented in [15] including descriptions of all major steps (excluding the PHAGE Tactics). All proofs and concepts considered in this section were introduced, used, known, and considered previously.

1.1.2 Main contributions

In this section, we discuss the main contributions of this dissertation and explicitly state which parts are original, and which were already present in the literature.

1.1.2.1 Rephrasing the phase gadgets T-count reduction

In chapter 3 we give a formal definition of a problem abstracting the goal of PHAGE Tactics, we also consider specific instances of the problem for the spider nest identities. Then we state an open question about the completeness of spider nest identities with respect to T-count reduction and consider an example of a possible way to create a counterexample. Finally, we extend our problem by a transformation resembling the way $CNOT$ gate commutes through the phase gadgets. We did not find any similar approaches to explicitly formalize in that way the problem of phase gadget T-count reduction in the literature. Here we should also note that an other known way of analyzing the problem in an abstract way is highly related to the Reed-Muller codes [7].

1.1.2.2 Dense spider nest identities

In chapter 4 we introduce a new, elegant way of phrasing the spider nest identities by considering dense spider nest identities. We also reformulate the problem from chapter 3 using the newly defined identities. We did not find any similar work in the literature.

1.1.2.3 Modifications of the PHAGE Tactics

In chapter 5 we consider various modifications of the way the PHAGE Tactics utilize the spider nest identities. We present approaches that improved the performance of the algorithm as well as discuss problems with other modifications. We also present a quantitative comparison of achieved results some of which are a new state-of-the-art. As this chapter focuses mostly on the PHAGE Tactics, it highly relies on [15], but the approaches considered there were not previously considered in the literature.

1.1.2.4 Modifications of the decomposition phase

In chapter 6 we utilize the results of [25] in order to modify and improve the way we process the Galois field multipliers family of circuits before it is in a form suitable for applying PHAGE Tactics. We discuss in detail how the preprocessing steps are executed, and we point out problems preventing us from using them for general quantum circuits. We quantitatively compare the results with other approaches. The work presented in this chapter highly relies on results of [15, 25], we present the proofs of (previously known) commutation steps for $CNOT$ and $Toffoli$ gates, which we did not find anywhere else in the literature. By combining both approaches we achieve and present new, state of the art results.

1.2 Tools and software

While working on the dissertation, we used and utilized various tools and software, which we present below.

- For generating our results, we used a modified version of `stomp` – *Haskell implementation of the T-count optimization algorithm "stomp"* [9].
- For validating our results, we used `Feynman` – *Quantum circuit analysis toolkit* [3, 4].
- For creating the diagrams, we used `TikZiT` – *a super simple GUI editor for graphs and string diagrams* [21].
- For automatic reasoning and validating our proves, we used `PyZX` – *Python library for quantum circuit rewriting and optimisation using the ZX-calculus* [22].
- As a template for this dissertation we used *A Thesis Class* provided at `https://www.maths.ox.ac.uk/members/it/faqs/latex/thesis-class`.

Chapter 2

Preliminaries

2.1 ZX-calculus

In this dissertation, we use the ZX-calculus, as introduced in [12] and described in [14]. For the sake of simplicity, we omit global, nonzero coefficients (so we use the equivalence up to a nonzero number as the equality). Here we introduce basic rules of ZX-calculus in order to familiarize the reader with the notation. For a more extensive introduction, we suggest reading and studying [14].

2.1.1 Diagrams of ZX-calculus

In ZX-calculus we consider (possibly empty) diagrams constructed of red and green nodes (also called spiders) each associated with some (possibly zero) phase, and some inputs and outputs. We also add special Hadamard nodes that are defined using the red and green nodes.

2.1.1.1 Phases

Phases, in case of ZX-calculus are representing angles, so they are real numbers up to an equivalence relation $\sim$ given by $2k\pi + \alpha \sim \alpha$ for all $\alpha \in \mathbb{R}$ and $k \in \mathbb{Z}$.

Spiders with zero phase

For simplicity, we draw spiders with zero phases without any phase. It is a substantial simplification as we tend to use zero phase spiders most commonly.

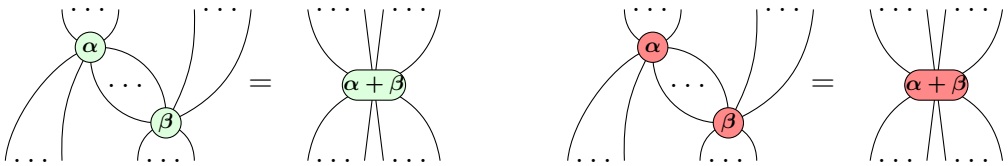
(2.1)

2.1.2 Rules

We present the set of rules as they are presented in [14]. Other, equivalent, sets of rules are known and present in the literature [27, 30, 13].

2.1.2.1 Spider-fusion

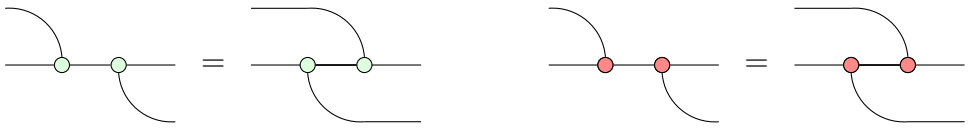
The spider-fusion rules allow us to fuse two connected nodes of the same colour into a single spider. The resulting spider has all the inputs and outputs of these nodes, excluding the ones between one another.



(2.2)

Consequences

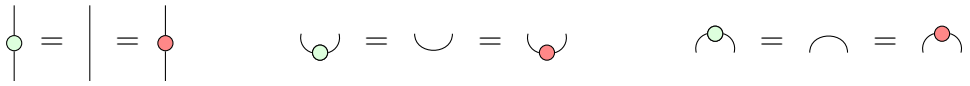
This rule, despite being extremely simple, has some far-fetched consequences. The most important one, for our application, is the fact that it causes phase gadgets (which we will introduce later) to commute, and more precisely (and generally), the following property holds.



(2.3)

2.1.2.2 Identity, cup, and cap

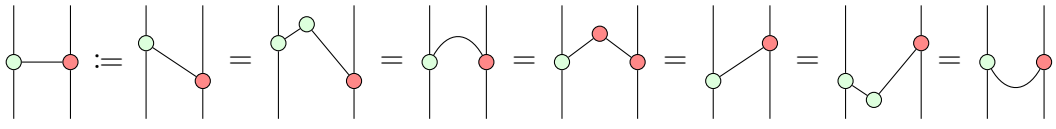
We say that a single node with no phase (phase 0) and exactly one input and one output, two inputs, or two outputs can be removed.



(2.4)

Consequence of cups and caps

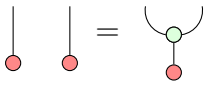
A notable consequence of these rules is the fact that if some edge is not an input nor an output of the whole diagram, it doesn't matter if it is an input or an output of nodes to which it is connected. This allows us to abuse the notation as follows.



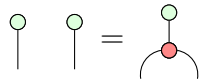
(2.5)

2.1.2.3 Strong complementarity

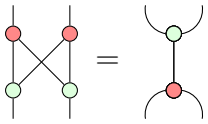
The rules of strong complementarity, or some of their equivalent forms, are a way to describe how the spiders (with no phase, or phase 0) of different colors can commute.



(2.6)



(2.7)



(2.8)

Disconnecting

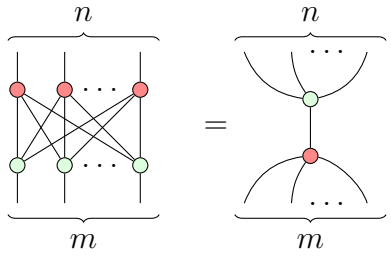
First, two rules allow us to replace two single spiders of the same color with a spider of opposing color with the same edges, additionally connected to a new single node of the original color (eqs. (2.6) and (2.7)).

Commutation

The rule eq. (2.8) introduces a way to commute spiders of different colors.

Equivalent formulation

We can use these simple rules of bounded size to introduce a more general and simpler to use equivalent rule:



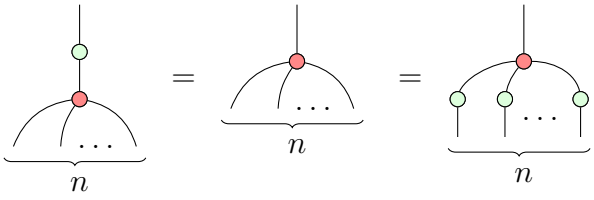
(2.9)

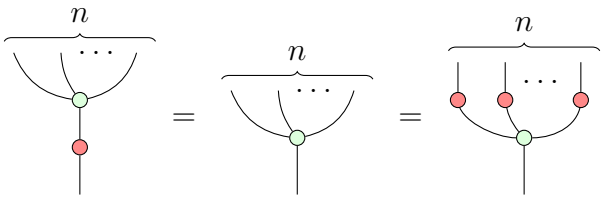
for every non-negative integers n and m.

Proof of the equivalence

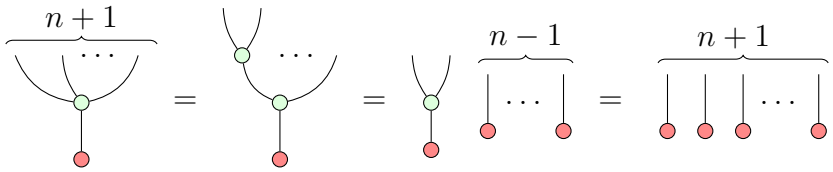
Equations (2.6) and (2.7) are cases of eq. (2.9) for $(n, m) = (2, 0)$ and $(n, m) = (0, 2)$ respectively. Additionally eq. (2.8) is eq. (2.9) for $(n, m) = (2, 2)$.

On the other hand, eq. (2.9) for $n = 1$, or for $m = 1$ are tautologies, by eq. (2.4).

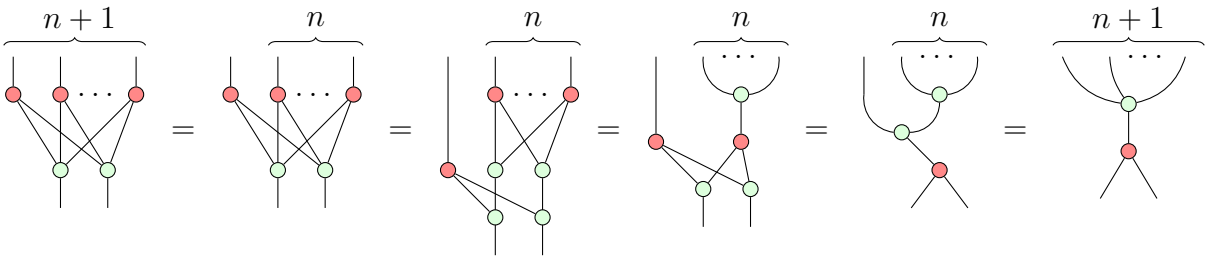




Moreover, by a simple induction and spider-fusion, eqs. (2.6) and (2.7) imply eq. (2.9) for $n = 0$ or $m = 0$.

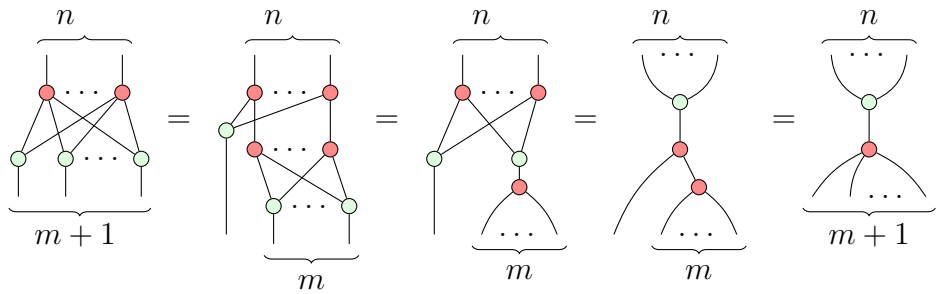


To prove eq. (2.9) for every $n \geqslant 2$ and every $m = 2$ (as in [14]), we first note that the base case, so $n = 2$ is exactly eq. (2.8), and now, if we assume it for some fixed n, then we can prove it for $n + 1$ as following.



(2.10)

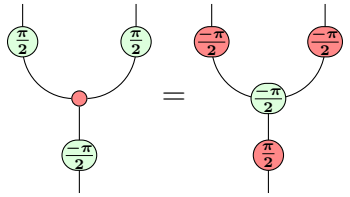
Finally, to get eq. (2.9) in other cases, we fix some m and n and, analogically to eq. (2.10), proceed as follows to get the equation for $m + 1$.



(2.11)

2.1.2.4 Rotations by $\frac{\pi}{2}$

All above the rules (except spider-fusion) were defined for zero phase spiders. In order to extend them to $\frac{\pi}{2}$ rotations, let's introduce the following rule.

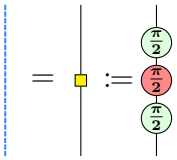


(2.12)

This rule gives us interesting implications, for example,  =  , and is highly related to the Bloch sphere interpretation of quantum states. It is also the last rule needed for completeness of ZX-calculus with phases that are multiples of $\frac{\pi}{2}$ so of Clifford maps.

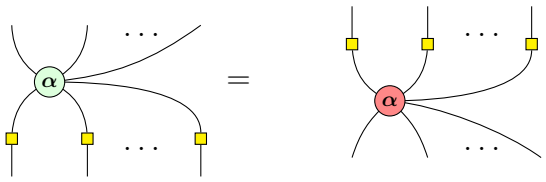
2.1.2.5 Hadamard gate

We will also use a Hadamard gate, sometimes presented as a different kind of wire. It is defined as follows:



(2.13)

which comes with the following rule.



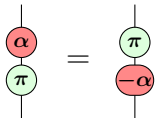
(2.14)

Let's note that this rule works for arbitrary angle α, which is a substantial difference from the previous rule, where all angles were multiples of $\frac{\pi}{2}$.

Remark 1. Using the Hadamard gate we can easily convert all diagrams to diagrams composed of spiders of only one color and Hadamard gates. Such an approach can be observed for example in [14].

2.1.3 Commutation of NOT

The final rule that we will introduce is a commutation rule for one spider with phase π.



(2.15)

2.1.4 Completeness

It is important to mention that the rules that we presented do not give us the completeness of the ZX-calculus (interpreted in terms of linear maps [14]). However, this notable result has already been achieved, first in [20] for ZX-calculus with phases that are multiples of $\frac{\pi}{4}$, so with Clifford+T maps, and finally, universal completion of the ZX-calculus was given in [27] (with some follow-up papers which introduce alternative sets of rules).

2.1.5 Notation

2.1.5.1 Horizontal vs vertical notation

As circuits are typically oriented horizontally, not vertically, we will, from now on, if we are drawing ZX-diagrams that correspond to circuits, draw them horizontally, not vertically.

2.1.5.2 Wires

For ZX-diagrams that correspond to circuits, by the kth wire, we call the kth vertical line (counting from the top) and edges that it contains, which contains an input and an output.

2.1.5.3 CNOT

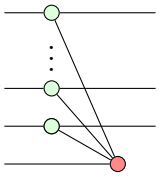
For simplicity, we will use the name $CNOT$ in the context of a ZX diagram, to describe the following diagram, which corresponds to the $CNOT$ gate. Moreover, we call the control wire, the wire to which the green node is connected, and the target wire, the wire to which the red node is connected.

(2.16)

2.2 Gadgets and phase gadgets

Using ZX-calculus we can construct multiple interesting diagrams and explore their properties. An interesting family of diagrams are gadgets and especially phase gadgets as presented in definitions 1 and 2 [23].

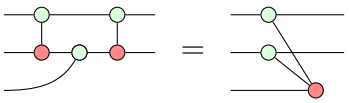
Definition 1.



A gadget on n wires is a diagram with $n+1$ inputs and n outputs, such that the last input is connected to a red node, which is connected to n green nodes, each of which is connected to exactly one input and exactly one output.

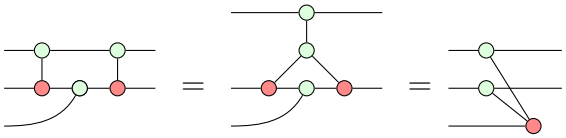
Before we consider the main advantages of phase gadgets, first let's focus on the underlying construction – what kind of quantum circuits are equivalent to the gadgets. To do it, let's first consider the following lemma, which answers this question for gadgets on two wires.

Lemma 1.



On the left-hand side, we have a gadget on n wires with one of its legs surrounded by two CNOT gates with control on some other wire. On the right-hand side, we have a gadget on $n+1$ wires.

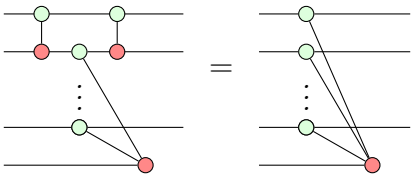
Proof.



□

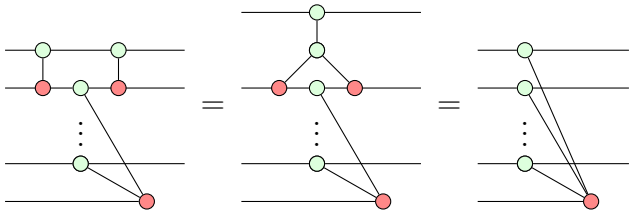
Using analogical reasoning, we can not only create the gadgets but also extend them, as presented in the following lemma 2.

Lemma 2.



First, we transform the initial diagram by merging two green spiders and splitting the new spider as presented above. Then we apply the complementary rule to get the final diagram.

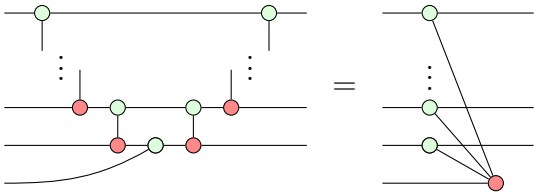
Proof.



This proof is a straightforward application of the strong complementarity equation. □

From this lemma we can conclude the following generalizations of lemma 1 – corollaries 1 and 2.

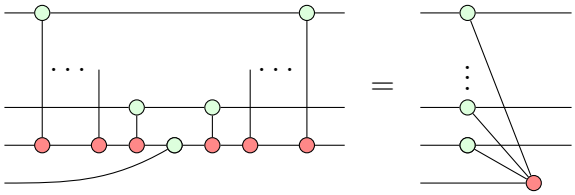
Corollary 1.



In this identity, on the right-hand side, we have just a gadget on n wired.

On the left-hand side, we have, going from the left, a green node on the first wire connected with a red node on the second wire, a green node on the second wire connected with a red node on the third wire, and so on up to a green node on $(n-1)$th wire connected with a red node on nth wire. Then we get a green node connected to the input of the phase gadget. Finally, we obtain a green node on $(n-1)$th wire connected to a red node on nth wire and so on up to a green node on the first wire connected with a red node on the second wire.

Corollary 2.



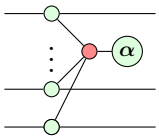
In this identity, on the right-hand side, we have just a gadget on n wired.

On the left-hand side, we have, going from the left, a green node on the first wire connected with a red node on the nth wire, a green node on the second wire connected with a red node on the nth wire, and so on up to a green node on $(n-1)$th wire connected with a red node on nth wire. Then we obtain a green node connected to the input of the phase gadget. Finally, we get a green node on $(n-1)$th wire connected to a red node on nth wire and so on up to a green node on the first wire connected with a red node on the nth wire.

2.2.1 Phase gadgets

We call a gadget that is connected to a single green spider (with possibly nonzero phase), a phase gadget (as in definition 2). Phase spiders have various properties that will be useful in our applications.

Definition 2.



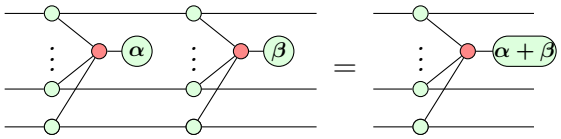
Phase gadget on n wires with phase α is a gadget on n wires with a green node with phase α connected to the last input.

2.2.1.1 Properties of phase gadgets

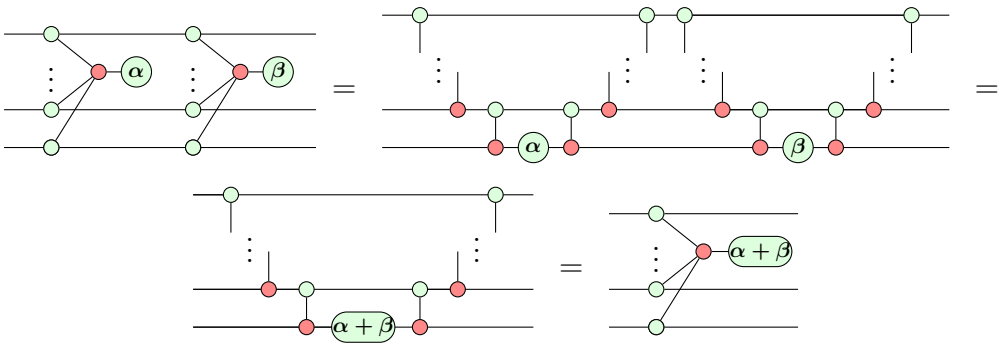
Commutativity Because all gadgets are connected to the wires by green spiders, all gadgets commute regardless of the sets of wires that they are connected to. This property holds not only for phase gadgets but for gadgets in general.

Additivity A diagram with two-phase gadgets on exactly the same set of wires is equivalent to a diagram with a single phase gadget on that set of wires with a phase equal to the sum of phases – as presented in lemma 3.

Lemma 3.



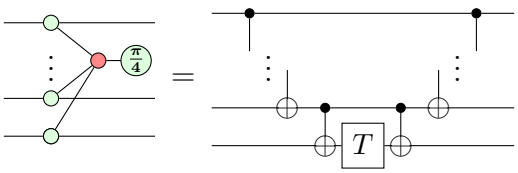
Proof.



First, we decompose both gadgets by corollary 1, then we eliminate adjacent not gates, by using the fact that pair of spiders with different colours connected by an even number of wires is equivalent to the two spiders not connected, and we merge two green spiders with phases α and β. Finally, we again use corollary 1 to get the desired spider. □

Easy translation to circuits Sometimes translating a non-unitary ZX diagram to an equivalent quantum circuit of minimal size poses a significant challenge [23]. In the case of phase gadgets, with phases that are multiples of $\frac{\pi}{4}$, we can do it as follows.

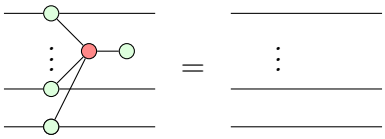
Observation 1.



Obvious relation to T-count Each phase gadget with a phase which is an odd multiple of $\frac{\pi}{4}$ increases the number of T gates in the circuit by exactly 1, regardless of the size of the spider.

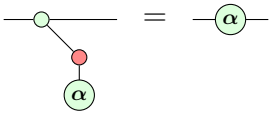
Phase gadget with no phase is an identity

Lemma 4.



Proof. By the strong complementarity, we transform the red node and green node with exactly one input, merge connected green nodes and remove all nodes with no phase, one input and one output. □

Phase gadget on a single wire



Phase gadget on one wire is equivalent to a green spider with the same phase on one wire as the red spider with no phase and two legs is a wire, and we can merge two connected green spiders.

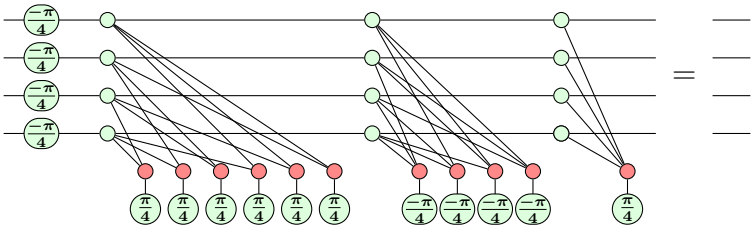
2.3 Spider nests

Except for the properties of single-phase gadgets (section 2.2.1.1), they manifest another, critical for our application, behaviour – that they can be composed to get spider nest identities [16, 15, 25].

2.3.1 Spider nest identity on 4 wires

The smallest, non-trivial [25] spider nest identity is the following identity on 4 wires.

Theorem 1.



For each size $s \in \{1, 2, 3, 4\}$ and each subset S of wires of size s, there is a phase gadget on these wires with phase $(-1)^s \frac{\pi}{4}$.

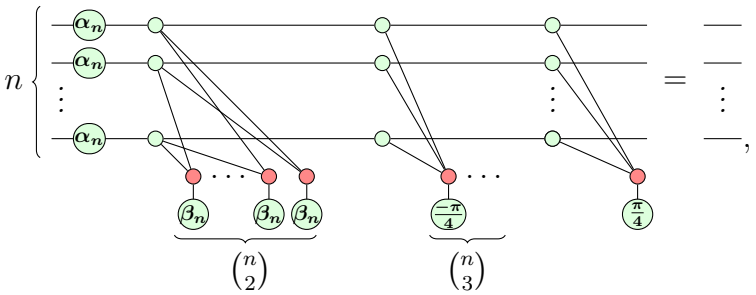
This identity has been proven in [5], and using only ZX-calculus, in [25]. Moreover, to get a "brute force" proof of this identity, it is sufficient to calculate in the matrix representation of the left-hand side diagram.

The significance of this single identity is clearly visible in the presented inductive proof of theorem 2, which, thanks to the identity on 4 wires, is quite straightforward and compact.

2.3.2 Spider nest identities of arbitrary size

An obvious limitation of the above identity is that it cannot be used to reason about bigger phase gadgets. This problem can be addressed by the introduction of spider nest identities on more wires, where the biggest phase gadget uses all of the wires.

Theorem 2.



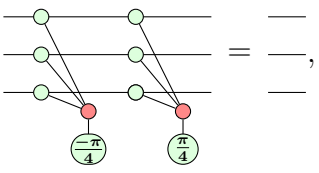
where $\alpha_n \coloneqq -(n-2)(n-3)\frac{\pi}{8}$ *and* $\beta_n \coloneqq (n-3)\frac{\pi}{4}$.

For each wire, there is a phase gadget with phase α_n *on this wire, for each pair of wires, there is a phase gadget with phase* β_n *on these wires, for each triplet of wires, there is a phase gadget with phase* $\frac{-\pi}{4}$ *on these wires, also there is a phase gadget of phase* $\frac{\pi}{4}$ *on all* n *wires.*

Remark 2. $\alpha_n = -(n-2)(n-3)\frac{\pi}{8}$ is always an integer multiple of $\frac{\pi}{4}$, as always one of $n-2$ and $n-3$ is an even number.

The following proof was introduced in [16] and relies heavily on theorem 1.

Proof. For $n = 3$ both $\alpha_n = 0$ and $\beta_n = 0$. Moreover, the whole identity simplifies to

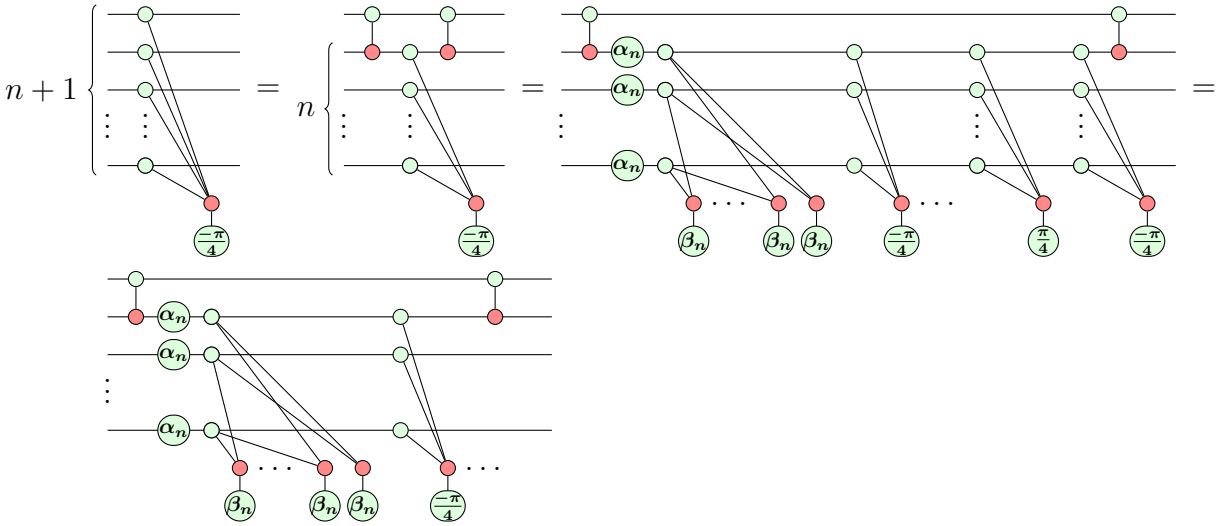


which holds by lemmas 3 and 4.

The case $n = 4$ is exactly theorem 1. We will also use this case as a base case for the induction. (We could use case $n = 3$ as the base case, but as we need case $n = 4$ (or theorem 1), as we will use it in the inductive step.)

If we assume that this identity holds for some fixed n, then to prove it for $n+1$, we take a phase gadget with phase $\frac{-\pi}{4}$ on $n+1$ wires, decompose it using lemma 2. Let's say that controls of the two created CNOT gates are on the first wire and targets on the second wire. Then we use the inductive assumption on n wires, and cancel out two gadgets on the same set of wires and with the same angles but with opposite signs by the properties of phase gadgets from section 2.2.1.1.

The step-by-step transformation of the diagrams looks as follows.



Now we commute one CNOT gate through all the phase gadgets until it reaches the other one, and when it does, we remove two consecutive CNOT gates. In that way, for all but the first two wires, we have gadgets on each set of 1, 2 or 3 wires with phases α_n, β_n and $\frac{-\pi}{4}$ respectively, and for each set of 2, 3 or 4 wires containing the first two wires we have phase gadgets with phases α_n, β_n and $\frac{-\pi}{4}$ respectively. In order to remove the phase gadgets on 4 wires, for each set of 4 wires on which we have a phase gadget, we use the spider nest identity of size 4. After this operation, all phase gadgets on 4 wires are "annihilated". Moreover

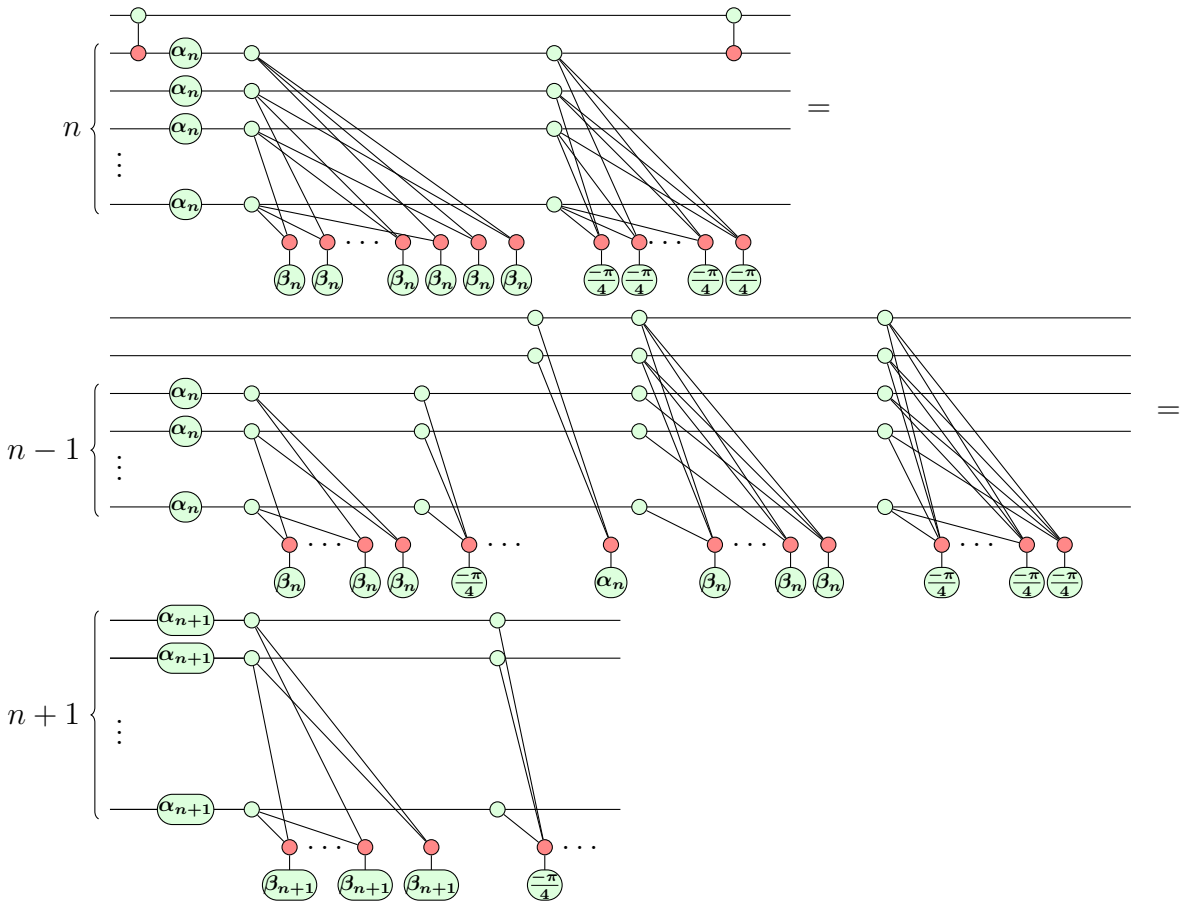
- for each single wire that is not the first or the second wire, we have a phase gadget on this wire with phase $\alpha_n + (n-2)\alpha_4 = -(n-2)(n-3)\frac{\pi}{8} - 2(n-2)\frac{\pi}{8} = -(n-1)(n-2)\frac{\pi}{8} = \alpha_{n+1}$ (phase α_n comes from the phase gadget that was on this wire initially, and $(n-2)\alpha_4$ comes from the spider nest identities of size 4 applied to $n-2$ sets containing this wire).

- for of the first and second wires we also have a phase gadget on this wire with phase $\binom{n-1}{2}\alpha_4 = -(n-1)(n-2)\frac{1}{2}\frac{\pi}{4} = \alpha_{n+1}$, as these wires are in $\binom{n-1}{2}$ sets to which the identities of size 4 are applied.

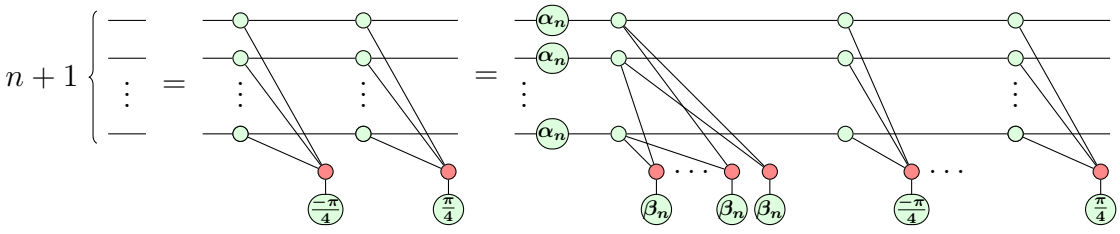
- for each pair of wires not including any of the first and second wires, we have a phase gadget on these two wires with phase $\beta_n + \beta_4 = (n-3)\frac{\pi}{4} + \frac{pi}{4} = \beta_{n+1}$.

- for each pair of wires with exactly one wire from the fires and second wires, we have a phase gadget on these two wires with phase $(n-2)\beta_4 = (n-2)\frac{\pi}{4} = \beta_{n+1}$.
- for first and second wire, we have a phase gadget on these wires with phase $\alpha_n + \binom{n-1}{2}\beta_4 = -(n-2)(n-3)\frac{\pi}{8} + (n-1)(n-2)\frac{1}{2}\frac{\pi}{4} = \beta_{n+1}$.
- Finally, for all sets of 3 wires we have a phase gadget with phase $\frac{-\pi}{4} = \beta_n + (n-2)\frac{-\pi}{4}$. This gadget either already was there, was added by a single spider nest identity of size 4, or was composed of one gadget with phase β_n and $(n-2)$ gadgets of phase $\frac{-\pi}{4}$.

Diagrammatic representation of these operations looks as follows.



Finally, in order to get the desired result, we combine two phase gadgets on $n+1$ wires to form an identity, as follows.



□

Remark 3. [25] presents a generalized version of this identity (see Theorem 5.1 in [25]), which proof also relies on induction with inductive step analogical to the presented proof of theorem 2.

2.3.3 Remarks on spider nest identities

As pointed in [15], phases α_n and β_n might be even multiples of $\frac{\pi}{4}$ (so integer multiples of $\frac{\pi}{2}$), and consequently have no influence on the T-count of the circuit. More precisely

- For odd n, β_n is an integer multiple of $\frac{\pi}{2}$, so all phase gadgets on 2 wires in this identity do not influence the T-count.
- For $n \equiv 2 \mod 4$, or $n \equiv 3 \mod 4$, α_n is an integer multiple of $\frac{\pi}{2}$, so all phase gadgets of size 1 in this identity do not influence the T-count.

These two observations combined mean that the spider nest identities of size n have T-count of:

- $\binom{n}{n} + \binom{n}{3} + \binom{n}{2} + \binom{n}{1}$ for $n \equiv 0 \mod 4$,
- $\binom{n}{n} + \binom{n}{3} + \binom{n}{1}$ for $n \equiv 1 \mod 4$,
- $\binom{n}{n} + \binom{n}{3} + \binom{n}{2}$ for $n \equiv 2 \mod 4$,
- $\binom{n}{n} + \binom{n}{3}$ for $n \equiv 3 \mod 4$.

2.3.4 Combining spider nest identities

As we pointed out before, the spider nest identities have T-count of $\frac{1}{6}n^3 + O(n^2)$. This might lead to difficulties when trying to effectively utilize them in a T-count reduction algorithm [16, 15] – in order to reduce the number of T gates in a circuit using a single such identity of increased size, multiple phase gadgets must be present circuit for the operation to give a significant decrease of T-count.

To overcome this difficulty, one can combine multiple spider nest identities in order to get composite spider nest identities with significantly lower T-count [16, 15]. The most basic example of such a combination is the one with two identities with a size difference of 1 where the smaller one using a subset of wires of the bigger one. This, allows us to reduce the T-count to $n^2 + O(n)$ [15].

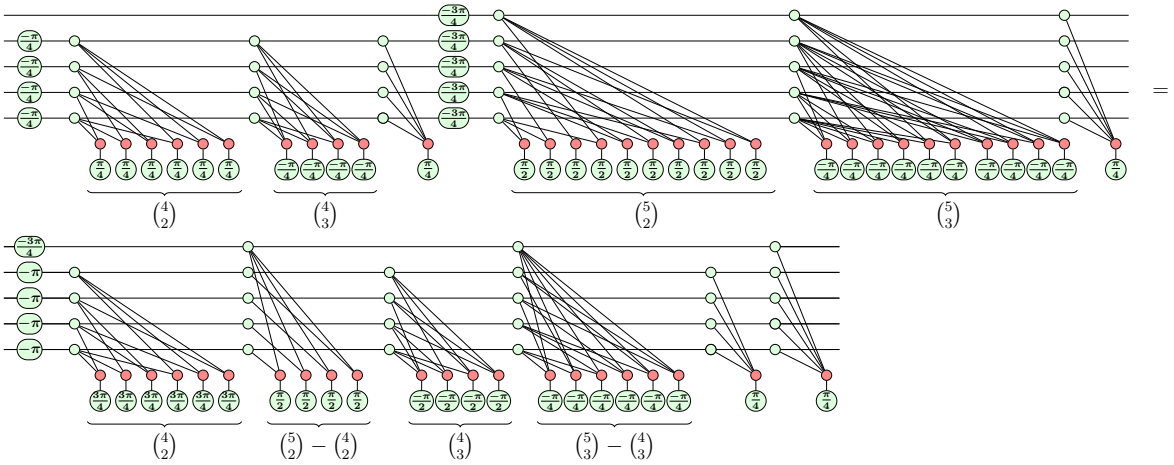


Figure 2.1: An example of an identity created by combining the spider nest identities on 4 and 5 wires.

2.4 PHAGE Tactic

In this section, we describe PHAGE Tactics, which were introduced in [15, 16].

The goal of the PHAGE Tactic is to reduce the T-count of diagonal circuits using families of circuits that are equivalent to an identity transformation. The spider nest identities are a notable example of such a family, and we, later in this chapter, discuss using them in that context.

2.4.1 Remark on circuits realizing identity transformation and circuits equivalence

Before we start considering this setup, we should note that, because of the unitary nature of quantum circuits, for every circuit, its adjoint circuit is its inverse. That means, that every equivalence of a form $A = B$, where A and B are quantum circuits, we can represent as $AB^\dagger = AB^{-1} = I$, where I is an identity transformation. That means that the choice of circuits equivalent to an identity transformation instead of just transformations of circuits of fixed size, in the case of the PHAGE Tactic will not be a limiting factor.

2.4.2 General PHAGE Tactic

The PHAGE Tactic is a very intuitive, greedy algorithm, which is parameterized with a family of identities. As described in [15], if we assume that we are given a family I_i of identities, the PHAGE Tactics applied to a circuit C works as follow:

1. Choose an identity I_i, such that I_i reduces the T-count of C.

2. Compute a circuit C', such that C' is C with applied I_i, and, if possible, further simplified (for example, if the considered circuit is composed of phase gadgets, then we might be able to reduce it using their additive property (section 2.2.1.1)).

3. Replace C with C' and repeat it until the T-count can be reduced.

Instances of this general algorithm can be found in various publications [15] (for example these from [6, 18]).

As noted in [15], the difficulty of effectively applying the PHAGE Tactics comes from the huge number of different subsets of wires to which the identities can be applied, as well as from the choice of the considered identities themselves. In section 5.1 we discuss the influence of the way we pick the next identity to be considered on the final result of the algorithm using real-world circuits as the examples.

It is also worth mentioning that the difficulty of the problem itself is tightly related to the difficulty of decoding the Reed-Muller codes [7].

2.4.3 PHAGE Tactics and spider nest identities

Combining the above algorithm, with the composite spider nest identities (which we describe in section 2.3.4), leads to the emergence of PHAGE4 and PHAGE5 tactics [15], which we describe here.

2.4.3.1 Definition of considered sets of identities

Before we look at the tactics themselves, let's first formally introduce the sets of identities that we will consider.

Definition 3. Let's say that, for finite $W \in \mathbb{Z}_+$, $|W| \geqslant 4$, $\mathcal{N}_W$ is a spider nest identity on wires from set W.

Now we define, $\mathbb{S}_i$, for $i \in \{4, 5\}$, as:

- $\mathbb{S}_4 \coloneqq \{N_{[4]}\}$,
- $\mathbb{S}_5 \coloneqq \left\{\mathcal{N}_{[5]}^{p_0}\mathcal{N}_{[5]\setminus\{1\}}^{p_1}\mathcal{N}_{[5]\setminus\{2\}}^{p_2}\mathcal{N}_{[5]\setminus\{3\}}^{p_3}\mathcal{N}_{[5]\setminus\{4\}}^{p_4}\mathcal{N}_{[5]\setminus\{5\}}^{p_5} : (p_i)_{i=0}^5 \in \{0,1\}^6 \setminus \{(0)^6\}\right\}$.

Remark 4. Some identities in $\mathbb{S}_5$ are equivalent up to a permutation of wires. This implicitly makes the PHAGE5 Tactic use them more often.

2.4.3.2 PHAGE4 and PHAGE5 Tactics

Let's take a list L of all elements of $\mathbb{S}_5$ (or $\mathbb{S}_4$).

The PHAGE5 (or PHAGE4) tactic work as follows:

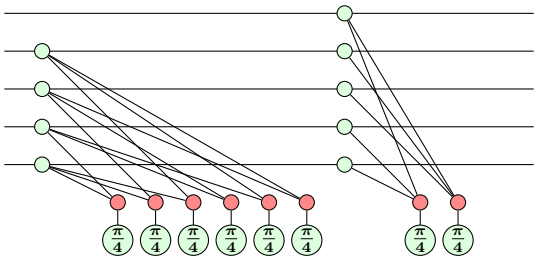
1. For each identity $\mathcal{I} \in L$, repeat R times (where R is some fixed constant):
   (a) Choose uniformly at random $S \subseteq [n]$, such that $|S| = 5$ (or $|S| = 4$).
   (b) Let $\mathcal{K}$ be the identity $\mathcal{I}$ but action on the qubits from the set S.
   (c) Apply the tactic PHAGE with the singleton set of identities $\{\mathcal{K}\}$.

These tactics, despite their simplicity, turned out to be of significant effectiveness [15], which indicates the expressiveness of the spider nest identities.

2.4.3.3 A small example of the advantage of composites spider nests

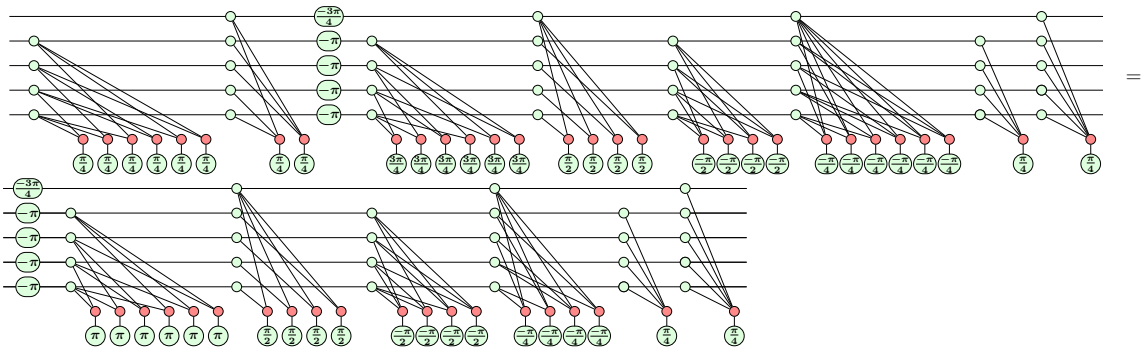
The greediness of the PHAGE Tactic makes the usage of the composite spider nests a significant advantage over a similar algorithm, but using only the spider nests on 4 and 5 wires as the identities. Let's illustrate it with the following, small example.

Let's take the following circuit (represented by a ZX diagram) on 5 wires.



This circuit consists of 8 phase gadgets, all with phase $\frac{\pi}{4}$, which means that it has T-count equal to 8. There are all possible 6 phase gadgets on each pair of the last four wires, as well as two phase gadgets on three wires – on first, second and third, and on first, fourth and fifth wire.

First of all, let's note that using a composite spider nest identity presented in section 2.3.4, we can reduce this circuit as presented below.



The resulting circuit has only 7 phase gadgets with phases that are odd multiples of $\frac{\pi}{4}$ – one on 1 wire, 4 on 3 wires, one on 4 wires and one on 5 wires – so it has a strictly smaller T-count.

Now, let's notice that using the spider nest identity on all five wires, we cannot reduce the T-count of this circuit, as it has no phase gadgets influencing the T-count on sets of 2 wires, and by using it, we can only "remove" the 2 phase gadgets on 3 wires, which leads to increase of the T-count from 8 to 20

If we try to apply a spider nest identity on the last four wires, then we will reduce all the phase gadgets on 2 wires, but we will still increase the T-count to 11 – we add 4 phase gadgets on 1 wire, 4 on 4 wires, 1 on 4 wires, and we don't reduce the phase gadgets on 3 wires that are connected to the first wire.

Finally, if we try to apply a spider nest identity on some other set of four wires, then we will reduce exactly 3 phase gadgets on 2 wires and one phase gadget on 3 wires, so we will have a circuit of T-count 16.

This example, although simple, illustrates well the power of a thoughtful way of choosing the considered identities.

2.5 Algorithm

In this section, we present a recap of the algorithm presented in [15] with a description of all individual parts.

The input of the algorithm is a circuit with gates NOT, $CNOT$, Toffoli, Z, CZ, CCZ, Hadamard, S, T, $SWAP$. The algorithm aims to reduce the T-count of the given circuit without changing its semantics. In order to achieve that goal, we proceed as follows.

1. Decompose all $Toffoli$ gates into Hadamard gate, CCZ gate and Hadamard gate.
2. Split the circuit into 3 parts:
   - The initial part, which contains only Clifford operations,
   - The main part, which contains all non-Clifford operations (so it is the only part that determines the T-count of the circuit),
   - The final part, which contains only Clifford operations.
3. Move as many Hadamard gates out of the main part as possible using the procedure `moveH` described below.
4. Replace all Haramard gates that remain in the main part, by Hadamard gate gadgetisation described below.

5. Decompose as many gates as possible into phase gadgets and commute all gates that are not phase gadgets to either the initial or the final part of the circuit as described below.

6. Treat the main part not as a sequence of gates, but as a set of gadgets. Sum the phases of gadgets that are using the same sets of wires.

7. Use the PHAGE Tactics with the spider-nest identities (as in section 2.4.3).

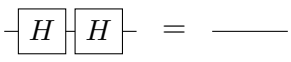
8. Transform the main part of the circuit back to a list of operations (in any order) and return the whole circuit.

Remark 5. The initial steps of the algorithm were first introduced in [18] and then modified in [15], in order to improve their performance.

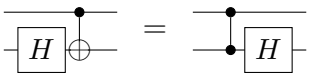
2.5.1 `moveH`

The procedure `moveH` attempts to move as many Hadamard gates from the main part of the circuit to the initial and final parts. It realizes it by the following rules.

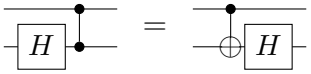
- Hadamard gate any other gate on strictly different wires commute,
- Two consecutive Hadamard on the same wire cancel out,



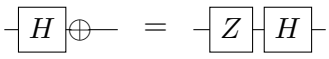
- Hadamard gate commutes through $CNOT$ changing it to CZ,



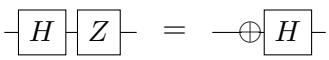
- Hadamard gate commutes through CZ changing it to $CNOT$,



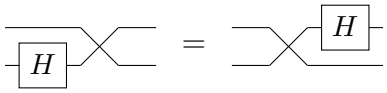
- Hadamard gate commutes through NOT changing it to Z,



- Hadamard gate commutes through Z changing it to NOT,

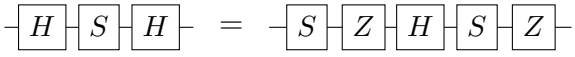


- Hadamard gate commutes through $SWAP$ by changing the wire to which it is applied.



Moreover, if a gate following a Hadamard gate does not commute, then it tries to move the following gate first by similar rules, for example, we can use these rules, among others, to move the $CNOT$ gate.

It also tries to apply the following rule, which does not commute the Hadamard gate but decreases the number of Hadamard gates in the main part.

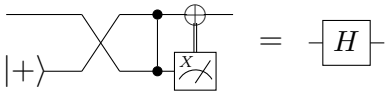


Moreover, it tries to use the following two rules, which does not decrease the number of Hadamard gates and does not move the Hadamard gates past the following gates, but as they change the location of the Hadamard gate, they might allow a one that is stuck, to commute further.

Combining all these rules we get a procedure that moves all Hadamard gates to one end of a circuit. We apply it two times to the main part - once to commute as many gates as possible to the final part of the circuit, and then once to move as many as possible remaining Hadamard gates to the initial part of the circuit. In that way, we reduce the number of Hadamard gates in the main part without increasing the number of wires and T-count.

2.5.2 Hadamard gate gadgetisation

To decompose the Hadamard gate, we use Hadamard gadgetisation [18], which can be represented as follows.



(2.17)

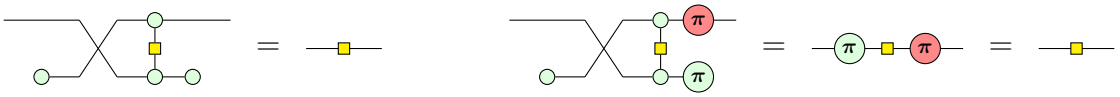
That means that using a single auxiliary qubit, $SWAP$ and CZ gates and one classically controlled NOT operation, we can decompose a Hadamard gate without introducing any

additional T gates, or in other words, the Hadamard gate can be rephrased as a post-selection which does not affect the T-count of a circuit. Moreover, as we will shortly see, the CZ gate can be easily transformed into a phase gadget.

For each Hadamard state that we have in the main part, we need to prepare an auxiliary state in the initial phase and perform a classically controlled NOT operation in the final phase. Unfortunately, this procedure increases the number of wires in the main part by the number of remaining Hadamard gates, so the procedure $moveH$ is indeed crucial for the effectiveness of this algorithm.

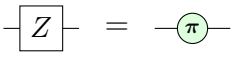
In terms of ZX-diagrams, one might rephrase eq. (2.17) as the following observation 2.

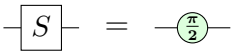
Observation 2.

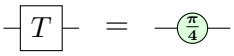


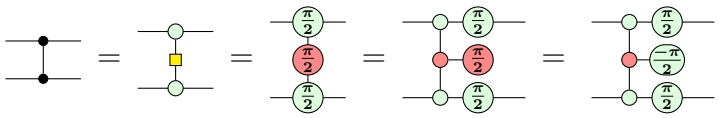
2.5.3 Decomposing gates into phase gadgets

Using the following identities for we transform the following gates into phase gadgets.

- 

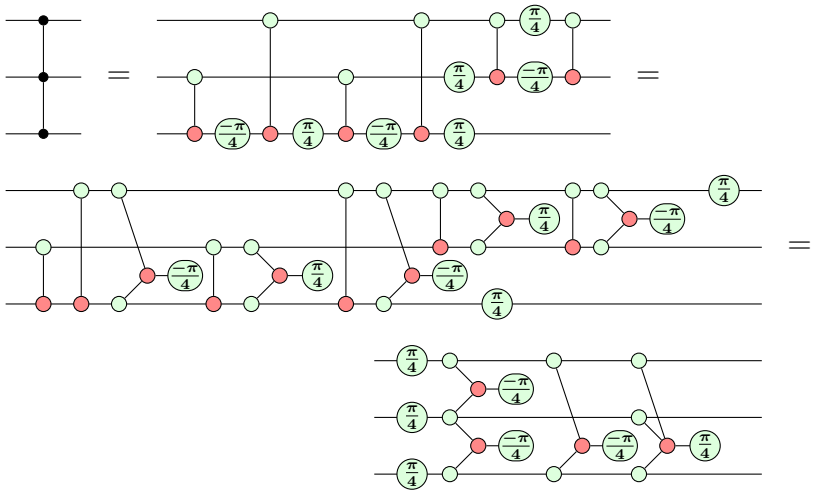
- 

- 

- 

- We can obtain the decomposition of CCZ gate in multiple ways. We move by a "brute force" all $CNOT$ gates to one side of the circuit, as it is presented in the following section, such that they cancel out, and we get the following identity. We present only the first step of the process of commuting the $CNOT$ gates as all other

steps are analogical.



2.5.4 Commuting other gates

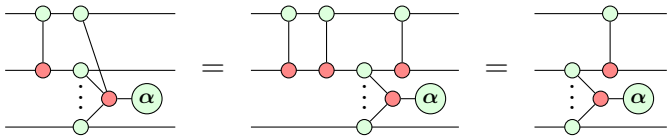
After the previous step, the main part of the circuit consists only of the phase gadgets and $SWAP$, $CNOT$ and (possibly classically controlled) NOT gates. Now we commute each of these gates to either the initial or final part of the circuit.

2.5.4.1 SWAP

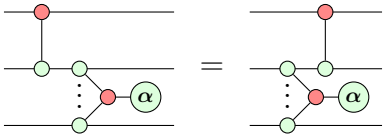
The commutation of the $SWAP$ gate is straightforward. We just swap the two swapped wires.

2.5.4.2 CNOT

In one case, we commute $CNOT$ gate using lemma 2 and cancelling out two consecutive $CNOT$ gates, as follows.

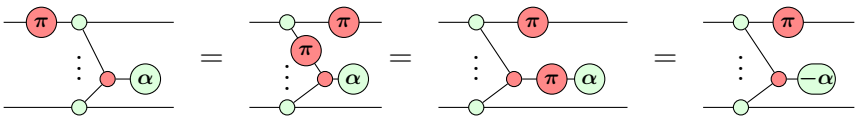


The second possible case, as demonstrated below, is straightforward.

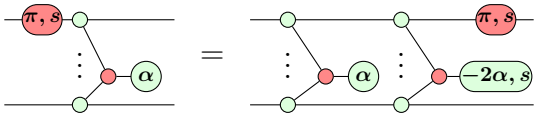


2.5.4.3 NOT

To commute the (possibly classically controlled) NOT gate, we first consider the commutation of NOT gate as follows.



Which, by lemma 3, means that we can represent the commutation of classically controlled NOT gate as follows.



This representation is particularly fortunate, as, because we only use α that is a multiple of $\frac{\pi}{4}$, the controlled phase gadget always is a multiple of $\frac{\pi}{2}$, so we can move it to the final part and keep all the phase gadgets with odd multiples of $\frac{\pi}{4}$, so all influencing the T-count, in the main part.

2.5.4.4 Direction of commutation

It is of high importance to note that the above commutation rules can be used to commute the gates in both directions. The direction of commutation of $SWAP$ gate, because of the nature of its commutation, does not influence the algorithm. It is similar with (possibly classically controlled) NOT gates, as they only negate the angles and introduce the phase gadgets with even multiples of $\frac{\pi}{4}$.

The commutation of the $CNOT$ gate is the only source of introducing phase gadgets on multiple wires. To minimize the average size of phase gates that are present in the main part of the circuit, the direction of the commutation is based on the difference of the number of phase gadgets that will be expanded by the given $CNOT$ gate (such that the common wire of the $CNOT$ and the phase gadget is the target of the $CNOT$) and phase gadgets that will get the number of wires decreased by the given $CNOT$ gate (such that both wires used by $CNOT$ are also used by the phase gadget).

Chapter 3

Rephrasing the phase gadgets T-count reduction

The final step of the algorithm, the PHAGE Tactics is used to optimize the T-count of a circuit expressed entirely using phase gadgets. Let's phrase it as a completely formal and abstract computational problem. In order to do so, let's first introduce the following definition.

Definition 4. For a set $\mathcal{A} \in P(P([k]))$ and an injective function $\sigma : [k] \to [n]$ we denote $\sigma(\mathcal{A}) \coloneqq \{\{\sigma(a) : a \in A\} : A \in \mathcal{A}\}$.

Remark 6. We use a standard notation $[n] \coloneqq \{1, ..., n\}$.

Problem 1. [Optimal phase gadgets T-count reduction]

Input: Positive integers n, k and l_i (for $i \in [k]$), a set $C \in P(P([n]))$, and k sets $I_i \in P(P([l_i]))$.

Output: A set $C' \in P(P([n]))$ of a form $C' \coloneqq C \triangle \sigma_1(I_{i_1}) \triangle ... \triangle \sigma_m(I_{i_m})$, where $i_j \in \{1, ..., k\}$ and $\sigma_{i_j} : [l_{i_j}] \to [n]$ are injective functions, such that C' has minimal possible size.

3.1 Correspondence to the PHAGE Tactics

The goal of the PHAGE Tactics is to reduce the number of phase gadgets with a phase which is an odd multiple of $\frac{\pi}{4}$ in the circuit as much as possible. Note that as we mentioned before – each such phase gadget corresponds to exactly one T gate. In that way, we can treat the set C as the description, where a single set $g \in C$ indicates a single phase gadget g on wires $w \in g$ with a phase which is an odd multiple of $\frac{\pi}{4}$.

In a similar way, we represent the identities that can be used in the *Phase* tactics. Each identity is represented by a set I_i, which is set containing some phase gadgets with odd multiples of $\frac{\pi}{4}$. The gadgets with even multiples of $\frac{\pi}{4}$ are ignored, as they don't

influence the T-count in any way. An example of such representation of identity is the representation of a spider nest identity of size 4, which is $P([4]) \setminus \{\varnothing\}$.

The injective functions σ_{i_j} correspond to choices of wires on which we want to apply the chosen identities. It is worth pointing out here that thanks to the commutative property of the phase gadgets (section 2.2.1.1), the choice of wires is sufficient to determine how the identity will transform the circuit.

Moreover, because of the additive property of the phase gadgets (section 2.2.1.1), a symmetric difference of the described representations of circuit and identity is a representation of a circuit obtained by applying chosen to identify to that circuit.

Let's note that, as mentioned in [15], the PHAGE Tactics, is an example of a greedy algorithm, which aims to solve this abstract problem. The optimization used in [15] – usage of composite spider nest identities – aims to improve the performance of the heuristics and is very effective in practice [15].

3.2 T-count optimization using spider nest identities

As the problem 1 is parametrized with the considered identities, we can give a more specific formulation – using exactly the identities that we consider, the spider nest identities, that will correspond to our application.

First let's define the identities in terms of sets (these are fixed, and are not parameters of the problem).

Definition 5. For $n \geqslant 4$, we define

$$\mathcal{I}_n \coloneqq \begin{cases} \binom{[n]}{n} \cup \binom{[n]}{3} \cup \binom{[n]}{2} \cup \binom{[n]}{1} & \text{if } n \equiv 0 \mod 4 \\ \binom{[n]}{n} \cup \binom{[n]}{3} \cup \binom{[n]}{1} & \text{if } n \equiv 1 \mod 4 \\ \binom{[n]}{n} \cup \binom{[n]}{3} \cup \binom{[n]}{2} & \text{if } n \equiv 2 \mod 4 \\ \binom{[n]}{n} \cup \binom{[n]}{3} & \text{if } n \equiv 3 \mod 4 \end{cases}$$

where $\binom{A}{k} \coloneqq \{B \subseteq A : |B| = k\}$ (so for example $\binom{[n]}{n} = \{[n]\}$, or $\left|\binom{[n]}{k}\right| = \binom{n}{k}$).

Problem 2. [Optimal phase gadgets T-count reduction using spider nest identities]

Input: Positive integer n and a set $C \in P(P([n]))$.

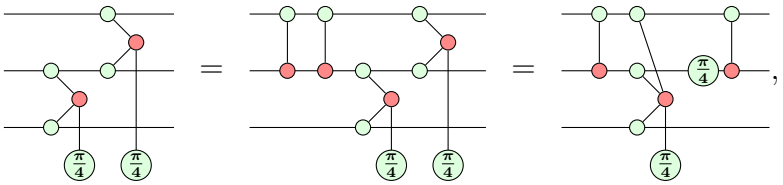
Output: A set $C' \in P(P([n]))$ of a form $C' \coloneqq C \triangle \sigma_1(\mathcal{I}_{i_1}) \triangle ... \triangle \sigma_m(\mathcal{I}_{i_m})$, where $i_j \in \{4, ..., n\}$ and $\sigma_{i_j} : [i_j] \to [n]$ are injective functions, such that C' has minimal possible size.

Remark 7. As all the identities $\mathcal{I}_k$, are symmetric – i.e. for every permutation $\sigma : [k] \to [k]$, $\sigma(\mathcal{I}_k) = \mathcal{I}_k$ – in the problem 2, instead of choosing the injective functions $\sigma_{i_j} : [i_j] \to [n]$, we can choose subsets of $[n]$ of size i_j.

Such a problem naturally corresponds to the problem that we consider – minimizing the T-count of a circuit using spider nest identities. The sets (or identities) $\mathcal{I}_n$, as described in section 3.1, are created from the spider nest identities, and contain exactly the sets of wires to which the phase gadgets with phases that are odd multiples of $\frac{\pi}{4}$ are connected.

3.2.1 Completeness of spider nest identities

When given such a general problem, a natural question about its completeness arises – if given a circuit on n wires, consisting of only phase gadgets with phase $\frac{\pi}{4}$ on sets of wires $C \subseteq P(P([n]))$, if we can reduce the T-count of such circuit to m, can we also get a set C', such that $|C'| = m$, in problem 2. The answer to this question is not apriori obvious, as there are some circuits transformations that we can perform on the circuits, but not in our problem – for example, for $n = 3$, we can do the following transformation of circuits,



, (3.1)

but using only spider nest identities, we cannot express the transformation

$$\{\{2,3\},\{1,2\}\} \mapsto \{\{1,2,3\},\{2\}\}\,.$$

Note that these sets, even though they don't represent equivalent circuits, should we considered equivalent in our problem, as they are equal up to preceding and following Clifford transformations (and we already ignore phase gadgets representing Clifford operations in our definition of the problem).

3.2.2 Extending the problem

If the spider nests are not enough for completeness result for this problem, one might consider an extended version of that problem with added relevant transformations. Following the above example, we can present such an extended version in the following way.

Definition 6. We call a function $f : P(P([n])) \to P(P([n]))$ such that

$$f(A) \coloneqq \{X \in A : c \notin X\} \cup \{X \triangle \{t\} : c \in X\}$$

a (t, c) $CNOT$ transition.

Remark 8. A (t, c) $CNOT$ transition corresponds to the transformation of phase gadgets in a circuit when a $CNOT$ gate commutes from one end of the circuit to another. t corresponds to the target wire, and c corresponds to the control wire.

Problem 3. [Optimal phase gadgets T-count reduction using spider nest identities and $CNOT$ transitions]

Input: Positive integer n and a set $C \in P(P([n]))$.

Output: A set $C' \in P(P([n]))$ of a form $C' \coloneqq f_m\left(f_{m-1}\left(...f_1\left(f_0\left(C\right) \triangle \sigma_1(\mathcal{I}_{i_1})\right) \triangle ...\right) \triangle \sigma_m(\mathcal{I}_{i_m})\right)$, where $i_j \in \{4, ..., n\}$ and $\sigma_{i_j} : [i_j] \to [n]$ are injective functions, such that C' has minimal possible size, and where for every $l \in \{0, 1, ..., m\}$, f_l is either an identity, or a composition of (t, c) $CNOT$ transitions (for $t, c \in [n]$).

3.3 Significance and open questions

Such a formal approach allows us to better understand what the spider nests are capable of and consider their possible limitations.

As the last part of this chapter, we leave an open question – whether the spider nest identities (or spider nest identities with $CNOT$ transitions) are complete in terms of T-count reduction.

Remark 9. Let's note that we ask about the completeness in terms of T-count reduction, as eq. (3.1) instantly proofs that spider nests themselves are not complete in terms of non-Clifford phase gadget transformations up to Clifford operations.

Chapter 4

Dense spider nest identities

In this chapter, we define and prove the existence of dense spider nest identities, which are an alternative (and possibly more natural) way of expressing the spider nest identities.

Definition 7. We say that a composite spider nest identity is dense, if, for every nonempty set of wires, it has a phase gadget on this set of wires with a phase which is an odd multiple of $\frac{\pi}{4}$. We call such a composite spider nest identity, a dense spider nest identity.

4.1 Existences of all dense spider nest identities

Theorem 3. *For every $n \geqslant 4$, there exists a dense spider nest identity on n wires.*

Proof. To prove this theorem, let's give a precise example.

Theorem 4. *A circuit composed of phase gadgets such that for each $W \subseteq [m]$, $W \neq \varnothing$, if*

- *$|W| \geqslant 4$, then on set of W of wires, we have a phase gadget with phase $\frac{\pi}{4}$,*
- *$|W| = 3$, then on set W of wires, we have a phase gadget with phase*

$$\frac{-\pi}{4}\left(2^{n-3}-1\right),$$

- *$|W| = 2$, then on set W of wires, we have a phase gadget with phase*

$$\frac{\pi}{4}\left(2^{n-3}n-2^{n-1}+1\right),$$

- *$|W| = 1$, then on set W of wires, we have a phase gadget with phase*

$$\frac{-\pi}{4}\left(7*2^{n-3}-7*2^{n-4}n+2^{n-4}n^2-1\right),$$

is equivalent to an identity transformation.

Proof. We compose the circuit out of spider nest identities on all sets of wires $W \subseteq [n]$ such that $|W| \geqslant 4$.

By the definition of the spider nest identities, on each set of at least 4 wires, we have a phase gadget with phase exactly $\frac{\pi}{4}$.

For a fixed set W of 3 wires, we get a phase gadget on W with phase $\frac{-\pi}{4}$ for each set of at least 4 wires, that is a superset of W, which means that sum of phases of these gadgets will be equal to

$$\frac{-\pi}{4} \sum_{k=4}^{n} \binom{n-3}{k-3} = \frac{-\pi}{4} \left(2^{n-3} - 1\right).$$

For a fixed set W of 2 wires, we get a phase gadget on W with phase $\frac{\pi}{4}(k-3)$ for each set of $k \geqslant 4$ wires, that is a superset of W, which means that sum of phases of these gadgets will be equal to

$$\frac{\pi}{4} \sum_{k=4}^{n} (k-3) \binom{n-2}{k-2} = \frac{\pi}{4} \left(2^{n-3} n - 2^{n-1} + 1\right).$$

For a fixed set W of 1 wire, we get a phase gadget on W with phase $\frac{-\pi}{8}(k-2)(k-3)$ for each set of $k \geqslant 4$ wires, that is a superset of W, which means that sum of phases of these gadgets will be equal to

$$\frac{-\pi}{4} \sum_{k=4}^{n} \frac{(k-2)(k-3)}{2} \binom{n-1}{k-1} = \frac{-\pi}{4} \left(7 * 2^{n-3} - 7 * 2^{n-4} n + 2^{n-4} n^2 - 1\right).$$

□

Now it remains to notice, that for $n \geqslant 4$, all numbers

- 2^{n-3},
- $2^{n-3} n - 2^{n-1}$,
- $7 * 2^{n-3} - 7 * 2^{n-4} n + 2^{n-4} n^2$,

are even. It concludes the proof of theorem 3. □

4.2 Elegant dense spider nest identities

The expressions for phases of small phase gadgets in theorem 4 are complicated enough not to give us precise information about the phases at a glance. Let's solve this problem with the following theorem and two observations.

Theorem 5. *For $n \geqslant 6$, a circuit composed of phase gadgets on all nonempty subsets of $[n]$ with phases exactly $\frac{\pi}{4}$ is equivalent to an identity transformation.*

Proof. It is enough to notice that all the numbers

- 2^{n-3},
- $2^{n-3}n - 2^{n-1}$,
- $7 * 2^{n-3} - 7 * 2^{n-4}n + 2^{n-4}n^2$,

for $n \geqslant 6$ and are multiples of 8 and use theorem 4. □

4.2.1 Two small dense spider nest identities

Finally we have two remaining dense spider nest identities, for $n = 4$ and $n = 5$.

Remark 10. The dense spider nest identity on 4 wires is exactly the spider nest identity on 4 wires, so it has

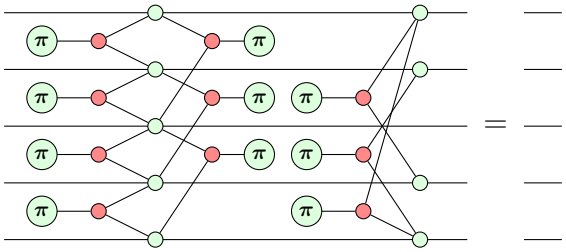
- phase gadgets on 1 wire with phases $\frac{-\pi}{4}$,
- phase gadgets on 2 wires with phases $\frac{\pi}{4}$,
- phase gadgets on 3 wires with phases $\frac{-\pi}{4}$,
- phase gadget on 4 wires with phase $\frac{\pi}{4}$.

Remark 11. The dense spider nest identity on 5 wires has

- phase gadgets on 1 wire with phases $\frac{\pi}{4}$,
- phase gadgets on 2 wires with phases $\frac{-3\pi}{4}$,
- phase gadgets on at least 3 wires with phases $\frac{\pi}{4}$.

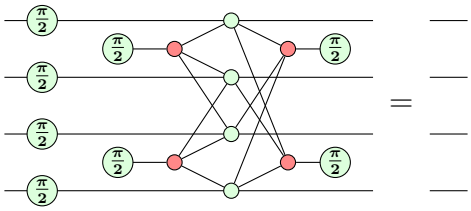
Finally, in order to reduce these two special cases, we can use the two following lemmas.

Lemma 5.

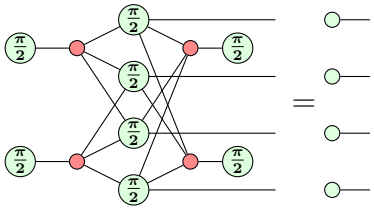


Proof. It follows directly from the π-copy rule (see π-copy rule in [14]). □

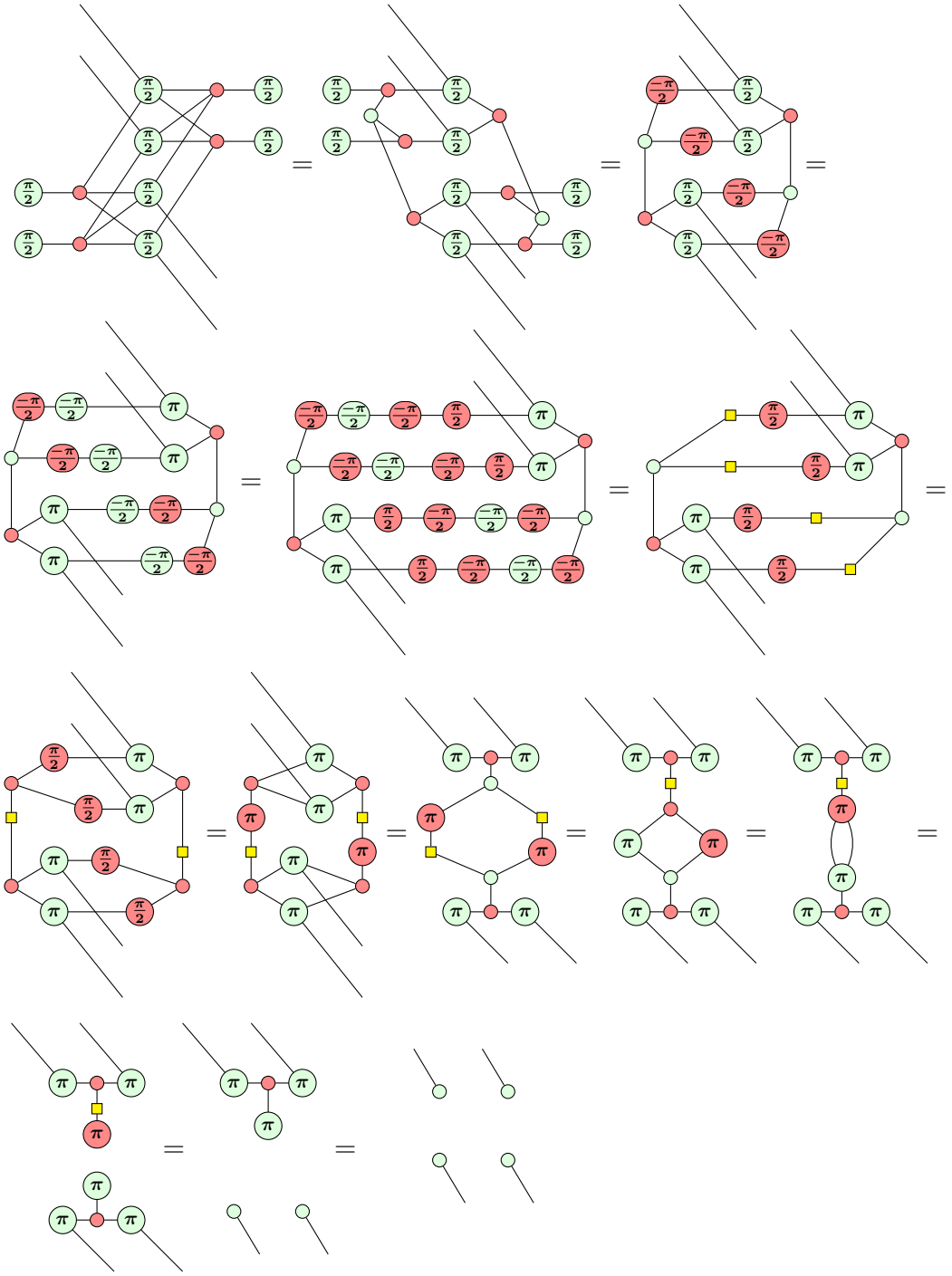
Lemma 6.



Proof. In order to prove this lemma, let's note that it is sufficient to prove the following identity, as then we can connect it, using green nodes, to four wires.



To do so, we first, using the strong complementarity, merge two pairs of gadgets (which is a generalization of their additive property). Then we transform the green $\frac{\pi}{2}$ spiders (with one input) to red spiders with phases $\frac{-\pi}{2}$ (see the Proposition 9.109 in [14]). Then, in a few steps, we create Hadamard gates (using equation (9.114) from [14]). Now, in order to use the strong complementarity again, we move Hadamard gates and red spiders and use it. Finally, we break the symmetry of the circuit, by moving both Hadamrd gates towards the same pair of outputs and moving green and red nodes with phases π down and up respectively. The few last steps are just removing the doubled wires, changing the colour of a red spider, and the π-copy rule (see π-copy rule in [14]).



□

4.3 All elegant dense spider nest identities

Combining all previous observations we can finally state the following, final theorem about elegant dense spider nest identities.

Theorem 6. *For $n \geqslant 4$, a circuit composed of phase gadgets on all nonempty subsets of $[n]$ with phases exactly $\frac{\pi}{4}$ is equivalent to an identity transformation.*

Proof. Follows directly from theorem 5, remarks 10 and 11, and lemmas 5 and 6. □

4.4 Modification of problems from chapter 3

Using theorem 3, we can instantly simplify the formulations of the problems 2 and 3 by replacing $\mathcal{I}_n$ with $\mathcal{P}([n]) \setminus \{\varnothing\}$ (for each $n \geqslant 4$), or simplify it even further, by replacing $\sigma_j(\mathcal{I}_{i_j})$ by $P(A_j) \setminus \{\varnothing\}$ for $A_j \subseteq [n]$, $|A_j| \geqslant 4$.

A formulation of problem 2, after this simplification, looks as follows.

Problem 4. [Optimal phase gadgets T-count reduction using dense spider nest identities]

Input: Positive integer n and a set $C \in P(P([n]))$.

Output: A set $C' \in P(P([n]))$ of a form $C' \coloneqq (C \triangle P(A_1) \triangle ... \triangle P(A_j)) \setminus \{\varnothing\}$, where for each $i \in [j]$, $A_i \subseteq [n]$ and $|A_i| \geqslant 4$, minimizing $|C'|$.

4.4.1 Natural generalisation

The formulation of problem 4 leads to a natural generalisation, where we consider powersets of sets of sizes restricted by some given integer, which in the case of problem 4 is equal to 4.

Problem 5. [Restricted optimal phase gadgets T-count reduction using big and dense spider nest identities]

Input: Positive integers k and n, and a set $C \in P(P([n]))$.

Output: A set $C' \in P(P([n]))$ of a form $C' \coloneqq (C \triangle P(A_1) \triangle ... \triangle P(A_j)) \setminus \{\varnothing\}$, where for each $i \in [j]$, $A_i \subseteq [n]$ and $|A_i| \geqslant k$, minimizing $|C'|$.

Remark 12. Problem 2 for $k \geqslant 4$ corresponds to a T-count reduction in a circuit composed of phase gadgets using dense spider nest identities of a size at least k. For $k < 4$ there is no obvious correspondence to the problem of T-count reduction. Moreover, for $k = 1$ the problem has a trivial solution.

Chapter 5

Modifications of the PHAGE Tactics

In this chapter, we focus on various modifications and improvements of the used PHAGE Tactics (which we described in section 2.4). We discuss their influence on the result of T-count reduction, examine their strong and week sides, as well as present quantitative results of the improved methods.

5.1 Order matters

In this section, we focus on the influence of the order of identities used in the PHAGE5 tactics on the final output of the algorithm. We quantitatively compare results for different orders and present our results with conclusions.

5.1.1 Motivation

Results presented in [15] reached and improved the state of the art for multiple of the considered circuits, however in the case of two noticeably small circuits, namely Barenco Tof_4 [15] (also know as $\Lambda_4(X)$ [6, 8]) and NC Toff_5 [15], it didn't reach the state of the art results missing them by a small margin – one T gate. That made us consider modifications of the algorithm that would improve results in cases of these circuits, and possibly in the case of both families of circuits from which these two circuits come. As the PHAGE5 Tactic is using a fixed list of identities (as described in section 2.4.3) which are considered in the given order, the natural question is how and if will reshuffling the list affect the efficiency of the procedure.

5.1.2 Process of constructing the order

For the sake of simplicity of the description, let's define PHAGE5(L), as a procedure analogical to PHAGE5, but using a different list of considered identities.

To systematically find out how the order of identities affect the result of PHAGE5(L) Tactics, we used the following procedure.

1. Choose a fixed number K (in our case we mostly used $K = 10$).
2. Prepare and store K empty lists.
3. Repeat until the desired T-count is reached.
   (a) For every list L of the K lists, and every element $\mathcal{I}$ of $\mathbb{S}_5$ such that $\mathcal{I}$ is not in L, take L' which is L with added $\mathcal{I}$ as the last element.
   (b) Execute the algorithm with PHAGE5(L') Tactic.
   (c) Choose K lists L' achieving the best results as the new stored lists.

Using this simple procedure we were able to extract various lists of length up to 5 such that both circuits of our interests reached the desired result. By considering the monotonicity of various features of the lists (for example if from some point on, every identity used was created with nonzero power of $\mathcal{N}_{[5]}$), we noticed that they share a common feature – they were decreasing with respect to the T-count. As this feature is not determining a total order of the identities, we sampled various orders holding this feature and confirmed that in all the cases we reached the desired T-count.

From now on, if we do not explicitly specify otherwise, the order refers to an instance of order in which identities with higher T-count are followed by the identities with lower T-count (decreasing with respect to T-count).

5.1.3 Results

As we found out, it is possible to affect (and sometimes improve) the result of PHAGE5 tactics. Unfortunately, this result turned out to be only a partial success, as in the case of some circuits, the opposite order was a strictly better choice. With multiple attempts, we were not able to find a single order that "suits them all". Moreover, identification an optimal order of identities, or at least deciding which out of multiple orders is the best, is a challenging task itself.

In table 5.1 we present a comparison of results from [15] and our results – using the described above order as well as the revered order.

Circuit	initial $\#T$	result from [15] $\#T$	the order $\#T$	the reversed order $\#T$
Adder_8	399	176	209	208
Barenco Tof_3	28	13	14	13
Barenco Tof_4	56	25	24	25
Barenco Tof_5	84	37	36	37
Barenco Tof_{10}	224	97	98	98
CSLA MUX_3	70	44	45	50
CSLA MUX_9	196	84	84	84
GF(2^4) Mult	112	53	56	57
GF(2^5) Mult	175	88	88	86
GF(2^6) Mult	252	128	130	128
GF(2^7) Mult	343	167	173	184
GF(2^8) Mult	448	229	236	232
GF(2^9) Mult	567	306	281	277
GF(2^{10}) Mult	700	357	355	353
GF(2^{16}) Mult	1792	972	973	966
Mod5_4	28	7	7	7
Mod Adder_{1024}	1995	1010	1007	1009
Mod Mult_{55}	49	19	20	20
Mod Red_{21}	119	65	61	63
QCLA Adder_{10}	238	147	152	143
QCLA Com_7	203	84	79	83
QCLA Mod_7	413	233	229	231
RC Adder_6	77	38	41	38
NC Toff_3	21	13	13	13
NC Toff_4	35	19	19	19
NC Toff_5	49	26	25	26
NC Toff_{10}	119	60	59	60
VBE Adder_3	70	20	20	20

Table 5.1: Comparison of initial T-count of the circuits, results presented in [15], achieved by the PHAGE5 tactics with identities ordered by the T-count in decreasing, and increasing order. All results, except those for Mod Adder_{1024} and QCLA Mod_7 circuits, were verified using [3].

5.1.4 Usage in practical applications

Even though we were not able to create an "optimal" ordering of the identities, we should note that in practical quantum circuit compilers, a useful approach would be the usage of PHAGE5(L) algorithm with various choices of L. Moreover, by utilizing the parallel computing capabilities of the modern CPUs, it would be feasible to execute simultaneously multiple instances of the PHAGE5(L) at the same time

5.2 Other modifications

Here we present some other modifications of PHAGE Tactics that we considered, which did not give us any notable results. We also discuss our approaches and related problems.

5.2.1 Changing the distribution of the identities

As pointed out in [15], and as we pointed in section 2.4.3, in the set of identities considered in PHAGE5 tactic, there are multiple occurrences of identities that are equivalent up to a permutation of wires. This makes us consider a natural question – is this distribution the optimal one, or maybe, could we improve the algorithm by choosing a different one.

5.2.1.1 Results

In order to check the influence of the distribution of identities, we tested various different distributions. For example a uniform distribution of all 10 unique (up to permutation of wires) composite spider nest identities, distribution proportional to the inverse of T-count and others. The results of this approach, as we quickly noticed, were highly dependent on the order of the considered identities, not on their distribution (assuming that the same number of identities was chosen and that we did not skip any identities). As we already considered the influence of the distribution on the result, we moved to different approaches.

5.2.2 Problems with bigger identities

A different modification of the algorithm that we considered, was incorporating bigger spider-nest identities. In our case, we have chosen to test the identities of sizes from 6 to 8. Before we further describe this approach, let's first discuss more the composite spider nest identities.

5.2.2.1 Composite spider nest identities on n wires

In order to classify the composite spider nest identities on n wires, let's have the following lemma.

Lemma 7. *There exists a bijection between composite spider nest identities on n wires with phase either 0 or $\frac{\pi}{4}$ for all phase gadgets on at least 4 wires and subsets of $P([n])$ containing elements of size at least 4.*

Proof. If we choose $A \subseteq P([n])$ such that for every $a \in A$, $|a| \geqslant 4$, then we create an unique composite spider nest identity from this set, as composition of all $\mathcal{N}_a$ for $a \in A$ (see definition 3). Let's note that two composite spider nest identities created from two different sets A and B are different – if $a \in A$ and $a \notin B$, then the phase gadget on wires from a has phase $\frac{\pi}{4}$ in the identity created from A and phase 0 in the identity created from B.

Moreover, for a given composite spider nest identity, we can create a set A corresponding to this identity taking as its elements all the sets of at least 4 wires, on which we have a phase gadget with phase $\frac{\pi}{4}$. □

This means that the number of all different, nontrivial composite spider nest identities on n wires grows double exponentially, and to be precise, it is equal to

$$2^{\left(2^n - \sum_{k=0}^{3} \binom{n}{k}\right)} - 1.$$

Let's note that already for $n = 6$ this number already is equal $2^{22} - 1$.

5.2.2.2 Results and discussion

As the number of all possible composite spider nest identities grows so rapidly, direct use of all possible composite spider nest identities on at least 6 wires is unrealistic. This motivates the search for an optimal subset of these identities to be used. Unfortunately, we were not able to choose the bigger identities in any effective way. We tested adding the bigger composite spider nest identities with lower T-count, higher T-count, choosing them at random and combining all these approaches, but none of them achieved better than PHAGE5 procedure results.

5.2.2.3 Possible explanations and need for other approaches

The spider nest identities of bigger sizes, not only, in their base form, have significantly higher T-count, but also act on more wires, which makes them significantly more challenging to use effectively. It is caused by the fact that there are significantly more possible choices of wires on which we can apply them. Moreover, if we decide to use the ones with

a higher T-count, we will not only have noticeably more choices of different sets of wires but also each choice will need to contain more phase gadgets influencing the T-count. This suggests a need for a different, more circuit-oriented approach to utilizing them in T-count reduction problems.

Chapter 6

Modifications of the decomposition phase

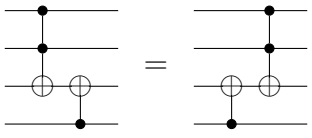
In the previous chapters, we focused on modifications of circuits composed of the phase gadgets assuming the decomposition to be fixed and given. In this chapter, we consider the usage of different prepossessing steps, motivated by the procedure presented in [25], which describes the commutation of generalized $Toffoli$ gates. We present alternative proofs based on the spider nest identities, for special cases of the commutation steps, that are necessary for transforming the family of Galois Field multipliers circuits. We use it then as a prepossessing step, to move the $CNOT$ gates away from the main part of the circuit [25], for the algorithm from [15] in order to reach a new state of the art results for optimization of the $GF(2^m)$ multipliers circuits family. Finally, we also discuss problems that arise from attempts to generalize these results for the other circuits.

6.1 Commutation rules for the $GF(2^m)$ multipliers

The $GF(2^m)$ circuits can be expressed using only $Toffoli$ and $CNOT$ gates [11], which allows us to move the $CNOT$ gates away from the middle of the circuit, as it is presented in [25]. It is done using the commutation rules which we present here along with proofs based on the phase gadgets and spider nest identities (lemmas 8 to 10). For the sake of completeness, we should note that [11, 25] present a more complete set of rules – for $Toffoli$ gates with controls of arbitrary size – which we do not consider here.

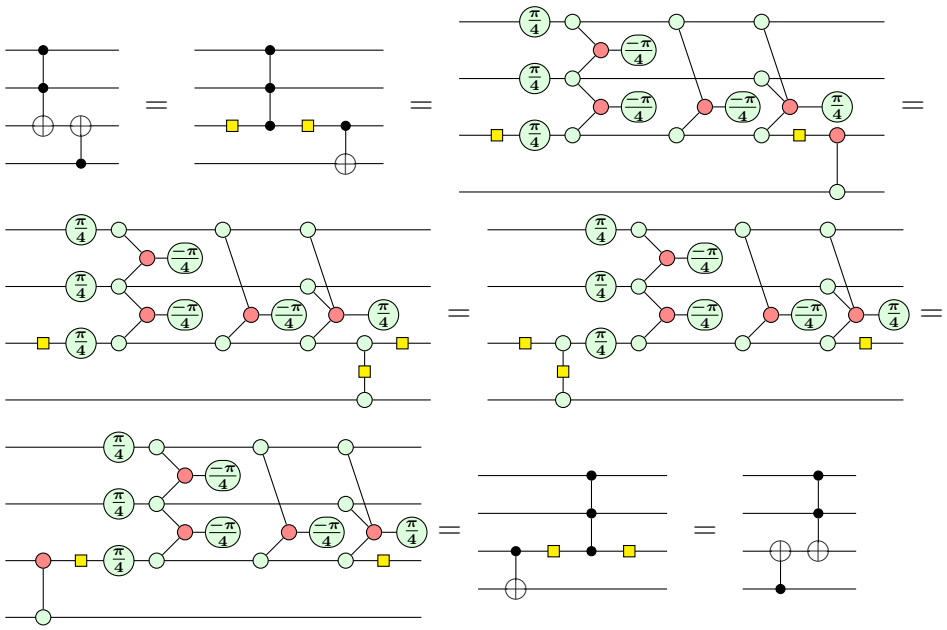
First, we consider the most basic case.

Lemma 8. *If the targets of a $Toffoli$ gate and a $CNOT$ gate are the same wire, we can commute these gates without any problems.*



The proof of this property is straightforward.

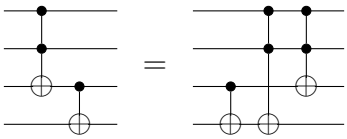
Proof. First, we decompose the $Toffoli$ gate, then we merge the $CNOT$ gate with the $Hadamard$ gate, we commute green nodes, and again move the $Hadamard$ gate to finally compose the gates back.



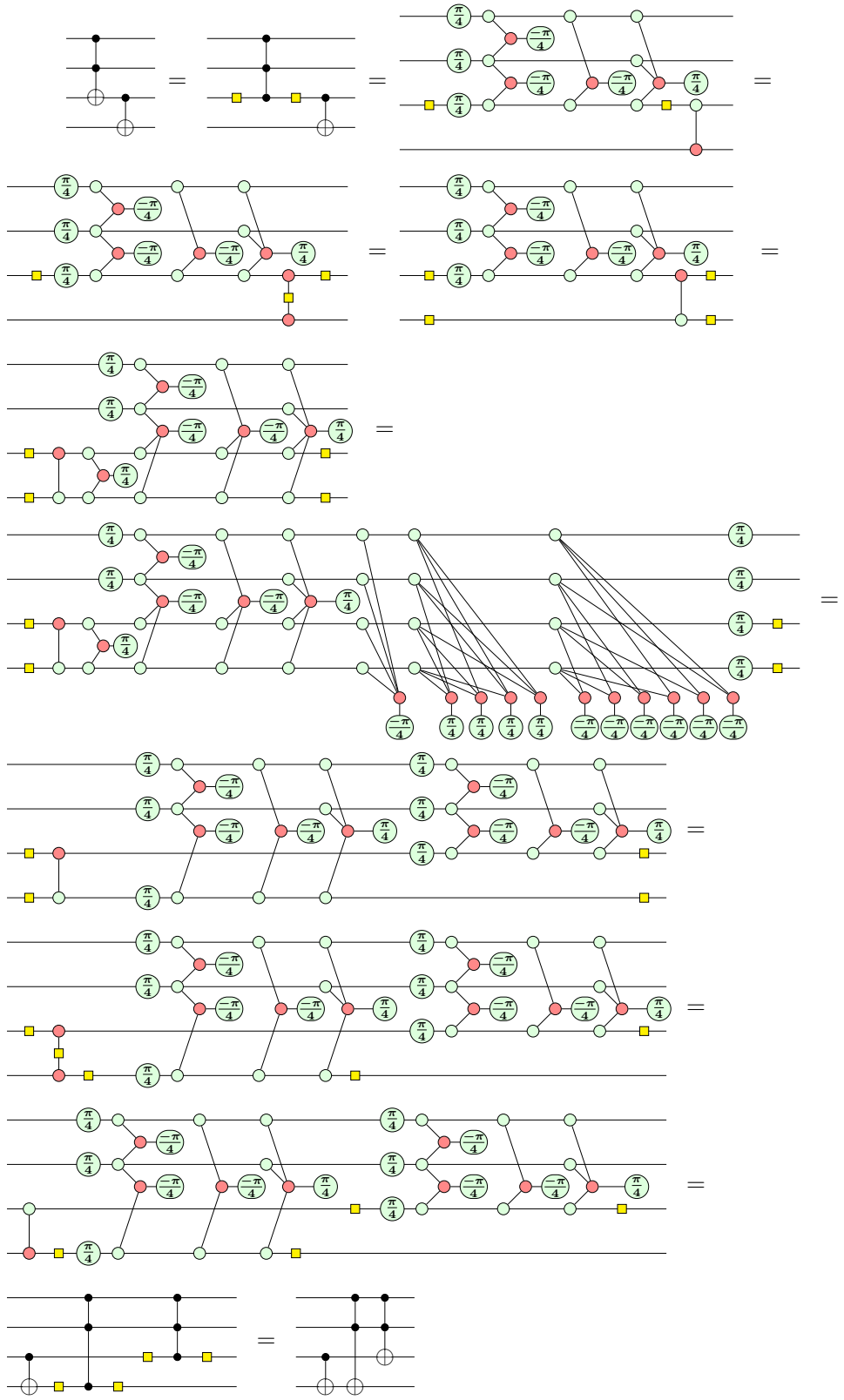
□

The more interesting case is the one of target gate of the $Toffoli$ gate on the same wire as the control of the $CNOT$ gate.

Lemma 9. *In that case commutation of the $CNOT$ will result in an emergence of a new $Toffoli$ gate.*



Proof. In order to prove this equality we start in a similar way as in the proof of lemma 8, then we need to commute the $CNOT$ gate through multiple phase gadgets, as we described it in section 2.5.4.2. Now, in order to remove the phase gadget on 4 wires, we use the spider nest identity (more precisely, we use its adjoint, as the adjoint of identity is an identity). Finally, we commute two $Hadamard$ gates through the $CNOT$ gate and compose all the gates back.

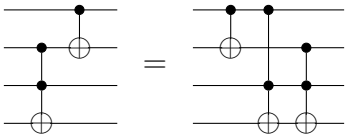


□

Analogical transformation happens when the target of a $CNOT$ gate is on the same

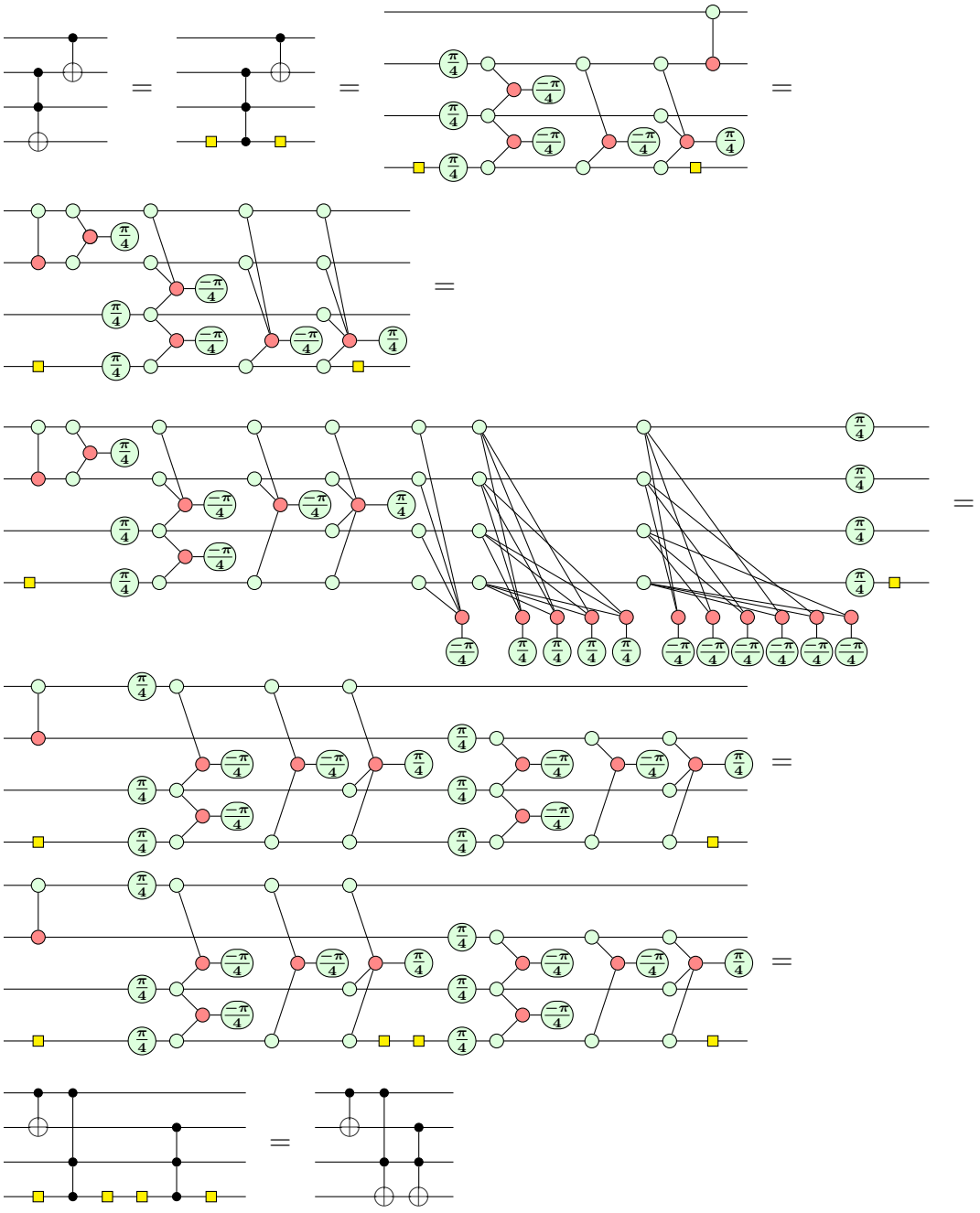
wire as one of the controls of a $Toffoli$ gate.

Lemma 10.



The proof of this equality is mostly analogical to the one of lemma 9.

Proof.



□

Circuit	initial $\#T$	[15] $\#T$	[25] $\#T$	Ours $\#T$
$GF(2^4)$ Mult	112	53	62	50
$GF(2^5)$ Mult	175	88	97	83
$GF(2^6)$ Mult	252	128	131	109
$GF(2^7)$ Mult	343	167	183	157
$GF(2^8)$ Mult	448	229	263	223
$GF(2^9)$ Mult	567	306	299	262
$GF(2^{10})$ Mult	700	357	361	314
$GF(2^{16})$ Mult	1792	972	1038	963

Table 6.1: Comparison of our T-count reduction results and those from [25, 15]. All results were verified using [3].

6.2 Preprocessing step

Using these commutation rules, we can, before processing the circuit in any other way, move all $CNOT$ gates away from the main part of the circuit. To be precise, for each $CNOT$ gate we check if we should commute it to the left, or to the right by counting the number of $Toffoli$ gates it interacts with during the commutation. It is a heuristics analogical to the one used in section 2.5.4.2. The choice of such a metric is motivated by lemmas 9 and 10 – each commutation of a $CNOT$ and $Toffoli$ gates produces a single new $Toffoli$ gate.

6.2.1 Differences and advantages

In the procedure from [15], the $Toffili$ gates will be decomposed into CCZ and $Hadamard$ gates, then we will attempt to move the $Hadamard$ gates, and only after it is done, we will attempt to move the $CNOT$ gates. This might lead, and in the case of all considered circuits leads, to a different structure of the main part of the circuit, and to a noticeable increase in its complexity. This motivates the question about the effectiveness of the algorithm with the new prepossessing steps.

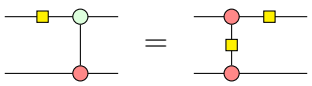
6.3 Results

In table 6.1, we present results of T-count reduction of the Galois field multiplier circuits. We compare against results from [15, 25]. A more detailed comparison is presented in chapter 7.

6.4 Problems with generalization to other circuits

As this approach gives us a significant improvement of the T-count, an obvious next step would be to turn it into a general procedure that could be applied for all kinds of circuits.

Unfortunately, it turns out to be a highly complicated problem, as the preprocessing phase heavily relies on the structure of the circuit – the fact that it consists only of $Toffoli$ and $CNOT$ gates. The main difficulty of the problem lies in the fact that commutation of a $CNOT$ gate through a $Hadamrad$ gate that is on the same wire as the control wire of $CNOT$ gate, leads to the emergence of a new quantum gate (which is a gate equivalent to CZ gate in the dual base). In terms of the ZX-calculus, we can express this commutation as follows.



One might try to consider various extensions of the set of considered quantum gates. Unfortunately, it is important to consider the limitations of our approach. We try to represent the circuit as a sequential composition of three parts where the first and the last parts are required to be composed of Clifford operations, and the main part should consist of only phase gadgets. Because of that, on a fixed set of n wires, we can represent only a finite number of circuits – each of the initial and final parts consist of (possibly classically controlled) Clifford operations, and the main part consists of phase gadgets, of which we can also have only a finite number. To be more precise, as using the spider nest identities we can decompose all phase gadgets of size bigger than 3, we can represent at most $2^{\left(\binom{n}{3}+\binom{n}{2}+\binom{n}{1}\right)}$ different main circuits up to a Clifford transformation.

Connecting it with the fact that the set of Clifford+T gates is capable of approximating arbitrary unitary transformations, without extending the size of a circuit (what we only do when we process the $Hadamard$ gates), we are not able to represent most of the circuits.

Chapter 7

Summary

In this dissertation, we covered the theoretical foundations of the circuit optimization using the spider nest identities and provided abstract definitions of the considered problems. We introduced dense spider nest identities as an alternative way of looking at the spider nest identities and used the spider nest identities to prove their existence. Then we looked closer at the performance issues of the novel method from [15] and proposed a way to overcome them in a systematic way. We also analyzed other, non-successful approaches to improve the performance of this algorithm. Finally, we used the method from [25, 19], for commuting the n-qubit $Toffoli$ gates, to move $CNOT$ gates away from the middle of the circuit as a preprocessing step instead of doing it at the later stage of the algorithm, which resulted in improved T-count.

Table 7.1 presents a detailed comparison of our results, best-known results, and those from [15].

7.1 Time of execution

It is important to note that the methods that we considered in this dissertation were not only meant to improve the algorithm presented in [15], but do so without a significant increase of the execution time. All presented modifications, as well as the results satisfy the requirements, keeping the complexity as well as the approximate execution time of this method. That means that we can obtain the notable T-counts reduction *in run-times which are typically less than the time required to make a fresh cup of coffee* [15].

Circuit	init.		best prev.		[15]	Ours		
						dec. ch. 5	inc. ch. 5	chapter 6
	n	$\#T$	$\#T$	source	$\#T$	$\#T$	$\#T$	$\#T$
Adder_8	24	399	**167**	[23]	176	209	208	
Barenco Tof_3	5	28	**13**	[16]	**13**	14	**13**	
Barenco Tof_4	7	56	**24**	[18]	25	**24**	25	
Barenco Tof_5	9	84	**34**	[18]	37	36	37	
Barenco Tof_{10}	19	224	**84**	[18]	97	98	98	
CSLA MUX_3	15	70	**40**	[18]	44	45	50	
CSLA MUX_9	30	196	**74**	[18]	84	84	84	
GF(2^4) Mult	12	112	**47**	[16]	53	56	57	50
GF(2^5) Mult	15	175	84	[16]	88	88	86	**83**
GF(2^6) Mult	18	252	118	[16]	128	130	128	**109**
GF(2^7) Mult	21	343	167	[15]	167	173	184	**157**
GF(2^8) Mult	24	448	214	[23]	229	236	232	**223**
GF(2^9) Mult	27	567	295	[18]	306	281	277	**262**
GF(2^{10}) Mult	30	700	351	[18]	357	355	353	**314**
GF(2^{16}) Mult	48	1792	**922**	[18]	972	973	966	963
Mod5_4	5	28	**7**	[16]	**7**	**7**	**7**	
Mod Adder_{1024}	28	1995	**978**	[18]	1010	1007	1009	
Mod Mult_{55}	9	49	**18**	[16]	19	20	20	
Mod Red_{21}	11	119	**55**	[16]	65	61	63	
QCLA Adder_{10}	36	238	147	[15]	147	152	**143**	
QCLA Com_7	24	203	81	[18]	84	**79**	83	
QCLA Mod_7	26	413	**216**	[23]	233	229	231	
RC Adder_6	14	77	**37**	[16]	38	41	38	
NC Toff_3	5	21	**13**	[16]	**13**	**13**	**13**	
NC Toff_4	7	35	**19**	[16]	**19**	**19**	**19**	
NC Toff_5	9	49	**25**	[18]	26	**25**	26	
NC Toff_{10}	19	119	**56**	[16]	60	59	60	
VBE Adder_3	10	70	**20**	[18]	**20**	**20**	**20**	

Table 7.1: Comparison of T-count achieved by our methods, the best results from [6, 18, 23, 15, 16], and those from [15]. "Init. n" denotes the number of input qubits of the circuit, "init. $\#T$" denotes the initial T-count of the circuit, column "best prev." contains the best previous results and indicates where were they first achieved. The next column contains the results from [15]. The three last columns (the ones labelled "Ours") contain the results that we achieved using methods presented in this dissertation. "Dec. ch. 5" refers to the decreasing order described in chapter 5, and "inc. ch. 5" to the increasing one. The final column contains the results of the method described in chapter 6. Our results, which strictly improved the ones from [15], are marked in colour. Additionally, for each circuit, we present in bold the best-achieved result. All results, except those for Mod Adder_{1024} and QCLA Mod_7 circuits, were verified using [3].

Bibliography

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5), Nov 2004.

[2] M. Amy, D. Maslov, M. Mosca, and M. Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, Jun 2013.

[3] Matthew Amy. Feynman, github, `https://github.com/meamy/feynman`, 2019.

[4] Matthew Amy. Towards large-scale functional verification of universal quantum circuits. *Electronic Proceedings in Theoretical Computer Science*, 287:1–21, Jan 2019.

[5] Matthew Amy, Jianxin Chen, and Neil J. Ross. A finite presentation of cnot-dihedral operators. *Electronic Proceedings in Theoretical Computer Science*, 266:84–97, Feb 2018.

[6] Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time t-depth optimization of clifford+t circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, Oct 2014.

[7] Matthew Amy and Michele Mosca. T-count optimization and reed–muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, Aug 2019.

[8] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, Nov 1995.

[9] Xiaoning Bian. stomp-code, github, `https://github.com/onestruggler/stomp-code`, 2020.

[10] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, Sep 2017.

[11] Donny Cheung, Dmitri Maslov, Jimson Mathew, and Dhiraj K. Pradhan. On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography. In Yasuhito Kawano and Michele Mosca, editors, *Theory of Quantum Computation, Communication, and Cryptography*, pages 96–104, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, Apr 2011.

[13] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, Apr 2011.

[14] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[15] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and effective techniques for t-count reduction via spider nest identities, 2020.

[16] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Techniques to reduce pi/4-parity-phase circuits, motivated by the zx calculus. *Electronic Proceedings in Theoretical Computer Science*, 318:131–149, May 2020.

[17] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the t-count, 2013.

[18] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, sep 2018.

[19] Kazuo Iwama, Yahiko Kambayashi, and Shigeru Yamashita. Transformation rules for designing cnot-based quantum circuits. In *Proceedings of the 39th Annual Design Automation Conference*, DAC '02, page 419–424, New York, NY, USA, 2002. Association for Computing Machinery.

[20] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the zx-calculus for clifford+t quantum mechanics, 2018.

[21] Aleks Kissinger. Tikzit: a super simple gui editor for graphs and string diagrams, `https://tikzit.github.io/`.

[22] Aleks Kissinger and John van de Wetering. PyZX: Large Scale Automated Diagrammatic Reasoning. 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 229–241. Open Publishing Association, 2020.

[23] Aleks Kissinger and John van de Wetering. Reducing the number of non-clifford gates in quantum circuits. *Physical Review A*, 102(2), Aug 2020.

[24] Olivia Di Matteo and Michele Mosca. Parallelizing quantum circuit synthesis. *Quantum Science and Technology*, 1(1):015003, Mar 2016.

[25] Anthony Munson, Bob Coecke, and Quanlong Wang. And-gates in zx-calculus: spider nest identities and qbc-completeness, 2020.

[26] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. optimizer, github, `https://github.com/njross/optimizer`, 2018.

[27] Kang Feng Ng and Quanlong Wang. A universal completion of the zx-calculus, 2017.

[28] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[29] Tommaso Toffoli. Reversible computing. *Technical Report MIT/LCS/TM-151,MIT LCS, Feb.*, 1980.

[30] Quanlong Wang. An algebraic axiomatisation of zx-calculus, 2021.