# Building the tree

**Abstract**

We present the precise mathematical algorithm that allows us to calculate how the complexity of a loop circuit grows when adding a specific vertex to a tree-shaped graph state. We then proceed to discuss how this complexity function grows when building the tree in a generic DFS order. We thus show that an optimal DFS order can be found. Furthermore we present a series of elementary bounds for the complexity of the circuit that apply for evey DFS order.

## Contents

**1 Preliminaries: characterizing the circuit** **2**

# 1 Preliminaries: characterizing the circuit

Let's start by some definitions.

**Tree.** A tree is a graph with no cycles. We will use a stronger definition: a tree is a graph with no cycles equipped with a special vertex called *head* of the the tree. Notice that the presence of the head induces a hierararchy: if we start building the tree from the head, when we add the vertex $v$, we know where to fuse it (to its parent $p(v)$).

**Order on a tree.** We define an order $O$ on a tree $T$ an array of vertices $[h, v_1, v_2, \dots]$ where $h$ is the head of the tree. This list represents the order in which we want to add the vertices to the tree: thus the parent $p(v)$ of a vertex $v$ must be present in the list before $v$.

**Depth of a photonic line.** Given a photonic line, there will be a last optical element on this line (remember that in our setup every optical element is a $SU(2)$ matrix that occupies two photonic lines). This optical element will belong to a specific outer loop numbered $n$, we define the *depth* of the photonic line as this number $n$.

**Complexity of a circuit.** The *complexity* of a circuit is the maximal depth of its photonic lines.

This series of definitions allow us to introduce our main character, the *complexity function* $C(T, O)$:

**The complexity function.** Given a tree and an order on this tree, this function $C(T, O)$ tells us the number of outer loops needed to implement the circuit.