Quantum Information and Computation, Vol. 15, No. 9&10 (2015) 0853–0884

OPTIMISING THE INFORMATION FLOW OF ONE-WAY QUANTUM COMPUTATIONS

EINAR PIUS and ELHAM KASHEFI
School of Informatics, University of Edinburgh, 10 Crichton Street Edinburgh EH8 9AB, UK

RAPHAEL DIAS DA SILVA
Instituto de Física, Universidade Federal Fluminense, Av. Gal. Milton Tavares de Souza s/n Gragoatá, Niterói, RJ, 24210–340, Brazil

Received May 20, 2013
Revised February 18, 2015

In the one-way quantum computing model, the information processing is driven by measurements performed on an entangled state, called the resource state. In order to achieve parallelism, one aims to increase the number of simultaneous measurements, such that the effects of decoherence on the resource state are minimised due to the reduction of the time required to run the computation. At the heart of this question is the notion of quantum information flow, which specifies the dependency relations between the measurements in the computations.

There exist two well-known techniques for reducing the time required to perform a one-way quantum computation without changing its semantics. The first one, called signal-shifting, transforms the flow of a graph (representing the resource state) within the measurement calculus formalism, whereas the second one, namely finding the maximally-delayed generalised flow, explores the geometry of the graph to increase the number of operations that can be performed simultaneously. In this paper, we show for the first time how these two techniques relate to each other.

We prove that the application of the signal-shifting rules to a measurement pattern with flow results in a generalised flow for the pattern. Then, we prove that in the particular case when the input size equals the output size, the gflow obtained using signal-shifting has the lowest possible depth. As a side result, we construct an $O(n^3)$-algorithm for finding maximally delayed gflows on graphs with flow. For those graphs, our algorithm is more efficient than the best previously known algorithm for the same task, which takes $O(n^4)$ operations to complete.

Keywords: measurement-based computing, parallelism, flow and determinism
Communicated by: R Jozsa & A Harrow

1 Introduction

After the emergence of the measurement based quantum computing (MBQC) model it was clear that not every operator describable in this models is deterministic, *i.e.* implements the same unitary operation in each of its computational branches. To describe the class of computations that are deterministic, the concepts of flow [1] and generalised flow (gflow) [2] were developed. Surprisingly, the aforementioned research unveiled that the suitability of a quantum resource for deterministic MBQC has a concrete graph theoretical interpretation.

The concepts of flow and gflow since then have been explored in many new directions as we briefly review below.

A major utilisation of gflow is as a tool for parallelisation. In particular, if it is known that a computation in MBQC is done according to a gflow, it is possible to find the maximally delayed gflow [3] — the gflow with the smallest computational depth. This is a complementary parallelisation method to signal shifting [4, 5] — a process which looks at the pattern representation of the computation instead of the structure of the underlying resource. For a long time it was not known how exactly the outcomes of these two methods relate to each other. In this paper we finally prove that if we consider computations on quantum resources with flow, these methods produce results with the same computational depth.

Parallelisation methods of MBQC can be used to optimise the depth of computations of quantum circuits via back and forth translation to the MBQC model [5]. This is especially interesting since it has been proven that the quantum depth of computations in the MBQC model is equal to the depth of quantum circuits with the addition of unbounded fan-out [6]. The translation of a parallelised MBQC computation to the circuit model introduces a considerable amount of extra qubits [5]. An algorithm for translating MBQC with flow to circuits without adding ancillae exists since its discovery [1] but similar algorithms for gflow have emerged only recently [7, 8]. Notably, the techniques introduced in this work have been also used to provide a new optimised translation method from MBQC to quantum circuit for a family of patterns with gflow [9].

In other context, flow and gflow have been used as tools in entanglement analysis. In particular it was shown how the algorithm for finding gflow leads to a more efficient method for establishing a bound on the entanglement of the graph states. This exposes a new approach to the study of the classical simulatability of resource states in MBQC [10]. As flow and gflow have such a wide applicability in the research of MBQC, efficient algorithms for finding them have been studied in the literature leading to different techniques based on network flow and linear programming [11, 3]. Our theoretical work on linking signal shifting and gflow allow us to present a new approach to obtain an algorithm for finding maximally delayed gflow of a given open graph state with flow. This new $O(n^3)$ algorithm has an improved runtime compared to the previously best maximally delayed gflow finding [3] and signal-shifting algorithm [4], whose number of computational steps are $O(n^4)$ and $O(n^6)$ correspondingly.

Flow has been also utilised in simplifying the design and analysis of various cryptographic protocols [12, 13] and the link between Adiabatic Quantum Computing [14] and One-clean qubit model with MBQC [15]. Our main result in linking signal shifting with gflow will indicate that gflow (for those graphs with the same input and output size) will not fundamentally change these findings. In other words our work confirms that the key difference of the gflow construction is in its inherent parallelisaiton structure that could be now obtained through the simple rewriting schemes of signal shifting.

2 Preliminaries

In 1999 Chuang and Gottesman showed how quantum teleportation could be used to implement arbitrary quantum gates [16]. This approach was further developed by other researchers [17, 18, 19, 20], enabling one in principle to perform arbitrary computations given a few primitives: preparation of maximally entangled systems of fixed, small dimension;

multi-qubit measurements on arbitrary set of qubits; and the possibility of adapting the measurement bases depending on earlier measurement outcomes.

These models draw on measurements to implement the dynamics of a computation, and as such are called measurement-based quantum computation (MBQC) models. For an overview see the paper by Jozsa [21]. An MBQC model using only single qubit measurements was proposed by Raussendorf and Briegel in 2001, which became known as the one-way quantum computing (1WQC) model [22]. Although two-dimensional cluster states can be used as a resource for universal quantum computation in the one-way model [23], arbitrary graph states may, or may not, serve the same purpose; investigating which kinds of entangled states are useful resources for MBQC is an active area of research [24, 25, 26, 27, 28]. We review the basic ideas behind the measurement-based quantum computation, with special attention to its description in terms of the formal language known as Measurement Calculus [4], and the flow theorems [1, 2].

2.1 Computation in MBQC

A *measurement pattern*, or simply a pattern, is defined by a choice of a set of working qubits V, a subset of input qubits (I), another subset of output qubits (O), and a finite sequence of commands acting on qubits in V.

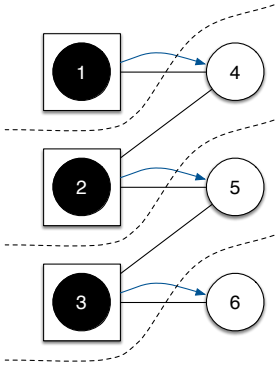
Definition 1 (open graph) *An open graph is a triplet (G, I, O), where $G = (V, E)$ is an undirected graph, and $I, O \subseteq V$ are respectively called input and output vertices.*

There are four types of command, the first is the qubit initialisation command N_i that prepares qubit i in the state $|+\rangle$. The input qubits are already given as a prepared state. The entangling command E_{ij} is defined to be controlled-Z_{ij} operator between qubits i and j. The single-qubit measurement command M_i^θ corresponds to a measurement of qubit i in the basis $|\pm_\theta\rangle \equiv \frac{1}{\sqrt{2}}(|0\rangle \pm e^{i\theta}|1\rangle)$, with outcome $s_i = 0$ associated with $|+_\theta\rangle$, and outcome 1 with $|-_\theta\rangle$. The measurement outcomes are usually referred as *signals*. Finally, the corrections may be of two types, either Pauli X or Pauli Z, and they may depend on any prior measurement results, denoted by $s = \oplus_{j \in J \subset V} s_j$ ($s_j = 0$ or 1 and the summation is done modulo two). This dependency can be summarised as correction commands: X_i^s and Z_i^s denoting Pauli X and Z corrections on qubit i which must be applied only when the parity of the measurement outcomes on qubits $j \in J \subset V$ equals one (as $Z^0 = X^0 = I$). A characteristic of the MBQC model is that the choice of measurement bases may depend on earlier measurement outcomes. These dependent measurements can be conveniently written as ${}_t[M_i^\theta]^s$, where

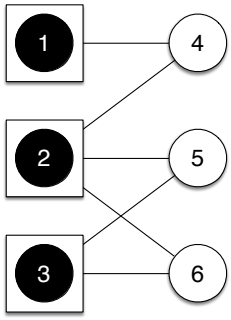
$$ {}_t[M_i^\theta]^s \equiv M_i^\theta X_i^s Z_i^t = M_i^{(-1)^s\theta + t\pi}, \tag{1} $$

where it is understood that the operations are performed in the order from right to left in the sequence. The left (t) and right (s) dependencies of the measurement M_i are called its Z and X *dependencies*, respectively.

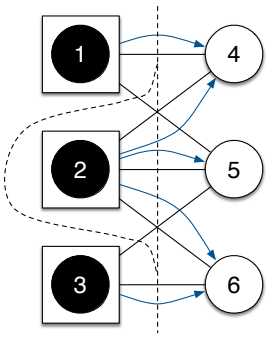
A pattern is runnable, that is, corresponds to a physically sound sequence of operations, if it satisfies the following requirements: (R0) no command depends on outcomes not yet measured; (R1) no command acts on a qubit already measured or not yet prepared, with the obvious exception of the preparation commands; (R2) a qubit undergoes measurement (preparation) if and only if it is not an output (input) qubit.



(a) An open graph with flow and gflow.



(b) An open graph with no flow and no gflow.



(c) An open graph with gflow and no flow.

Fig. 2. Examples of open graphs with flow (a), gflow (b) and without either (c). Blue lines represent the flow/gflow functions and dashed lines group together measurements that can be performed simultaneously according to the flow/gflow partial order.

2.2 Flow in MBQC

Due to the probabilistic nature of quantum measurement, not every measurement pattern implements a *deterministic* computation — a completely positive, trace-preserving (cptp) map that sends pure states to pure states. We will refer to the collection of possible measurement outcomes as a *branch* of the computation. In this paper, we consider deterministic patterns which satisfies three conditions: (1) the probability of obtaining each branch is the same, called *strong determinism*; (2) for any measurement angle we have determinism, called *uniform determinism*; and (3) which are deterministic after each single measurement, called *stepwise determinism*. We will call those patterns simply *deterministic patterns*. As explained earlier, measurement patterns can act on resource states called open graphs. Similarly to measurement patterns, which not always implement deterministic computations, there exist open graphs which cannot be used for deterministic computation. Identification and characterisation of the graphs that could be used has been done in [1, 2, 29]. Following sections summarise the results most relevant to this work.

Sufficient conditions (known as the *flow*) for open graphs which can be used for deterministic computation were presented in [1]. Flow is the basis on which we construct the *signal shifted flow* in Section 3 and is a central concept in this work. In what follows, we denote non-input vertices as I^C (complement of I in the graph) and non-output vertices as O^C (complement of O in the graph).

Definition 2 (Flow [1]) *We say that an open graph (G, I, O) has flow iff there exists a map $f : O^C \to I^C$ and a strict partial order $\prec_f$ over all vertices in the graph such that for all $i \in O^C$*

- *(F1) $i \prec_f f(i)$;*
- *(F2) if $j \in N(f(i))$, then $j = i$ or $i \prec_f j$, where $N(v)$ is the neighbourhood of v;*
- *(F3) $i \in N(f(i))$.*

An example of a flow is given in Figure 1a. Given a flow $(f, \prec_f)$ of an open graph (G, I, O) it is possible to write a deterministic measurement pattern on the graph [1]:

$$P = \prod_{i \in O^C}^{\prec_f} \left(X_{f(i)}^{s_i} Z_{N(f(i)) \setminus \{i\}}^{s_i} M_i^{\alpha_i} \right) E_G N_{I^C}. \tag{2}$$

Note that since this pattern represents a uniformly deterministic computation, it implements a unitary operator irrespective of the measurement angles α_i. From Equation 2 we see that a Z-correction on a vertex j depending on the measurement outcome of another vertex i appears only if j is a neighbour of $f(i)$. This is formally stated in the next corollary, to which we will refer in several places.

Corollary 1 *If (G, I, O) is an open graph with a flow $(f, \prec_f)$, then there exists a Z-correction from vertex i to another vertex j iff $j \in N(f(i)) \setminus \{i\}$.*

Another trivial but useful property of flow is stated in the next lemma.

Lemma 1 *Let $(f, \prec_f)$ be a flow on an open graph (G, I, O). The function f is an injective function, i.e. for every $i \in O^C$, $f(i)$ is unique.*

Proof. Let us assume that for some $i \in O^C$, $f(i)$ is not unique, *i.e.* there exists $j \in O^C$ such that $i \neq j$ but $f(i) = f(j)$. Then according to the flow definition:

$$j \in N(f(j)) = N(f(i)) \Rightarrow i \prec_f j,$$
$$i \in N(f(i)) = N(f(j)) \Rightarrow j \prec_f i,$$

and we arrive at a contradiction because $i \prec_f j$ and $j \prec_f i$ cannot be true at the same time. Hence $f(i)$ has to be unique $\square$.

2.3 Gflow in MBQC

Flow provides only a sufficient condition for determinism but one can generalise it by allowing correcting sets with more than one element to obtain a condition that is both necessary and sufficient. In what follows we define $Odd(K) = \{k\,, |N_G(k) \cap K| = 1 \mod 2\}$ to be the set of vertices where each element is connected with the set K by an odd number of edges.

Definition 3 (Generalised flow [2]) *We say (G, I, O) has generalised flow if there exists a map $g : O^C \to 2^{I^C}$ (the set of all subsets of non-input qubits) and a partial order $\prec_g$ over all vertices in the graph such that for all $i \in O^C$,*

- *(G1) if $j \in g(i)$ then $i \prec_g j$;*
- *(G2) if $j \in Odd(g(i))$ then $j = i$ or $i \prec_g j$;*
- *(G3) $i \in Odd(g(i))$.*

The set $g(i)$ is often referred to as the *correcting set* for qubit i. Flow is a special case of gflow, where $g(i)$ contains exactly one element. An example of a gflow is given in Figure 1c. Interestingly, adding one edge to the flow in Figure 1a as done in Figure 1b can remove both the flow and gflow from the open graph, but adding two edges as in Figure 1c will remove only the flow from the graph. Such graphical representation of underlying entanglement and their

link to flow and gflow is fully explained in [10]. Similar to the flow scenario, a deterministic pattern P for an open graph (G, I, O) can be derived given a gflow $(g, \prec_g)$ on the graph:

$$P = \prod_{i \in O^C}^{\prec_g} \left(X_{g(i)}^{s_i} Z_{Odd(s(i))}^{g_i} M_i^{\alpha_i} \right) E_G N_{I^C}. \tag{3}$$

The gflow partial order leads to an arrangement of the vertices into layers (see below), in which all the corresponding measurements can be performed simultaneously. The number of layers corresponds to the number of parallel steps in which a computation could be finished, known as the *depth* of the pattern.

Definition 4 (Depth of a gflow [3]) *For a given open graph (G, I, O) and a gflow $(g, \prec_g)$ of (G, I, O), let*

$$V_k^{\prec_g} = \begin{cases} \max_{\prec_g}(V(G)) & \text{if } k = 0 \\ \max_{\prec_g}(V(G) \setminus (\cup_{i<k} V_i^{\prec_g})) & \text{if } k > 0 \end{cases}$$

where $\max(X)_{\prec_g} = \{u \in X \text{ s.t. } \forall v \in X, \neg(u \prec_g v)\}$ is the set of maximal elements of X according to $\prec_g$. The depth *$d^{\prec_g}$ of the gflow is the smallest d such that $V_{d+1}^{\prec_g} = \emptyset$, $(V_k)_{k=0 \ldots d^{\prec_g}}$ is a partition of $V(G)$ into $d^{\prec_g} + 1$ layers.*

We define the layering function of a gflow based on the above distribution of vertices into layers.

Definition 5 (Layering function) *Given a gflow $(g, \prec_g)$ on an open graph (G, I, O) we define its* layering function *$L_g : V(G) \to \mathbb{N}$ to be the natural number k such that $L(i) = k$ iff $i \in V_k^{\prec_g}$.*

There is another useful way to understand the depth of a gflow by representing it as a directed graph on top of the entanglment graph. The longest path from inputs to outputs over those directed edges corresponds to the depth of the gflow. In [3] it was shown, that a special type of gflow, called a maximally delayed gflow, has minimal depth.

Definition 6 (Maximally delayed gflow [3]) *For a given open graph (G, I, O) and two given gflows $(g, \prec_g)$ and $(g', \prec_{g'})$ of (G, I, O), $(g, \prec_g)$ is more delayed than $(g', \prec_{g'})$ if $\forall k$, $|\cup_{i=0 \ldots k} V_i^{\prec_g}| \geq |\cup_{i=0 \ldots k} V_i^{\prec_{g'}}|$ and there exists a k such that the inequality is strict. A gflow $(g, \prec_g)$ is* maximally delayed *if there exists no gflow of the same graph that is more delayed.*

Note that in [3] it was proven that the layering of the vertices imposed by maximally delayed gflows is always unique, however the gflow itself might not be unique. This is an important property, which together with the following lemmas is exploited later in linking gflow to other known structures of MBQC.

Lemma 2 (Lemma 1 from [3]) *If $(g, \prec)$ is a maximally delayed gflow of (G, I, O) then $V_0^{\prec} = O$.*

Lemma 3 (Lemma 2 from [3]) *If $(g, \prec)$ is a maximally delayed gflow of (G, I, O) then $(\tilde{g}, \prec_{\tilde{g}})$ is a maximally delayed gflow of $(G, I, O \cup V_1^{\prec})$ where $\tilde{g}$ is the restriction of g to $V(G) \setminus (V_1^{\prec} \cup V_0^{\prec})$ and $\prec_{\tilde{g}} = \prec \setminus V_1^{\prec} \times V_0^{\prec}$.*

A simpler characterisation of stepwise strong determinism called *focused gflow* was introduced in [29].

Definition 7 (Focused gflow [29]) *$g : O^C \to 2^{I^C}$ is a focused gflow of (G, I, O) if*

- *(FG1) g is extensive i.e. the transitive closure of the relation $\{(i, j)$ s.t. $j \in g(i)\}$ is a partial order over $V(G)$;*
- *(FG2) $\forall i \in O^C$, $Odd(g(i)) \cap O^C = \{i\}$.*

Both flow and focused gflow have been shown to be unique for open graphs where input size is equal to output size [11, 29]. One way of explaining the uniqueness of focused gflow is through its relation with the flow. Namely, signal shifting a flow results in a focused gflow as will be shown in Section 3.

3 Signal Shifting

The parallel power of MBQC is proven to be equivalent to quantum circuits augmented with unbounded fanout [6]. This motivates us to use MBQC as an automated tool for circuit parallelisation as it was first presented in [5]. Another way to obtain parallel MBQC structure is to extract the entanglement graph of the pattern and obtain the maximally delayed gflow of the graph [3]. Then one performs the required corrections according to this structure. Our main result is to show the equivalence between these two seemingly very different techniques for the patterns obtained from a quantum circuit, that is those with flow. More precisely we show how the effect of performing signal shifting optimisation (that is the core idea in [5]) result in a maximally delayed gflow. This structural link sheds further light on the complicated structure of maximally delayed gflow and permits us to find a new efficient algorithm for finding it for the large class of patterns obtained from a circuit. These concepts are formalised below.

We proceed with reviewing the rules for signal shifting defined in [4] that will essentially permit one to delay the classical dependencies between commands to the latest possible stage in the computation. By classical dependency between commands we are referring to action of the Z correction that only effects the classical outcome of a measurement.

$$ {}_t[M_i^\alpha]^s \Rightarrow S_i^t \, [M_i^\alpha]^s \tag{4} $$

$$ {}_t[M_j^\alpha]^s \; S_i^{t'} \Rightarrow S_i^{t'} \; {}_{t[(t'+s_i)/s_i]}[M_j^\alpha]^s \tag{5} $$

$$ X_j^s \; S_i^t \Rightarrow S_i^t \; X_j^{s[(t+s_i)/s_i]} \tag{6} $$

$$ Z_j^s \; S_i^t \Rightarrow S_i^t \; Z_j^{s[(t+s_i)/s_i]} \tag{7} $$

where S_i^t is the signal shifting command (adding t to s_i) and $s[t/s_i]$ denotes the substitution of s_i with t in s. Signal shifting can be utilised to parallelise MBQC patterns and quantum circuits [5]. The rest of this section is focused on various structural properties of signal shifting.

As can be seen from the above rules, signal shifting rewrites the X- and Z-corrections of a measurement pattern in a well defined manner. An example illustrating that is given in Figure 4. In particular, it will move all the Z-corrections to the end of the pattern, thereby introducing new X-corrections when Rule 6 is applied. It is proven in [5] that signal shifting will never increase the depth of an MBQC pattern, although it can decrease it. In the case

when the depth decreases, it is the consequence of the removal of the Z-corrections on the measured qubits by applying Rule 4.ons on the measured qubits by applying Rule 4.

The Rules 4 - 7 can be interpreted in the following way. Signal shifting takes a signal from a Z-correction on a measured qubit i (Rule 4) and adds it to the corrections that depend on the outcome of the measurement of i (Rules 5 - 6). When the signal moves to an X-correction command, then it won't propagate any further. If the signal was added to another Z-correction of a measured vertex, then signal shifting can be applied again until no Z-corrections are left on non-output vertices. An example of the process of signal shifting can be seen in Figure 4. Therefore signals move along a path created by the Z-corrections. The propagation of signals in an MBQC pattern can be described by a *Z-path* as defined below.

Definition 8 (Z-Path) *Let M be a measurement pattern on an open graph (G, I, O). Then we define a directed acyclic graph, called Z-correction graph and noted G_Z, on the vertices of G such that there exists a directed edge from i to j iff there exists a correction command $Z_j^{s_i}$ in M. A path in G_Z between two vertices v and u is called a* Z-path.

When we say that two vertices are connected, we always mean connected *via* an edge. When paths between vertices are considered it will noted explicitly to avoid confusion. The above definition allows us to state a simple observation about the connectivity of a graph with flow.

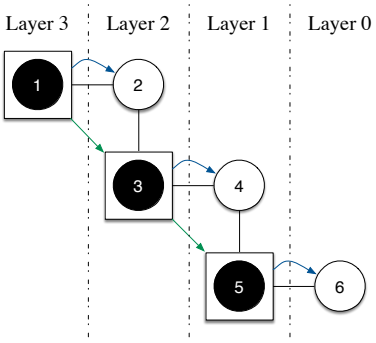
Lemma 4 *If $(f, \prec_f)$ is a flow on an open graph (G, I, O), and there exists a Z-path from vertex i to vertex j, then the vertices i and $f(j)$ cannot be connected.*

Proof. The existence of a Z-path from i to j implies that $i \prec_f j$. The Z dependency graph is an acyclic graph, thus $i \neq j$. If i would be connected to $f(j)$, then according to the flow property (F2) $j \prec_f i$. Now we have two contradicting strict partial order relations $i \prec_f j$ and $j \prec_f i$. Therefore i cannot be connected to $f(j)$ $\square$.

Recall that the addition of signals is done modulo 2, therefore, if an even number of signals from a measured vertex i is added to a correction command on vertex j, the signals will cancel out (since $Z^2 = X^2 = I$). Furthermore, it is evident from the rewrite Rules of 4 - 6 that after signal shifting, the measurement result of vertex i will create a new X-correction over vertex j if there exists an odd number of Z-paths from i to a vertex k such that j is X-dependent k in the original pattern. Similarly a new Z-correction from i to j will be created if there exists an odd number of Z-paths from i to j. Either way, *the number of Z-paths from a vertex i to another vertex j*, denoted as $\zeta_i(j)$, can be used to determine if the signal from i should be added to a correction. We define $\zeta_i(i)$ to be 1 to simplify further calculations and definitions in this paper. The importance of the number of Z-paths will manifest itself in the next subsection, when the relation between signal shifting and gflows is studied.

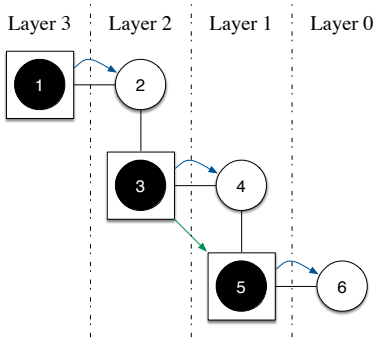
3.1 Signal Shifted Flow

We define a new structure called the *signal shifted flow* (SSF), and show that it satisfies the three gflow properties in Definition 3. Before constructing the SSF, some definitions and lemmas are needed to justify our definition. We define the *Z-dependency neighbourhood* of a vertex j to be the set of vertices from which j is receiving a Z-correction from. This set has an explicit form given as $N_Z(j) = \{k \in O^C | f(k) \in N(j) \setminus \{f(j)\}\}$. This is due to the following facts: first, $f(k)$ has to exist for all vertices $k \in O^C$ because of the flow definition; second, since $f(k) \neq f(j)$ the vertex k cannot be equal to j. Because $f(k) \in N(j) \Rightarrow j \in N(f(k))$ and Corollary 1 there exists a Z-correction from k to j. It is easy to see, that $\zeta_i(j)$ can be



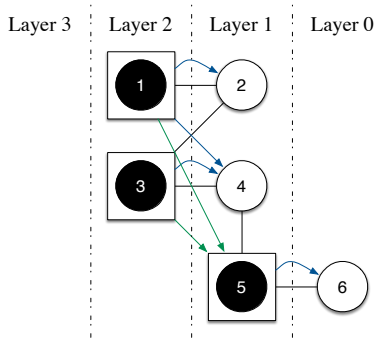
$X_6^{s_5}\ {}_{s_3}[M_5^{\alpha_5}]X_4^{s_3}\ {}_{s_1}[M_3^{\alpha_3}]X_2^{s_1}\ [M_1^{\alpha_1}]E_G N_{I^C}$

(a) The measurement pattern before signal shifting.



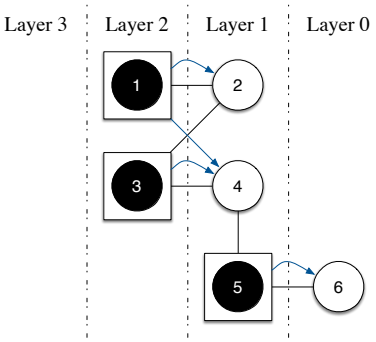
$X_6^{s_5}\ {}_{s_3}[M_5^{\alpha_5}]X_4^{s_3}\ S_3^{s_1}\ [M_3^{\alpha_3}]X_2^{s_1}\ [M_1^{\alpha_1}]E_G N_{I^C}$

(b) Creating the signal command from qubit 3.



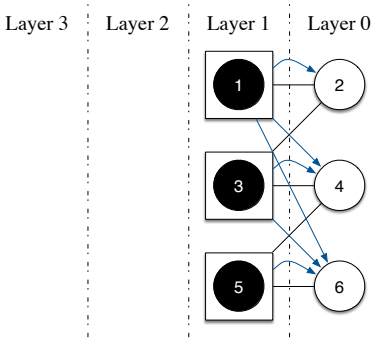
$S_3^{s_1} X_6^{s_5}\ {}_{s_3+s_1}[M_5^{\alpha_5}]X_4^{s_3+s_1}\ [M_3^{\alpha_3}]X_2^{s_1}\ [M_1^{\alpha_1}]E_G N_{I^C}$

(c) Moving the signal from qubit 3 to the end of the pattern.



$S_3^{s_1} X_6^{s_5}\ S_5^{s_3+s_1}[M_5^{\alpha_5}]X_4^{s_3+s_1}\ [M_3^{\alpha_3}]X_2^{s_1}\ [M_1^{\alpha_1}]E_G N_{I^C}$

(d) Creating the signal command from qubit 5.



$S_3^{s_1} S_5^{s_3+s_1} X_6^{s_5+s_3+s_1}\ [M_5^{\alpha_5}]X_4^{s_3+s_1}\ [M_3^{\alpha_3}]X_2^{s_1}\ [M_1^{\alpha_1}]E_G N_{I^C}$

(e) Moving the signal from qubit 5 to the end of the pattern.

Fig. 4. Signal shifting a measurement pattern. In this particular example, the depth of the computation will be reduced from 4 to 2 through signal shifting. The green and blue arrows represent Z- and X-corrections correspondingly.

written as:

$$\zeta_i(j) = \sum_{k \in N_Z(j)} \zeta_i(k). \tag{8}$$

There exists a Z-correction from every $k \in N_Z(j)$ to j. These Z-corrections can be used to extend every such Z-path to k to reach j. If i is in the sum, then because $\zeta_i(i) = 1$ the correct number of Z-paths is obtained with Equation 8.

All the properties proved so far allow us to present a new algorithm (Algorithm 1) for performing the signal shifting rules over pattern with flow in the form of Equation 2. We keep in mind that the order in which we apply the signal shifting rules does not matter [1]. This algorithm as we discuss later leads to a more efficient gflow finding algorithm.

Algorithm 1: *SignalShift*

Input: A measurement pattern P of a flow $(f, \prec_f)$ as defined in Equation 2.
Output: The signal shifted pattern P_{SS} of P.

begin
  $toShift = O^C$;
  $P_{SS} = P$;
  while $toShift \neq \emptyset$ **do**
    select any vertex $i \in toShift$ which is the smallest according to $\prec_f$;
    $toShift = toShift \setminus \{i\}$;
    while $\exists k \in toShift$ *s.t.* $Z_k^{s_i} \in P_{SS}$ **do**
      Select the smallest $k \in toShift$ s.t. $Z_k^{s_i} \in P_{SS}$ according to $\prec_f$;
      In P_{SS}, move the $Z_k^{s_i}$ command next to the $M_k^{\alpha_k}$ command;
      Use Rule 4 on P_{SS} to create the signal command $S_k^{s_i}$;
      `// Removes the` $Z_k^{s_i}$ `command from` P_{SS};
      Use Rule 6 on P_{SS} to create a new $X_{f(k)}^{s_i}$ command;
      foreach $j \in N(f(k)) \setminus \{k\}$ **do**
        Use Rule 7 on P_{SS} to create a new $Z_j^{s_i}$ command.
      In P_{SS}, move $S_k^{s_i}$ to the end of the pattern and remove it.

Proposition 1 *Given as input the measurement pattern P of a flow $(f, \prec_f)$ defined in Equation 2, Algorithm 1 outputs the signal shifted measurement pattern of P.*

Proof. We will prove this proposition by showing that:

- Algorithm 1 always terminates.
- Every step in Algorithm 1 that modifies the pattern P_{SS} is a valid application of a signal shifting rewrite rule.
- The output of Algorithm 1, the pattern P_{SS}, is signal shifted.

Note that the "for" loop in the algorithm terminates because the underlying open graph is finite. The first "while" loop will terminate since we decrease the number of elements in $toShift$ on each loop iteration and never add anything to it. As we'll explain now, the second

"while" loop will terminate since each k is selected only once and the open graph is finite. Note that since $j \in N(f(k)) \setminus \{k\}$, then according to the flow definition $k \prec_f j$. Every new $Z_j^{s_i}$ command added to P_{SS} is such that $k \prec j$ and hence no removed $Z_k^{s_i}$ command can be added to P_{SS} again. This is true also on on subsequent loop iterations since we choose k to be the smallest according to $\prec_f$. Hence the second "while" loop and the whole algorithm terminates.

For Algorithm 1 to actually perform the signal shifting, its operations have to be either trivial commuting rules or the four signal shifting Rules 4 - 7. As can be easily seen from the algorithm, the operations done are indeed the signal shifting rewrite Rules 4 - 7. We still need to prove, that these rules can be applied in the order shown in the algorithm. Obviously we can use Rule 4 on line 8 to create the signal command due to the fact that $k \in toShift \subseteq O^C$ and that every non-output qubit is measured. Hence we have the measurement required for the creation of the signal command in the pattern. We know that $Z_k^{s_i}$ has to be in the pattern after the command $M_i^{\alpha_i}$ and before $M_k^{\alpha_k}$. The entanglement and creation commands are the first commands in the pattern and we do not need to move the $Z_k^{s_i}$ command past them. Hence we only need to move $Z_k^{s_i}$ past measurement commands on qubits that are not i and k and other correction commands. These can be done trivially and hence we can always move the $Z_k^{s_i}$ command next to $M_k^{\alpha_k}$ to apply Rule 4.

Next we want to move the newly created S_k^i command to the end of the measurement pattern. To do that we need to commute it past the commands that appear after it. The only commands S_k^i commutes non-trivially with are the ones that depend on the measurement of qubit k as can be seen from Rules 4 - 7. Those are the X- and Z-corrections depending on the measurement outcome of qubit k. According to Equation 2 there is exactly one such X-correction in the pattern P, namely $X_{f(k)}^k$. The previous steps of the algorithm could not have created any dependencies from qubit k — the Z-correction commands have only been created depending on vertices that we already moved from $toShift$. Therefore we need to create exactly one new X-correction command using Rule 6. We also look at the Z-corrections depending on k and from Equation 2 we see that in the original pattern these are on vertices from the set $N(f(k)) \setminus \{k\}$. Just like for the X-corrections, we have not created any new Z-corrections from k in the previous steps of the algorithm. Hence this is exactly the set of corrections we need to commute with and apply Rule 7. We are only left with commands after S_k^i in the pattern that commute trivially with S_k^i. We can move the command to the end of the pattern. The signal command at the end of the pattern does not influence the computation and we will not add any new commands to the end of the pattern. Hence we can remove the S_i^k command.

Finally we show that no more signal shifting rules can be applied after the completion of Algorithm 1, *i.e.* the pattern P_{SS} is signal shifted. We eliminate all Z-corrections acting on a non-output qubit depending on a vertex i after removing it from the set $toShift$ and will afterwards never create any new Z-corrections depending on that vertex. At the end of the algorithm the set $toShift$ is empty, hence there cannot exist any non-output qubit that has a Z-correction acting on it and Rule 4 cannot be applied anymore. Moreover, since every signal command is at the end of the pattern, we cannot apply the Rules 5 to 7 either. This completes the proof □.

Proposition 2 *Algorithm 1 completes in $O(n^3)$ steps, where n is the number of qubits the*

input pattern acts on.

Proof. The number of times the outermost and innermost loops are executed is easy to estimate. The first `while` loop is entered $|O^C| = O(n)$ times and the inner `foreach` loop $\deg(G) = O(n)$ times, where $\deg(G)$ is the degree of the graph. The number of times the second `while` loop is entered depend on the number of times a $Z_j^{s_i}$ command is added on line 15. Adding new Z-correction commands cannot create any loops, as otherwise the algorithm would not halt as proven in Proposition 1. Since there cannot be created any loops, the number of new $Z_j^{s_i}$ corrections added can only be $O(n)$ and the second `while` loop can only be entered $O(n)$ times. Therefore the Algorithm 1 completes in $O(n) \cdot O(n) \cdot O(n) = O(n^3)$ steps □.

We consider any trivial commutation of the commands of a pattern resulting in an equivalent pattern. Therefore given the measurement pattern P of a flow $(f, \prec_f)$ as defined in Eq. 2 as input to Algorithm 1, the output will be the unique signal shifted measurement pattern of P. Note that Algorithm 1 works almost like a directed graph traversal, where there is a directed edge from vertex i to k iff there exists the command $Z_k^{s_i}$ in the measurement pattern. The only difference from a classical directed graph traversal is that we allow visiting of a vertex more than once. Hence we will traverse through every different path in the graph. However we do that exactly once. As mentioned before, the evenness of the number of Z-paths can be used to determine if a signal is added to a correction command. If an open graph has a flow, the oddness of $\zeta_i(j)$ can be found as described in the following lemma.

Lemma 5 *For every two vertices i and j in an open graph (G, I, O) with flow $(f, \prec_f)$*

$$\zeta_i(j) \bmod 2 = |\{k \in N_Z(j) \mid \zeta_i(k) = 1 \bmod 2\}| \bmod 2$$

i.e. the oddness of $\zeta_i(j)$ depends only on the number of vertices in the Z-dependency neighbourhood which have an odd number of Z-paths from i.

Proof. $\zeta_i(j) \bmod n$ can be written as

$$\begin{aligned}
\zeta_i(j) \bmod 2 &= \left(\sum_{k \in N_Z(j)} \zeta_i(k) \right) \bmod 2 \\
&= \sum_{k \in N_Z(j)} (\zeta_i(k) \bmod 2) \bmod 2 \\
&= \sum_{\{k \in N_Z(j) | \zeta_i(k) \bmod 2 = 1\}} (\zeta_i(k) \bmod 2) \bmod 2 \\
&= |\{k \in N_Z(j) \mid \zeta_i(k) \bmod 2 = 1\}| \bmod 2
\end{aligned}$$

□.

All these notions will allow us to define the structure of the pattern after signal shifting is performed.

Proposition 3 *Given a flow $(f, \prec_f)$ on an open graph (G, I, O), let s be a function from $O^C \to P^{I^C}$ such that $j \in s(i)$ iff $\zeta_i(f^{-1}(j)) \bmod 2 = 1$. Also define L_s to be a layering function from $V(G)$ into a natural number:*

$$\begin{aligned}
L_s(i) &= 0 && \forall i \in O \\
L_s(i) &= \max_{j \in s(i)} (L_s(j) + 1) && \forall i \notin O
\end{aligned}$$

Define the strict partial order $\prec_s$, where $i \prec_s j \Leftrightarrow L_s(i) > L_s(j)$. The application of signal shifting Rules 4 - 7 over an MBQC pattern with flow $(f, \prec_f)$ will lead to the following pattern:

$$P = \prod_{j \in O, i \in I^C} Z_j^{s_i \zeta_i(j) \bmod 2} \prod_{i \in O^C}^{\prec_s} \left(X_{s(i)}^{s_i} M_i^{\alpha_i} \right) E_G N_{I^C}. \tag{9}$$

Proof. The proof is divided into three parts. First we will show that signal shifting creates exactly the commands comprising the pattern shown in Equation 9. These commands do not have to be in the same order as in Equation 9. We proceed by showing that the layering function L_s is defined for every $i \in V(G)$. Lastly, we need to prove that using the partial order $\prec_s$ derived from L_s for ordering the commands as in Equation 9 gives a valid measurement pattern.

Note that the preparation commands (N_I^C), entanglement commands (E_G) and measurement commands ($M_i^{\alpha_i}$) are the same for Equations 2 and 9. Because signal shifting would not change these commands (Rules 4 - 6) these are as required for a signal shifted pattern. Hence we need only to consider the correction commands.

We will look at the correction commands that would appear in a signal shifted pattern. We do this by examining the signal shifting algorithm (Algorithm 1). As mentioned before, the algorithm works as a directed graph traversal, in a way that every distinct path is traversed. As seen in the algorithm every $Z_k^{s_i}$ correction acting on a non-output qubit is removed from the pattern. This is in accordance with the proposed pattern in Equation 9. Let us examine which new corrections are created.

The number of newly created $X_j^{s_i}$ depends on the number of times we enter the first loop with command $Z_{f^{-1}(j)}^{s_i}$. As the algorithm is a directed graph traversal algorithm, this happens as many times as there are different paths over the Z-dependency graph from i to $f^{-1}(j)$. Because the same two $X_l^{s_i}$ corrections cancel each other, a new X-correction appears in a signal shifted pattern only if $\zeta_i(f^{-1}(j)) \mod 2 = 1$. We also note that no new $X_{f(i)}^{s_i}$ correction is created since there exists no Z-path between i and $f^{-1}(f(i))$. On the other hand Algorithm 1 leaves the already existing X- corrections unchanged and moreover since we have defined $\zeta_i(i) = 1$, we have $f(i) \in s(i)$. This implies that the set $s(i)$ does indeed contain all the vertices that have an X-correction depending on s_i after signal shifting is performed.

The number of newly created Z-corrections on an output vertex j depending on a vertex i appearing in the signal shifted pattern is equal to the number of different paths from i to j. The difference with non-output qubits is that these will not be removed through the process of signal shifting. As with X-corrections, two Z-correction commands on the same qubit will cancel each other out and hence the existence of a $Z_j^{s_i}$ in the final pattern depends on the parity of the number of paths from i to j. This can be written in short as $Z_j^{s_i \zeta_i(j) \bmod 2}$. Hence the measurement pattern in Equation 9 has exactly the same commands as the signal shifted pattern in Equation 2.

Another thing we need to prove is that the layering function L_s is defined for every $i \in V(G)$. As proven above, the X-corrections depending on the measurement of qubit v correspond to the set $s(v)$. Hence we can interpret the definition of $L_s(v)$ as finding the maximum value of L_s for every vertex that has an X-correction from v and adding 1 to it. The recursive definition of $L_s(v)$ is well defined, if for every non-output qubit we can find a

path over X-corrections ending at an output qubit. We know that signal shifting of a valid pattern creates another valid pattern. This implies that the X-corrections cannot create a cyclic dependency structure and hence every path over the X-corrections has an endpoint. Moreover such a path cannot end on a non-output qubit k since $f(k) \in s(k)$ and one could always extend that path with $f(k)$. Therefore $L_s(v)$ is well defined.

Finally, it is easy to show that the partial order $\prec_s$ as used in Equation 9 gives a valid ordering of the commands. Every vertex j that has an X-correction depending on the measurement of qubit i has a smaller L_s number and hence $i \prec_s j$. This way no X-correction command acts on an already measured qubit and because the Z-corrections are applied only on output qubits, the correction ordering is valid. Every other command is applied before the measurement command and hence the pattern in Equation 9 is a valid measurement pattern $\square$.

Given an open graph with a flow, we refer to the construction of the above proposition as its corresponding *signal shifted flow* (SSF).

4 SSF and gflow

As mentioned before, gflow is a sufficient and necessary condition for determinism while flow is only a sufficient condition. At first it seems that the simple local rewriting rules of signal shifting could not upgrade a flow to the more powerful gflow construction. Indeed the proof of this statement is not trivial either and is based on discovering various properties of flow of information in an SSF pattern.

Theorem 1 *Given any open graph (G, I, O) with flow $(f, \prec_f)$, the corresponding signal shifted flow $(s, \prec_s)$ is a gflow.*

The proof is based on the following lemmas, demonstrating that s is a gflow by satisfying all the properties of Definition 3. The first property (G1) of gflow is satisfied by SFF implicitly from Definition 3, *i.e.* for every $i \in V(G)$ it holds that $i \prec_s j$ if $j \in s(i)$. Consider the second gflow property (G2), *i.e.* if $j \in Odd(s(i))$ then $j = i$ or $i \prec_s j$. We will show that every vertex with an odd number of connections to $s(i)$ has to be either i itself or an output qubit.

Lemma 6 *If $(s, \prec_s)$ is an SSF then every non-output vertex $v \neq i$ connected to $s(i)$ has an even number of connections to $s(i)$, i.e.*

$$\forall v \in N(s(i)) \setminus \{O \cup i\} \quad \Rightarrow \quad v \notin Odd(s(i)).$$

Proof. Let $v \neq i$ be a vertex connected to $s(i)$, we show the following two sets have the same number of elements.

$$\{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\} \quad \text{and} \quad s(i) \cap N(v) \setminus \{f(v)\}.$$

For every $j \in s(i) \cap N(v) \setminus \{f(v)\}$, we prove $f^{-1}(j)$ is the unique element in

$$\{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\}.$$

Because $j \in s(i)$, from Proposition 3 there must exist $f^{-1}(j)$. Also, since $j \in N(v)$ therefore $v \in N(j) = N(f(f^{-1}(j)))$. Moreover since $j \neq f(v)$, Corollary 1 implies the existence of a Z-correction from $f^{-1}(j)$ to v, *i.e.* $f^{-1}(j) \in N_Z(v)$. Proposition 3 says that because $j \in s(i)$, it must hold that $\zeta_i(f^{-1}(j)) \bmod 2 = 1$. Therefore $f^{-1}(j) \in \{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\}$.

On the other hand, for every vertex $u \in \{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\}$, as $\zeta_i(u) \bmod 2 = 1$ then from Proposition 3 we have $f(u) \in s(i)$. Also, $f(u) \in N(v)$ because of Corollary 1 and finally, $f(u) \neq f(v)$ because v cannot have a Z-correction from itself, *i.e.* $v \notin N_Z$. Hence it holds that $f(u) \in s(i) \cap N(v) \setminus \{f(v)\}$ and

$$|s(i) \cap N(v) \setminus \{f(v)\}| = |\{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\}|.$$

According to Lemma 5 $\zeta_i(v) \bmod 2 = |\{k \in N_Z(v) \mid \zeta_i(k) \bmod 2 = 1\}| \bmod 2$. If $\zeta_i(v) \bmod = 0$, then $s(i) \cap N(v) \setminus \{f(v)\}$ must have an even number of elements. Proposition 3 says that $f(v)$ cannot be in $s(i)$ and therefore v can have only even number of connections to $s(i)$. If $\zeta_i(v) \bmod = 1$, then we know that $s(i) \cap N(v) \setminus \{f(v)\}$ must have an odd number of elements. If $f(v)$ exists, it must be in $s(i)$ because of Proposition 3. In the case of $f(v) \in s(i)$, we can conclude that $|s(i) \cap N(v)| \bmod 2 = 0$ and v has an even number of connections to $s(i)$. On the other hand if $f(v)$ does not exist, v has to be an output qubit because the flow function f is defined for every non-output vertex. The only possibility of k having odd many connections to $s(i)$ is therefore if k is an output vertex, which proves the lemma $\square$.

The next lemma directly proves that an SSF also satisfies the last gflow property (G3) which states that $i \in Odd(s(i))$.

Lemma 7 *If $(s, \prec_s)$ is an SSF, then for every $i \in O^C$ it holds that $i \in Odd(s(i))$.*

Proof. First we show that, performing signal shifting creates new X-corrections only between unconnected vertices. Recall that signal shifting creates a new X-correction between vertices i and j iff there exists a Z-path from i to $f^{-1}(j)$ and an X correction from $f^{-1}(j)$ to j therefore from the Flow definition we have:

$$i \prec_f f^{-1}(j) \prec_f j.$$

Let us assume that there exists an edge between i and j. According to the Flow definition

$$\left.\begin{aligned} i \in N(j) \\ j = f(f^{-1}(j)) \end{aligned}\right\} \quad \Rightarrow \quad i \in N(f(f^{-1}(j))) \quad \Rightarrow \quad f^{-1}(j) \prec_f i.$$

This contradicts the partial order $i \prec_f f^{-1}(j) \prec_f j$ of the Flow $(f, \prec_f)$ and therefore there cannot be an edge between vertices i and j.

Next we claim that there is exactly one edge between i and $s(i)$. According to Definition 3 of SSF, the set $s(i)$ consists only of the vertex $f(i)$ and the vertices to which signal shifting created a new X dependency from i. We showed that signal shifting does not create X dependencies between connected edges. Hence, $f(i)$ is the only vertex in $s(i)$ that can be connected to i, and there must be an edge between i and $f(i)$ because of the flow property (F3) $(i \in N(f(i)))$ $\square$.

Proof of Theorem 1. Note that the definition of SSF implies the gflow property (G1). Lemma 6 implies that every SSF satisfies the gflow condition (G2). As the third and last gflow condition is satisfied by SSF according to Lemma 7, SSF is indeed a gflow and Theorem 1 holds.

The above theorem for the first time presents a structural link between two seemingly different approach for parallelisation, gflow and signal shifting, for patterns with flow. The

next sections explores further links between SSF and gflow, showing the usefulness of SSF in parallelisation.

We conclude with a simple corollary showing that SSF is in fact a special case of focused flow, where the later was defined to capture a wider notion of determinism, *i.e.* information preserving map.

Corollary 2 *Given any open graph (G, I, O) with flow $(f, \prec_f)$ and corresponding signal shifted flow $(s, \prec_s)$ the function s is a focused flow.*

Proof. For $(s, \prec_s)$ to be a focused gflow, s must satisfy the two properties (FG1) and (FG2) in Definition 7. First, it is clear from Proposition 3 that the transitive closure of the relation $\{(i, j)\ s.t.\ j \in g(i)\}$ is a strict partial order over $V(G)$ and hence property (FG1) is satisfied. Second, Lemma 6 proves that $Odd(s(i)) \cap O^C \setminus \{i\} = \emptyset$ and Lemma 7 states that $i \in Odd(s(i))$. Combining these two lemmas creates exactly the second focused flow property (FG2) $\forall i \in O^C, Odd(s(i)) = \{i\}$ $\square$.

5 Properties of SSF

The notions of *influencing walks* and *partial influencing walks* on open graphs with flow was introduced in [5] to describe the set of all vertices that a measurement depends on. An influencing walk starts with an input and ends with an output vertex, a partial influencing walk starts with an input vertex but can end with a non-output vertex. We will use a modified definition of influencing walks that can start from any non-output vertex i and end at any vertex $j \in s(i)$ and call it a stepwise influencing path. This will allow us to conveniently explore the dependency structure of a pattern with SSF.

Definition 9 *Let $(s, \prec_s)$ be an SSF that is obtained from a flow $(f, \prec_f)$ of an open graph (G, I, O) and vertices i and j in $V(G)$ such that $j \in s(i)$. We say that a path between vertices i and j is an* stepwise influencing path*, noted as $\wp_i(j)$, iff*

- *The path is over the edges of G.*
- *The first two elements on the path are i and $f(i)$.*
- *Every even-placed vertex k on the path $\wp_i(j)$, starting from $f(i)$, is in $s(i)$.*
- *Every odd-placed vertex on the path $\wp_i(j)$ is the unique vertex $f^{-1}(k)$ of some $k \in s(i)$ such that k is the next vertex on the path $\wp_i(j)$.*

It is easy to see that every second edge, in particular the edges between $f^{-1}(k)$ and $k \in s(i)$, in the stepwise influencing path is a flow edge. Hence the path contains no consecutive non-flow edges. If we restrict the first vertices of the stepwise influencing path to be input vertices, the stepwise influencing path would be a partial influencing path, but not *vice versa*. Before we can show how stepwise influencing paths relate to SSF we need to prove the following property of SSF.

Lemma 8 *If (G, I, O) is an open graph with flow $(f, \prec_f)$ and SSF $(s, \prec_s)$ then for every vertex i and j such that $f(j) \in s(i) \backslash \{f(i)\}$, we can find another vertex k, such that $f(k) \in s(i) \cap N(j)$.*

Proof. If $f(j) \in s(i)$, then from the Proposition 3 of SSF we can conclude that $\zeta_i(j) \bmod = 1$. We know that $j \neq i$ from the assumptions. Lemma 5 says that there must exist at least one other vertex k from which j has a Z-correction, such that $\zeta_i(k) \bmod 2 = 1$. The flow

definition says that j must therefore be a neighbour of $f(k)$. Definition 3 of SSF states that $f(k)$ must therefore be in $s(i)$, hence $f(k) \in s(i) \cap N(j)$ $\square$. Now it can be shown that when we have a SSF $(s, \prec_s)$, there exist stepwise influencing paths from $i \in O^C$ to every vertex in $s(i)$.

Lemma 9 *Let $(s, \prec_s)$ be an SSF obtained from a flow $(f, \prec_f)$ of an open graph (G, I, O) and vertices i and j in $V(G)$ such that $j \in s(i)$. Then there always exists a stepwise influencing path $\wp_i(j)$.*

Proof. We start by constructing such a path backward from j to i. We select j and $f^{-1}(j)$ as the last two vertices on the path and apply Corollary 8 to find the vertices on the path, until we reach i. The formation of cycles is impossible, as this would imply a cyclic dependency structure, impossible for a flow. We have to reach i as the set of vertices we choose from is finite $\square$.

Note that there might be more than one stepwise influencing path from i to j. We conclude this section with the following lemma (illustrated in Figure 5) which explains how a stepwise influencing path can be extended.

Lemma 10 *Let (G, I, O) be an open graph with flow $(f, \prec_f)$ and corresponding SSF $(s, \prec_s)$ and let i and j be two non-output vertices of the open graph such that $f(j) \in s(i)$. If $v \in N(j) \cap s(i) \setminus \{f(j)\}$ then every stepwise influencing path $\wp_i(v)$ can be extended by the vertices j and $f(j)$ to create another stepwise influencing path $\wp_i(f(j))$.*

Proof. Adding j and $f(j)$ to $\wp_i(v)$ satisfies the conditions for stepwise influencing paths. There exists an edge between vertices j and v and vertices j and $f(j)$, hence it is a valid path. Moreover, $f(j) \in s(i)$ would be an even-placed vertex on the extended path, and j would be the unique oddly-placed vertex with $f(j) \in s(i)$ $\square$.

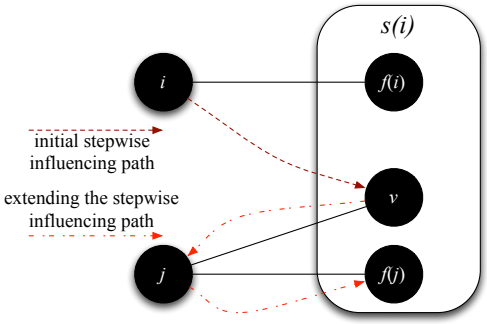


Fig. 5. Extending a stepwise influencing path ending at vertex v according to Lemma 10.

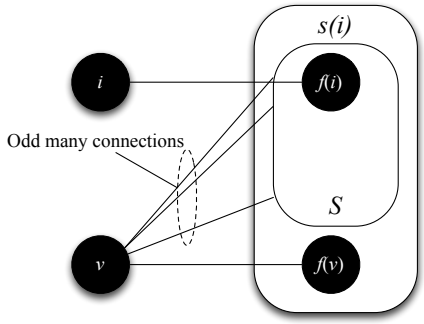


Fig. 6. For every $S \subsetneq s(i)$ containing $f(i)$ we can find a vertex $v \in Odd(S)$ such that $f(v) \notin S$.

6 Computational Depth of SSF

Given an MBQC pattern with gflow, finding the maximally delayed gflow of its underlying graph could potentially further reduce the depth of the computation [3]. A natural question that arises is how SSF is linked with the maximally delayed gflow. In this section, we prove that if the input and output sizes of the pattern are equal, then SSF is indeed the maximally delayed gflow. Hence we can conclude the optimal parallelisation that one could obtain via

translation of a quantum circuit into an MBQC pattern is achieved by the simple rewriting rules of SSF. This will also lead to a more efficient algorithm than the one presented in [3] for finding the maximally delayed gflow of a graph as we discuss later.

Theorem 2 *Let (G, I, O) be an open graph with flow $(f, \prec_f)$ such that $|I| = |O|$. Let $(s, \prec_s)$ be the SSF obtained from $(f, \prec_f)$. Then $(s, \prec_s)$ is the maximally delayed gflow of (G, I, O).*

The proof of the theorem is rather long, an outline is presented below. A general reader could omit the next subsections, however various novel constructions has been introduced in the proof that could be explored for other MBQC results and hence could be valuable for an MBQC expert. In Section 6.1 we show that the penultimate layers of an maximally delayed gflow and an SSF of an open graph where $|I| = |O|$, are equal. Next we introduce the concept of a *reduced open graph* in Section 6.2. We prove two key properties of the maximally delayed gflow and SSF of the reduced open graph. This highlights the recursive structures of the gflow and SSF leading to the possibility of extending these notions to new domains.[a] In Section 6.3 we put the pieces together, by showing that the previous properties imply that reduced gflow (implicitly also maximally delayed gflow and SSF) layers are equal to the original gflow layers from layer 1 onward. This allows us to construct a recursive proof for Theorem 2, which we present in Section 6.4.

6.1 The Last Two Layers

The equality of the last layers of an SSF and maximally delayed gflow follows from Lemma 2 and Proposition 3 — the last layer of an maximally delayed gflow and an SSF is always the set of output vertices. To prove Theorem 2 we need to show that the penultimate layers of SSF and maximally delayed gflow have the same size. One might think that because the penultimate layer of a gflow contains all the vertices that can be corrected by the vertices in the output layer, surely when $|I| = |O| = n$ the number of elements in the penultimate layer would also be n. If that would be true we could omit the proofs in this section and skip to section 6.2, but this is not the case as demonstrated in Figure 7. To prove that

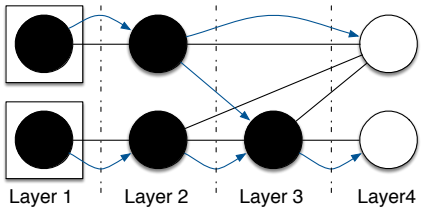


Fig. 7. An example of a gflow where $|I| = |O|$ and the penultimate layer has less element than the last one.

the penultimate layers of SSF and maximally delayed gflow are equal we need the following properties of open graphs with SSF. An illustration of the property proven in the first of the two lemmas is shown in Figure 6.

Lemma 11 *Let (G, I, O) be an open graph with flow $(f, \prec_f)$ and corresponding SSF $(s, \prec_s)$. If $i \in O^C$ then for every strict subset S of $s(i)$ containing $f(i)$ there must exist a non-output*

[a]For example, the authors are currently exploring this structure to define the concept of partial flow, for patterns with no deterministic computation.

vertex v that is oddly connected to S such that $f(v) \in s(i) \setminus S$, i.e.

$$\forall i \in O^C \quad \forall S \subset s(i) \quad s.t. \quad f(i) \in S \quad \exists v \in Odd(S) \quad s.t. \quad f(v) \in s(i) \setminus S$$

Proof. If $s(i) = \{f(i)\}$ the lemma holds trivially, as there does not exist any nonempty strict subsets of $s(i)$. Consider the case where $s(i)$ contains more than one element and S is a strict subset of $s(i)$. Then we select any vertex $j \notin S$ from $s(i)$ and look at the stepwise influencing paths from i to j. Note that there might be more than one such path. We move backwards from j towards i over the stepwise influencing paths in the following way:

1. Move by two vertices
   (a) If possible, choose any stepwise influencing path where the previous even-placed element is not in S and move to that element.
   (b) If the previous even-placed elements in all the stepwise influencing paths from i to j are in S, then stop.
2. Repeat step 1.

Let u be the vertex to where we moved using the above process, u has to exist because of the way we initially selected j. There are a couple of other observations that we can make about u. First, $u \in s(i) \setminus S$, because of the selection of j and the way we moved on the paths. Second, u cannot be the first even placed vertex on a stepwise influencing path from i to u because the first element is $f(i) \in S$ (according to Definition 9). Third, for every stepwise influencing path ending in u, the previous even-placed vertex has to be in S as otherwise we could have moved one more step towards i.

Considering the previous three observations we can show that the vertex $v = f^{-1}(u)$ must be oddly connected to S. We begin by noting that v cannot be connected to any vertex $k \in s(i) \setminus (S \cup \{f(v)\})$. Otherwise, according to Lemma 10, we could extend any stepwise influencing path ending at k with v and $f(v)$. Hence $k \notin S \cup \{f(v)\}$ would then be an even-placed vertex on a stepwise influencing path from i to $f(v)$. In particular, k would be the second to last even-placed vertex on a stepwise influencing path from i to $f(v) = u$ Every such vertex, except $f(v)$ itself, is in S as mentioned before. According to Lemma 6, v has to be evenly connected to $s(i)$ and thus oddly connected to S and Lemma 11 holds. □.

Note that because of Definition 6 if a gflow is not maximally delayed, its penultimate layer has to either be equal to the penultimate layer of the maximally delayed gflow or there exists a vertex in the penultimate layer of maximally delayed gflow that is not included in the penultimate layer of the other gflow. In the proof of the main result we assume that the penultimate layers are not equal, hence we could choose a vertex with particular properties (described in the next two lemmas) to derive a contradiction.

Lemma 12 *Let (G, I, O) be an open graph where $|I| = |O|$ with flow $(f, \prec_f)$, corresponding SSF $(s, \prec_s)$ and a gflow $(g, \prec_g)$ such that $V_0^{\prec_g} = O$. Assume there exists a vertex $i \in V_1^{\prec_g} \setminus V_1^{\prec_s}$, then*

- $g(i) \subseteq O$
- $g(i) \cap s(i) \subsetneq s(i)$

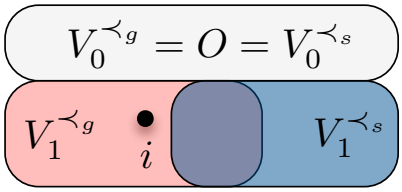


Fig. 8. The initial conditions required for Lemma 12.

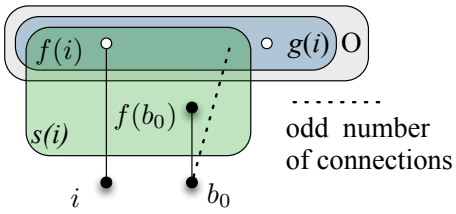


Fig. 9. The final conditions proved in Lemma 12.

- $f(i) \in g(i)$

and there exists a vertex b_0 *such that*

- $b_0 \in Odd(g(i) \cap s(i))$
- $f(b_0) \in s(i) \setminus g(i)$

Proof. Because i is in $V_1^{\prec_g}$ the set $g(i)$ must be a subset of $V_0^{\prec_g} = O$ according to Definition 4. Proposition 3 implies that $V_0^{\prec_s} = O$. This and the fact that $i \notin V_1^{\prec_s}$ implies that $s(i)$ is not a subset of the output vertices $O = V_0^{\prec_s}$. Therefore there must exist a non-output vertex in $s(i)$ and, because $g(i) \subseteq O$, this vertex cannot be contained in $g(i)$. Thus the intersection of $s(i)$ and $g(i)$ cannot be equal to $s(i)$ and $g(i) \cap s(i) \subset s(i)$.

We now show that $f(i) \in g(i)$. Let us assume that $f(i) \notin g(i)$, and choose a vertex $a_1 \in g(i)$ connected to i, such a vertex has to exist because the gflow definition says that i is oddly connected to $g(i)$. As $a_1 \in g(i)$ then by the gflow definition a_1 cannot be an input qubit. It can be shown that since $a_1 \notin I$ and $|I| = |O|$, $f^{-1}(a_1)$ has to be defined and because of the definition of flow it is connected to a_1. By the definition of flow, $f^{-1}(a_1)$ cannot be an output vertex and thus is not in layer $V_0^{\prec_g}$. As $g(i) \subseteq O$ this also means $f^{-1}(a_1) \notin g(i)$. On the other hand $f^{-1}(a_1)$ is connected to $a_1 \in g(i)$. Because $i \in V_1^{\prec_g}$ and $f^{-1}(a_1) \notin V_0^{\prec_g}$ we know from Definition 4 that $i \not\prec_g f^{-1}(a_1)$. As $f^{-1}(a_1)$ is connected to $g(i)$ we can conclude from the gflow definition that $f^{-1}(a_1)$ has to be evenly connected to $g(i)$ and therefore has at least one more connection to a vertex $a_2 \in g(i)$.

Using the same argument for a_2 as for a_1 we can say that there must exist $f^{-1}(a_2) \notin g(i)$ to which a_2 is connected to. Let us assume that $f^{-1}(a_2)$ is not connected to a_1. This means it has only one connection to the set $A_2 = \{a_1, a_2\} \subseteq g(i)$ and is therefore oddly connected to it. We can continue this procedure of selecting vertices from $g(i)$ until we select a vertex a_n such that $f^{-1}(a_n)$ is connected to at least one vertex a_j in $A_{n-1} = \{a_1, \dots a_{n-1}\} \subseteq g(i)$. If this happens we can no longer say with certainty that $f^{-1}(a_n)$ is oddly connected to $A_n \subseteq g(i)$, which means we cannot select any more elements from $g(i)$ using this method. Because (G, I, O) is a finite open graph we must find this a_n in finite number of steps.

We created the set A_n in such a way that:

$$\forall j \in \{1, 2, \dots, n-1\} \quad f^{-1}(a_j) \in N(a_{j+1}) = N(f(f^{-1}(a_{j+1})))$$

Hence we have a Z-correction from every $f^{-1}(a_{j+1})$ to $f^{-1}(a_j)$ and thus there exists a Z-path from $f^{-1}(a_n)$ to every $f^{-1}(a_j)$ such that $a_j \in A_{n-1}$ and, because of Lemma 9, $f^{-1}(a_n)$ cannot

be connected to any vertex in A_{n-1}. This leads to a contradiction with the assumption that it is connected to at least one vertex in A_{n-1}. Therefore our initial assumption that $f(i) \notin g(i)$ must be false and $g(i)$ must contain $f(i)$.

From the definition of SSF we have that $f(i) \in s(i)$ and therefore also $f(i) \in g(i) \cap s(i)$. Now we know that $g(i) \cap s(i)$ is a strict subset of $s(i)$ containing $f(i)$; the existence of b_0 follows from Lemma 11. $\square$.

Now we prove that if we have a vertex with the same properties as b_0 in Lemma 12 and a (possibly empty) subset A of vertices with particular properties (which will be defined in the next lemma) we can always increase the size of A and find another vertex with properties of b_0. This would imply the possibility of increasing the size of A to infinity and will give us the contradiction we need.

Lemma 13 *Let (G, I, O) be an open graph where $|I| = |O|$ with flow $(f, \prec_f)$, corresponding SSF $(s, \prec_s)$ and a gflow $(g, \prec_g)$. If we have a vertex i in the open graph such that*

- $g(i) \subseteq O$
- $g(i) \cap s(i) \subsetneq s(i)$
- $f(i) \in g(i)$

and if we have a subset $A \subseteq g(i)$ and another vertex b_0 such that

- $b_0 \in Odd(g(i) \cap s(i))$
- $f(b_0) \in s(i) \setminus g(i)$
- $\forall j \in A \quad \exists \quad b_0 \xrightarrow{Z} f^{-1}(j)$

then there exists another vertex c_o and a non empty set $B \subseteq g(i)$ such that

- $B \neq \emptyset$
- $B \cap A = \emptyset$
- $c_0 \in Odd(g(i) \cap s(i))$
- $f(c_0) \in s(i) \setminus g(i)$
- $\forall j \in A \cup B \quad \exists \quad c_0 \xrightarrow{Z} f^{-1}(j)$

Proof. The proof consists of three steps: we start by constructing the set B; we proceed with finding the vertex c_0; and finally we prove that c_0 has the required properties.

Define $S = g(i) \cap s(i)$, since $f(b_0)$ exists hence b_0 cannot be an output vertex. Also since $g(i) \subseteq O$ therefore b_0 is not in $g(i)$. As $b_0 \notin O = V_0^{\prec_g}$ and $g(i) \subseteq O$ we can conclude from Definition 4 that $i \in V_1^{\prec_g}$ and $i \not\prec_g b_0$. Therefore according to the gflow definition, b_0 must be in the even neighbourhood of $g(i)$. We also know from the initial conditions of this lemma that b_0 is in the odd neighbourhood of $g(i) \cap s(i)$. Thus there has to exist a vertex v_1 in $g(i)$ to which b_0 is connected to, but which is not included in $g(i) \cap s(i)$, *i.e.* $v_1 \in g(i) \setminus s(i)$. As $g : O^C \to P^{I^C}$, $v_1 \in g(i)$ cannot be an input qubit. It can be shown that since $|I| = |O|$, f^{-1} exists for every non-input qubit, there must exist a vertex $f^{-1}(v_1) = b_1$. It is also important

for the later part of the proof to note that $f(b_1) = v_1 \notin A$. This is due to Lemma 4, which implies that b_0 cannot be connected to any vertex in A.

Define $B_0 = S$ and consider the case when b_1 is evenly connected to B_0. Remember that the flow property (F3) says that there is always an edge between b_1 and $f(b_1)$. This means that b_1 is oddly connected to $B_1 = \{f(b_1)\} \cup B_0$ which is a subset of $g(i)$. But again because of the gflow property (G2) we have that b_1 must be evenly connected to $g(i)$. Thus there must exist another vertex b_2 such that b_1 is connected to $f(b_2) \in g(i) \setminus B_1$, otherwise b_1 could not be in the even neighbourhood of $g(i)$. If b_2 is evenly connected to B_1, it must be oddly connected to $B_2 = \{f(b_2)\} \cup B_1$ which is again a subset of $g(i)$. If b_2 is oddly connected to B_2 there must exist a vertex b_3 such that b_3 is connected to $f(b_3) \in g(i) \setminus B_2$, otherwise b_2 could not be in the even neighbourhood of $g(i)$. We can continue this scheme until we get to vertex b_n that is oddly connected to B_{n-1}. As $B_n = \{f(b_n)\} \cup B_{n-1}$ and there exists an edge between b_n and $f(b_n)$ we get that b_n must be evenly connected to B_n. Such vertex b_n must exist, otherwise we could continue selecting elements from $g(i)$ infinitely, but (G, I, O) is a finite open graph. We select $B = B_n \setminus S$. Recall that $f(b_1)$ must exist, therefore B must have at least on element.

Next we show b_n is oddly connected to S. We note that we have the following:

$$\forall j \in \{1, 2, \ldots, n\} \quad b_j \in N(f(b_j)) \quad \wedge \quad b_{j-1} \in N(f(b_j))$$

Corollary 1 implies that for every $j > 0$ there exists a Z-correction from b_j to b_{j-1}. Thus we have a Z-path from b_n to every other b_j where $j < n$, hence from Lemma 4 we conclude b_n cannot be connected to any vertex $f(b_j) \in B_{n-1}$ where $j < n$. The number of edges that connect the vertices in B_{n-1} to vertex b_n has to be the same as the number of edges between vertices of S and b_n, because $B_{n-1} = \{f(b_1), f(b_2), \ldots, f(b_{n-1})\} \cup S$. As b_n was oddly connected to B_{n-1}, it must also be oddly connected to S. Note that however b_n does not have the required properties for c_0, but will be used to find such a vertex.

The gflow definition says that b_n must be evenly connected to $s(i)$. It is also oddly connected to $s(i) \cap g(i)$ hence there must exist a vertex $c \in s(i) \setminus g(i)$ to which b_n is connected to. According to Lemma 9 there exists a stepwise influencing path $\wp_i(c)$ and due to Definition 9, $f(i)$ has to be on on this path. Therefore there exists at least one element in $\wp_i(c)$ that is in S. Let $f(a_0)$ be the last element of the path $\wp_i(c)$ in S.

Define a_1 to be the vertex in $\wp_i(c)$ that comes after $f(a_0)$. We know that a_1 has odd many Z-paths from i because Definition 9 implies that $f(a_1) \in s(i)$. If a_1 is already oddly connected to S, then we are done and $a_1 = c_0$. If a_1 is evenly connected to $S \subset s(i)$, then we know that it must be oddly connected to $S \cup \{f(a_1)\} \subseteq s(i)$. There must exist another vertex $f(a_2) \in s(i) \setminus (S \cup \{f(a_1)\})$ to which a_1 is connected to for it to be evenly connected to $s(i)$ as is required by Lemma 6. Because $f(a_2) \in s(i)$ we know there exists a stepwise influencing path $\wp_i(f(a_2))$ (Lemma 9) and we can extend that path by a_1 and $f(a_1)$ as was proven in Lemma 10. We move backward on this path and find the element a_2. If a_2 is oddly connected to S, we are done and set $c_0 = a_2$. Otherwise we can continue as was the case for a_1 until we find an element a_m that is oddly connected to S. This element must exist since graph is finite and the Z corrections do not create any loops. We select $c_0 = a_m$. Note that a_m cannot be i because $f(i) \in S = s(i) \cap g(i)$ but $f(a_m) \notin g(i)$.

There is a Z-path from $a_m = c_0$ to a_1 (we moved backwards along this path to find a_m) and from a_1 to b_n because of the way we selected a_1. There also exists a Z-path from b_n to every other b_j such that $0 \leq j < n$, thus a_m will also have a Z-path to every b_j in $\{b_1, b_2, \ldots, b_n\}$. Even more, because:

$$a_m \xrightarrow{Z} b_n \quad \wedge \quad b_n \xrightarrow{Z} b_0 \quad \wedge \quad \forall j \in A \qquad b_0 \xrightarrow{Z} f^{-1}(j) \quad \Rightarrow$$
$$\Rightarrow \quad \forall j \in A \qquad a_m \xrightarrow{Z} f^{-1}(j)$$

This completes the proof □.

Finally we can combine Lemmas 12 and 13 and prove that the penultimate layers of a SSF and maximally delayed gflow are equal.

Lemma 14 *Let (G, I, O) be an open graph with flow $(f, \prec_f)$, corresponding SSF $(s, \prec_s)$ and maximally delayed gflow $(g, \prec_g)$ such that $|I| = |O|$. Then $V_1^{\prec_s} = V_1^{\prec_g}$.*

Proof. Assume $V_1^{\prec_s} \neq V_1^{\prec_g}$ we show how we can choose infinitely many different vertices from $V(G)$. Due to Definition 6 we have $|V_1^{\prec_s}| \leq |V_1^{\prec_g}|$ and since $V_1^{\prec_s} \neq V_1^{\prec_g}$ hence trivially $V_1^{\prec_g} \not\subset V_1^{\prec_s}$ and there must exist a vertex i in $V_1^{\prec_g} \setminus V_1^{\prec_s}$. Then from Lemma 2 we have $V_g^0 = O$ and using Lemma 12 we obtain the constraints which together with an empty set A allow us to apply Lemma 13. Lemma 13 is constructed in such a way that whenever we can apply it to a (possibly empty) set A, it proves the existence of another set B such that that $|A| < |A \cup B|$ and Lemma 13 is applicable to the new set $A \cup B$. Thus it is possible to apply Lemma 13 infinitely many times and construct a subset of $V(G)$ containing infinitely many vertices. This leads to a contradiction as G is a finite graph □.

6.2 Reducing the Open Graph

The equality of penultimate layers of SSF and gflow might suggest that one could prove the equality of other layers simply by removing the last layer from the open graph and reapply the lemmas from the last section. However this would fail as the vertices in any layers can also use the output vertices in their correcting sets. Therefore we need to be careful which vertices we remove such that the reduced graph still have a gflow.

Definition 10 *If (G, I, O) is an open graph with flow $(f, \prec_f)$ and corresponding SSF $(s, \prec_s)$ then we call the open graph (G', I, O') a* reduced open graph according to $(s, \prec_s)$*, where*

- $R = \{v \in O \mid f^{-1}(v) \in V_1^{\prec_s}\}$ *is the set of removed vertices.*
- $G' = (V', E')$ *where*
  $$V' = V \setminus R$$
  $$E' = E \setminus (V \times R)$$
- $O' = (V_1^{\prec_s} \cup O) \setminus R$

We will omit "according to ..." and call (G', I, O') just reduced open graph when it is clear from the text which SSF is used for constructing it. An example of a reduced open graph is shown in Figure 10

As we saw in the previous section, we needed the fact that $|I| = |O|$ to be able to prove that the penultimate layers of SSF and maximally delayed gflow are equal. If we want to

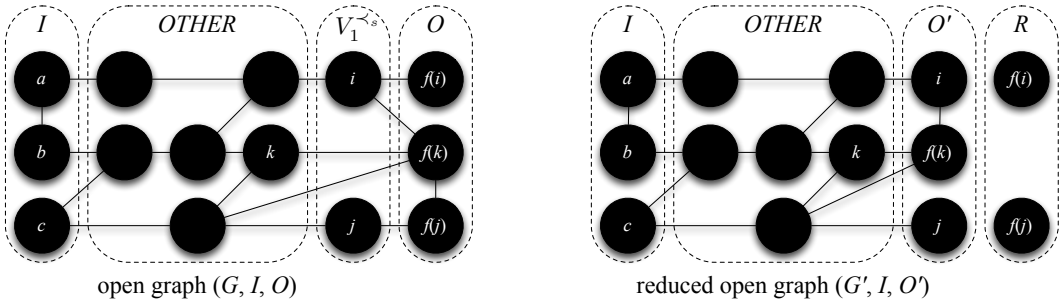


Fig. 10. An example of an SSF reduced open graph (right) together with the original open graph (left).

apply the same lemmas to the new reduced open graph, we need to guarantee that if we start with a graph where input size equals output size, the same holds for the reduced open graph.

Lemma 15 *Let (G', I, O') be a reduced open graph of the open graph (G, I, O), then $|O| = |O'|$.*

Proof. Let R be the set of vertices removed from G, then for every vertex $i \in V_1^{\prec_s}$ we have a corresponding unique vertex $f(i)$ in R since Proposition 3 implies that $s(i) \subseteq O$ and $f(i) \in s(i)$. On the other hand, for every vertex in R there exists a corresponding vertex in $V_1^{\prec_s}$ from the definition of R. Therefore for every vertex $v \in R$ that we remove from O when constructing $O' = (V_1^{\prec_s} \cup O) \setminus R$ we add another vertex $f^{-1}(v) \in V_1^{\prec_s}$ and it must hold that $|O| = |O'|$ $\square$.

The next lemma is used later to construct a gflow of the reduced open graph from the gflow of the original open graph.

Lemma 16 *Let (G, I, O) be an open graph and A and B two sets in O such that $Odd(B) \cap O^C = \emptyset$. Then $Odd((A \cup B) \setminus (A \cap B)) \cap O^C = Odd(A) \cap O^C$.*

Proof. Note that the symmetric difference of A and B is defined as $A \triangle B = (A \setminus B) \cup (B \setminus A)$ Since $Odd(B) \cap O^C = \emptyset$, every vertex in O^C has to be evenly connected to B. Thus there are altogether four different possibilities for a vertex $v \in O^C$ to be connected to the sets A and B satisfying $Odd(B) \cap O^C = \emptyset$ as shown in Figure 11:

$$v \in Even(A) \cap Odd(A \setminus B) \Rightarrow v \in Odd(A \cap B) \Rightarrow v \in Odd(B \setminus A) \Rightarrow v \in Even(A \triangle B)$$
$$v \in Even(A) \cap Even(A \setminus B) \Rightarrow v \in Even(A \cap B) \Rightarrow v \in Even(B \setminus A) \Rightarrow v \in Even(A \triangle B)$$
$$v \in Odd(A) \cap Odd(A \setminus B) \Rightarrow v \in Even(A \cap B) \Rightarrow v \in Even(B \setminus A) \Rightarrow v \in Odd(A \triangle B)$$
$$v \in Odd(A) \cap Even(A \setminus B) \Rightarrow v \in Odd(A \cap B) \Rightarrow v \in Odd(B \setminus A) \Rightarrow v \in Odd(A \triangle B)$$

We see that every time v is evenly connected to A it is also evenly connected to $(A \setminus B) \cup (B \setminus A)$ and every time v is oddly connected to A it is also oddly connected to $(A \setminus B) \cup (B \setminus A)$. Because $(A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B)$ and v is in O^C it must hold that $Odd((A \cup B) \setminus (A \cap B)) \cap O^C = Odd(A) \cap O^C$ $\square$.

We create a function that will later be proven to have the required properties of the gflow.

Lemma 17 (Finding the reduced gflow function) *Let (G, I, O) be an open graph with flow $(f, \prec_f)$, SSF $(s, \prec_s)$ and maximally delayed gflow $(g, \prec_g)$ such that $|I| = |O|$. Let*

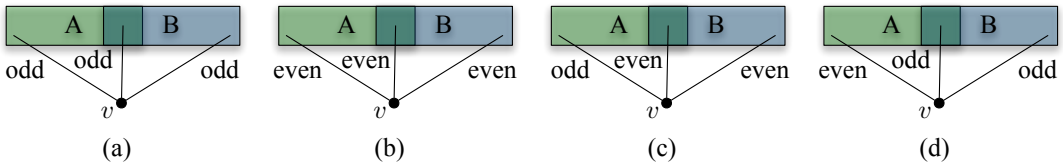


Fig. 11. The four possibilities for a vertex $v \in O^C$ to be connected connected to A, B and $A \cap B$.

(G', I, O') be the SSF reduced open graph of (G, I, O) with the removed vertices set R, then there exists a function $g' : O'^C \to P^{I^C \cap V(G')}$ such that:

1. $\forall i \in O'^C \quad g'(i) \cap O'^C = g(i) \cap O'^C$
2. $\forall i \in O'^C \quad Odd(g'(i)) \cap O'^C = Odd(g(i)) \cap O'^C$

Proof. We start by noting that according to Lemma 3 we can create an maximally delayed gflow $(\tilde{g}, \prec_{\tilde{g}})$ of the open graph $(G, I, O \cup V_1^{\prec_g}) = (G, I, O' \cup R)$ by restricting g to $V(G) \setminus (V_1^{\prec_g} \cup V_0^{\prec_g}) = O'^C$ and setting $\prec_{\tilde{g}} = \prec_g \setminus V_1^{\prec_g} \times V_0^{\prec_g}$. We construct our desired g' function from $\tilde{g}$.

We consider $i \in O'^C$, if there exists a vertex $j \in R \cap \tilde{g}(i)$ then from the reduced open graph definition we have $f^{-1}(j) \in V_1^{\prec_g}$. Also from Lemma 14 we have $V_1^{\prec_g} = V_1^{\prec_s}$ and thus $f^{-1}(j) \in V_1^{\prec_s}$. According to Proposition 3 this means that $s(f^{-1}(j)) \subseteq O$. We have $Odd(s(f^{-1}(j))) \cap O'^C = \emptyset$ since the only odd neighbours of $s(f^{-1}(j))$ are either output vertices or the vertex $f^{-1}(j) \in V_1^{\prec_g} \subseteq O'$.

Now we define $g'(i) = (\tilde{g}(i) \cup s(f^{-1}(j))) \setminus (\tilde{g}(i) \cap s(f^{-1}(j)))$, hence $j \notin g'(i)$. Moreover Lemma 16 implies that $Odd(g'(i)) \cap O'^C = Odd(\tilde{g}(i)) \cap O'^C$. Also $g'(i) \cap O'^C = \tilde{g}(i) \cap O'^C$ since $s(f^{-1}(j)) \subseteq O'$. Note that, since the new set will be constructed via a union of two sets we might add another vertex $k \in R$ to the set $g'(i)$. However, we can remove any such vertex k added to $g'(i)$ by applying the same procedure recursively. For every such vertex k, it must hold that $f^{-1}(j) \prec_f f^{-1}(k)$ since $k \in s(f^{-1}(j))$ and Proposition 3 implies the existence of a Z-path from $f^{-1}(j)$ to $f^{-1}(k)$. Now we remove k via the above procedure *i.e.* defining $g'(i) = (g'(i) \cup s(f^{-1}(k))) \setminus (g'(i) \cap s(f^{-1}(k)))$. If this would add vertex j again to $g'(i)$, hence there exists a Z-path from $f^{-1}(k)$ to $f^{-1}(j)$ and $f^{-1}(k) \prec_f f^{-1}(j)$ which contradicts the previous relation. This procedure will eventually terminate and remove all undesired vertices $k \in R$ since in the above procedure we never create any Z-path loops $\square$.

We call a function which satisfies properties (1) and (2) of Lemma 17 the *reduced gflow function* of g. We can interpret these properties as saying that the gflow function g' differs from the gflow function g only by the vertices in O', *i.e.* the other elements in the correcting set are left unchanged. As a gflow consists of a function and a partial order, we still need to define a valid partial order. The one that is most useful to us is such that it preserves as much relations as possible from the original gflow, hence the layering structures remain similar.

Lemma 18 (Constructing the reduced gflow) *Let (G, I, O) be an open graph with SSF $(s, \prec_s)$, gflow $(g, \prec_g)$ and (G', I, O') a reduced open graph of (G, I, O). If g' is a reduced gflow*

function of $(g, \prec_g)$*, then* $(g', \prec_{g'})$ *is a gflow of* (G', I, O')*, where*

$$\begin{aligned} &\forall i, j \in O'^C \quad i \prec_g j \Leftrightarrow i \prec_{g'} j \\ &\forall i \in O'^C, \forall j \in O' \cap g'(i) \quad \Rightarrow \quad i \prec_{g'} j \end{aligned}$$

Proof. We will show that $(g', \prec_{g'})$ satisfies the three gflow properties (G1) to (G3) in Definition 3. First property requires that if $j \in g'(i)$, then $i \prec_{g'} j$. This is obviously true if $j \in O'$. If $j \in g'(i) \cap O'^C$, from Lemma 17 we have $j \in g(i)$ which implies that $i \prec_g j$ because $(g, \prec_g)$ is a gflow. Now according the definition of $\prec_{g'}$ it must also hold, that $i \prec_{g'} j$.

Now we consider the gflow property (G2). For every $j \in Odd(g'(i))$ it must be that $j = i$ or $i \prec_{g'} j$. If $j \in O'$, then again this is obviously true because of the definition of $\prec_{g'}$. If $j \in Odd(g'(i)) \cap O'^C$ then we know that $j \in Odd(g(i))$ and $j = i$ or $i \prec_g j$. According to the definition of $\prec_{g'}$, $i \prec_g j$ implies that $i \prec_{g'} j$ and we have that if $j \in Odd(g'(i))$ then either $i = j$ or $i \prec_{g'} j$. Thus the gflow property (G2) is satisfied.

Finally, we require for gflow property (G3) that $i \in Odd(g'(i))$ and as $i \in O'^C$ this is true because of the properties of g' $\square$.

We call the gflow $(g', \prec_{g'})$ from Lemma 18 the *reduced gflow* of $(g, \prec_g)$. Similarly we can construct the SSF of the reduced open graph.

Lemma 19 (Constructing the reduced SSF) *Let* (G, I, O) *be an open graph with flow* $(f, \prec_f)$ *and SSF* $(s, \prec_s)$*. If* (G', I, O') *is the reduced open graph, according to* $(s, \prec_s)$*, then there exists an SSF* $(s', \prec_{s'})$ *of* (G', I, O') *such that* $(s', \prec_{s'})$ *is the reduced gflow of* $(s, \prec_s)$*.*

Proof. For a SSF to exist on (G', I, O'), it has to have flow. It can be shown that $(f', \prec_{f'})$, where

- $\forall i \in O'^C \quad f'(i) = f(i)$
- $\prec_{f'} = \prec_f \setminus [(V \times R) \cup (V_1^{\prec_s} \times O)]$,

satisfies all the properties in Def. 2 and is thus a flow on (G', I, O'). Define $(s', \prec_{s'})$ to be the SSF derived from this flow. Let R be the set of vertices removed from (G, I, O) to get (G', I, O'). Assume $(s', \prec_{s'})$ is not a reduced gflow of $(s, \prec_s)$, then one of the properties of Lemma 17 should not hold, We show a contradiction in both cases. If the first property does not hold then

$$\begin{aligned} &\exists i \in O'^C \quad s.t. \quad s'(i) \cap O'^C \neq s(i) \cap O'^C \Rightarrow \\ &\exists j \in O'^C \cap [(s(i) \setminus s'(i)) \cup (s'(i) \setminus s(i))] \Rightarrow \\ &(\zeta_i^s(f^{-1}(j)) \bmod 2 = 1 \wedge \zeta_i^{s'}(f^{-1}(j)) \bmod 2 = 0) \vee \\ &(\zeta_i^{s'}(f^{-1}(j)) \bmod 2 = 1 \wedge \zeta_i^g(f^{-1}(j)) \bmod 2 = 0) \Rightarrow \\ &\zeta_i^s(j) \bmod 2 \neq \zeta_i^{s'}(j) \bmod 2 \end{aligned}$$

Hence by removing vertices and edges from the open graph (G, I, O) we must have changed $\zeta_i^s(f^{-1}(j))$ by an odd number to get $\zeta_i^{s'}(f^{-1}(j))$.

Next we look at how removing the vertices in R from the open graph (G, I, O) changes $\zeta_i^s(f^{-1}(j))$. Removing a vertex v changes the number of Z-paths from i to $f^{-1}(j)$ if by removing it we also remove an edge in the Z-correction graph G_Z. Let this removed edge

be (k, l), then Corollary 1 implies that $l \in N(f(k))$ and v has to be either k, l or $f(k)$. Corollary 1 also implies that for v to have an outgoing edge in G_Z, $f(v)$ has to be defined. Since f is not defined for output vertices and $v \in R \subseteq O$, there cannot be any outgoing edges from v. Therefore v cannot be k as there is an edge from k to l in G_Z. Also v cannot be l since again v cannot have an outgoing edge in G_Z, hence v would have to be the last element on the Z-path from i to $f^{-1}(j)$, which is $f^{-1}(j)$. This is impossible, as $f^{-1}(j)$ cannot be an output vertex. Therefore the only possibility is that $v = f(k)$.

Let v be the first vertex removed from G, such that $\zeta_i^s(f^{-1}(j))$ changes by an odd number. Hence all the paths from i to $f^{-1}(j)$ that disappear due to removal of v have to go through $f^{-1}(v)$. Therefore there must also exist an odd number of paths from $f^{-1}(v)$ to $f^{-1}(j)$. We know that because of Proposition 3, $j \in s(f^{-1}(v))$ and $f^{-1}(v) \prec_s j$. On the other hand because of Definition 10 it must also hold that $f^{-1}(v) \in V_1^{\prec_s}$, which together with Definition 4 implies that $j \in V_0^{\prec_s} = O$. This leads to a contradiction, because j has to be in $O'^C \subseteq O^C$ and cannot be in O. Therefore property (1) must be true for $(s', \prec_{s'})$.

Now we show that property (2) has to hold. According to Lemma 6:

$$Odd(s'(i)) \cap O'^C = \{i\} = Odd(s(i)) \cap O^C$$

Because $i \in O'^C \subseteq O^C$, it must also hold that

$$Odd(s'(i)) \cap O'^C = \{i\} = Odd(s(i)) \cap O'^C$$

and property (2) has to be true for $(s', \prec_{s'})$ $\square$.

In the specific case where the input size of an open graph equals the output size the flow will be unique [11] and because of Proposition 3 the SSF will also be unique. Lemma 15 implies that if $|I| = |O|$, then $|I| = |O'|$ and combining this with Lemma 19 we can say that the unique SSF of a reduced open graph is the reduced gflow of the original SSF.

6.3 Moving Back

We have proven that the penultimate layers of SSF and maximally delayed gflow are equal if $|I| = |O|$. Then we showed how to remove some vertices from the open graph and construct an SSF and maximally delayed gflow on the new reduced graph. Both of them are reduced gflows, a property which we will use in this section to show that they preserve the layering of the gflows they were derived from.

Lemma 20 *Let (G, I, O) be an open graph with SSF $(s, \prec_s)$ and gflow $(g, \prec_g)$ such that (G', I, O') is the reduced open graph of (G, I, O) with the removed vertices set R. For every reduced gflow $(g', \prec_{g'})$ of (G', I, O') such that $V_0^{\prec_s} = V_0^{\prec_g} = O$ and $V_1^{\prec_s} = V_1^{\prec_g}$ it must hold that*

$$\forall n > 0 \quad V_n^{\prec_{g'}} = V_{n+1}^{\prec_g}$$

Proof. The proof consists of two parts, first we show that

$$\forall n \geq 0 \quad \cup_{k=0}^{n} V_k^{\prec_{g'}} = \cup_{k=0}^{n+1} V_k^{\prec_g} \setminus R. \tag{10}$$

This is done by induction, showing first that Eq. 10 holds if $n = 0$, *i.e.* we need to prove that

$$V_0^{\prec_{g'}} = (V_1^{\prec_g} \cup V_0^{\prec_g}) \setminus R$$

Lemma 2 tells that $V_0^{\prec_{g'}} = O'$ and $V_0^{\prec_g} = O$. Because the penultimate layers of SSF $(s, \prec_s)$ and gflow $(g, \prec_g)$ are equal we also have that $V_1^{\prec_s} = V_1^{\prec_g}$. Now we take the definition of O' from Definition 10 of the reduced open graph and substitute the appropriate sets:

$$O' = (V_1^{\prec_s} \cup O) \setminus R \quad \Rightarrow V_0^{\prec_{g'}} = (V_1^{\prec_g} \cup V_0^{\prec_g}) \setminus R$$

Thus Eq. 10 holds for $n = 0$. For the induction step we assume that Eq. 10 holds for $n = m-1$, *i.e*

$$\cup_{k=0}^{m-1} V_k^{\prec_{g'}} = \cup_{k=0}^{m} V_k^{\prec_g} \setminus R \tag{11}$$

and show that it holds for $n = m$. We use contradiction and assume that

$$\cup_{k=0}^{m} V_k^{\prec_{g'}} \neq \cup_{k=0}^{m+1} V_k^{\prec_g} \setminus R$$

There are two possibilities: either $\exists i \in V_m^{\prec_{g'}} \setminus V_{m+1}^{\prec_g}$ or $\exists i \in V_{m+1}^{\prec_g} \setminus V_m^{\prec'_g}$. We note that according to Lemma 18 $i \prec_g j \Leftrightarrow i \prec_{g'} j$ if $i \in O'^C$ and $j \in O'^C$. Because $V_m^{\prec_{g'}} \subseteq O'^C$ for every $m > 0$ we have that

$$\begin{aligned}
\exists i \in V_m^{\prec_{g'}} \setminus V_{m+1}^{\prec_g} \Rightarrow \quad & \exists j \in V_{m+1}^{\prec_g} \cap g'(i) \quad s.t. \quad i \prec_g j \quad \Rightarrow \\
\Rightarrow \quad & i \prec_{g'} j \quad \Rightarrow \quad j \in \cup_{k=0}^{m-1} V_k^{\prec_{g'}} = \cup_{k=0}^{m} V_k^{\prec_g} \\
\exists i \in V_{m+1}^{\prec_g} \setminus V_m^{\prec_{g'}} \quad \Rightarrow \quad & \exists j \in V_m^{\prec_{g'}} \cap g(i) \quad s.t. \quad i \prec_{g'} j \quad \Rightarrow \\
\Rightarrow \quad & i \prec_g j \quad \Rightarrow \quad j \in \cup_{k=0}^{m} V_k^{\prec_g} = \cup_{k=0}^{m-1} V_k^{\prec_{g'}}
\end{aligned}$$

As can be seen above, both of the possible cases lead to a contradiction and hence it must hold that

$$\cup_{k=0}^{m} V_k^{\prec_{g'}} = \cup_{k=0}^{m+1} V_k^{\prec_g} \setminus R$$

This completes the induction step and proves that Eq. 10 holds. Now we have that if $n > 0$

$$\cup_{k=0}^{n} V_k^{\prec_{g'}} = \cup_{k=0}^{n+1} V_k^{\prec_g} \setminus R \quad \text{and} \quad \cup_{k=0}^{n-1} V_k^{\prec_{g'}} = \cup_{k=0}^{n} V_k^{\prec_g} \setminus R.$$

We can now subtract the elements of the second set from the first.

$$\cup_{k=0}^{n} V_k^{\prec_{g'}} \setminus \cup_{k=0}^{n-1} V_k^{\prec_{g'}} = (\cup_{k=0}^{n+1} V_k^{\prec_g} \setminus R) \setminus (\cup_{k=0}^{n} V_k^{\prec_g} \setminus R) \quad \Rightarrow \quad V_n^{\prec_{g'}} = V_{n+1}^{\prec_g} \setminus R.$$

Because of Def. 10 of the SSF reduced open graph we know that $R \subseteq O$. Lemma 2 says that $O = V_0^{\prec_g}$, hence we know that no element in R can be included in $V_{n+1}^{\prec_g}$ for $n \geq 0$ and $V_n^{\prec_{g'}} = V_{n+1}^{\prec_g}$ $\square$.

It turns out, that if the gflow we have for the original open graph is the maximally delayed one, then the reduced gflow will be maximally delayed for the reduced open graph.

Lemma 21 (Constructing the reduced maximally delayed gflow) *Let (G, I, O) be an open graph with SSF $(s, \prec_s)$ and maximally delayed gflow $(g, \prec_g)$. If (G', I, O') is the reduced open graph of (G, I, O) then the reduced gflow $(g', \prec_{g'})$ of $(g, \prec_g)$ is the maximally delayed gflow of (G', I, O').*

Proof. First, because of Lemma, 18 $(g', \prec g')$ has to be a gflow of (G', I, O'). Let $(d, \prec_d) \neq (g', \prec_{g'})$ be the maximally delayed gflow of (G', I, O'). Then according to Definition 6

$$\exists n > 0, i \in O'^C \quad s.t. \quad i \in V_n^{\prec_d} \setminus V_n^{\prec_{g'}} \quad \wedge \quad \forall k < n \quad V_k^{\prec_{g'}} = V_k^{\prec_d}$$

and since $d(i) \in \cup_{k=0}^{n-1} V_k^{\prec_d} = \cup_{k=0}^{n-1} V_k^{\prec_{g'}}$, from Lemma 20 we obtain $d(i) \in \cup_{k=0}^{n} V_k^{\prec_g} \setminus R$, where the R is the set of vertices removed from the original graph. Definition 6 states that i is in $V_{n+1}^{\prec_g}$ which according to Lemma 20 must be equal to $V_n^{\prec_{g'}}$. This leads to a contradiction because $i \in V_n^{\prec_d} \setminus V_n^{\prec_{g'}}$ and thus $(g', \prec_{g'})$ has to be the maximally delayed gflow of (G', I, O') □.

6.4 Proof of the Optimality Theorem

We can now prove Theorem 2 by showing that the vertex layering of any SSF and an maximally delayed gflow is exactly the same. Let (G, I, O) be an open graph with flow $(f, \prec_f)$ such that $|I| = |O|$. Let $(s, \prec_s)$ be the SSF obtained from $(f, \prec_f)$ according to Proposition 3 and $(g, \prec_g)$ the maximally delayed gflow of (G, I, O). According to Proposition 3 the last layer of any SSF is the set of output vertices. Lemma 2 says that this is also true for the last layer of an maximally delayed gflow, therefore $V_0^{\prec_s} = V_0^{\prec_g} = O$. The layers $V_1^{\prec_s}$ and $V_1^{\prec_g}$ are equal because of Lemma 14.

Now we need to show that layers $V_n^{\prec_s}$ and $V_n^{\prec_g}$ are equal for $n > 1$. We can construct a reduced open graph (Definition 10) (G', I, O') from (G, I, O). We now consider the unique SSF $(s', \prec_{s'})$ and reduced gflow $(g', \prec_{g'})$ of (G', I, O'), which according to Lemma 21 is maximally delayed. According to Lemma 20, $V_n^{\prec_{g'}} = V_{n+1}^{\prec_g}$ for every $n > 0$ and because SSF is by Theorem 1 a gflow the same lemma also implies that $V_n^{\prec_{s'}} = V_{n+1}^{\prec_s}$. Because of the way a reduced open graph is defined, we know that $|I| = |O'|$ (Lemma 15). Thus we can again use Lemma 14 to say that $V_2^{\prec_g} = V_1^{\prec_{g'}} = V_1^{\prec_{s'}} = V_2^{\prec_s}$. We can now take (G', I, O') and find its reduced open graph to show using the same technique that $V_3^{\prec_g} = V_3^{\prec_s}$. This can be continued until we reach the empty layers, in which case we have considered all the layers according to $(s, \prec_s)$ and $(g, \prec_g)$. As every layer of $(s, \prec_s)$ and $(g, \prec_g)$ will be equal and $(g, \prec_g)$ is the maximally delayed gflow, the SSF of a flow of an open graph (G, I, O) is an maximally delayed gflow if $|I| = |O|$, which proves Theorem 2.

The above theorem together with Algorithm 1 for signal shifting implies a new method for finding maximally delayed flows on graphs with flow. In particular, if an open graph has flow and its input size equals output size, the maximally delayed gflow can be found in $O(n^3)$ steps. This is more efficient in the number of operations than previous best known algorithm which completes in $O(n^4)$ steps [3], but can find the maximally delayed gflow (if it exists) of arbitrary open graphs.

Corollary 3 *The maximally delayed gflow on an open graph (G, I, O) with flow $(f, \prec_f)$ and $|I| = |O|$ can be found in $O(n^3)$ computational steps, where n is the number of vertices in G.*

Proof. We prove that the following process gives us the maximally delayed gflow of (G, I, O) in $O(n^3)$ steps, by showing that each step in the process takes at most $O(n^3)$ elementary operations.

1. Find the unique flow $(f, \prec_f)$ using $O(n^2)$ operations.
2. Create the flow pattern (Equation 2) of $(f, \prec_f)$ using $O(n^2)$ operations.

3. Signal shift the resulting pattern using $O(n^3)$ operations.

4. Construct the SSF from the signal shifted pattern suing $O(n^2)$ operations.

Step (1) can be completed using $O(n^2)$ operations with the algorithm from [3] and Step (3) takes no more than $O(n^3)$ operations if Algorithm 1 is used. To estimate the number of operations required for steps (2 and 4) we first estimate the number of commands in a gflow pattern, since flow is a stricter version of gflow this upper bound holds also for flow.

- One preparation command N_i for every vertex $i \in I^C$ — total of $O(n)$ preparation commands.
- One entanglement command $E_{i,j}$ for for every edge $E_{i,j} \in E(G)$ — total of $O(n^2)$ entanglement commands.
- One measurement command $M_i^{\alpha_i}$ per vertex $i \in V(G)$ — total of $O(n)$ measurement commands.
- Up to n X- and Z-correction commands per vertex in $V(G)$ — total of $O(n^2)$ correction commands.

The non-correction commands for the pattern in flow Step (2) can be created by processing the open graph and the correction commands by parsing a description of the flow. Since the open graph has at most $O(n^2)$ elements (n vertices and $O(n^2)$ edges) the non correction commands can be created in $O(n^2)$ steps. The measurement commands should be added to the pattern respecting the flow partial ordering. This can be done in $O(n \log n)$ steps by first sorting the vertices using any reasonable sorting algorithm. After the non-correction commands are added to the pattern, the correction commands can be inserted before the measurement command acting on the vertex they act on. These can be read from the description of the flow (Equation 2) and since there is a total of $O(n^2)$ of them the whole Step (2) takes $o(n^2)$ operations.

The gflow $(g, \prec_g)$ in Step (4) can be created with one read over the measurement pattern (Equation 9) resulting from Step (3). This can be done by taking $j \in s(i)$ iff there exists an $X_j^{s_i}$ command in the pattern and $i \prec_g j$ iff there exists a $X_j^{s_i}$ or $Z_j^{s_i}$ command in the pattern. Since there are at most $O(n^2)$ commands in the pattern, the gflow can be constructed in $O(n^2)$ operations. Finally, because of Theorem 2 the constructed SSF is an maximally delayed gflow □.

7 Conclusion

Initially MBQC was proposed as an alternative architecture for the implementation of quantum computing. However, from early on the distinct parallel power of the model attracted researchers to explore further this unique feature of the model. In a series of results the key concepts of flow, signal shifted flow, gflow, maximally delayed gflow, focused gflow and information preserving flow were introduced [1, 4, 2, 3, 29]. They address the general question of determinism in MBQC while shedding light on the parallelism as well. Although it is now known that the MBQC parallel power is equivalent to the quantum circuit with unbounded fanout [6], further investigation is required to fully take advantage of this extra power. In

this paper we continue this line of research by presenting a surprising link between signal shifted flow and maximally delayed gflow. The surprise comes from the fact that the former is obtained via a simple rewrite rules of pushing the Z-dependencies of a pattern to the end of the computation, while the latter is constructed directly from the structure of the underlying graph. This leads to a new efficient procedure for finding the maximally delayed gflow of graphs with flow, as we discussed in the last section. It remains an open question for which flows exactly our algorithm outputs an maximally delayed gflow if $|I| \neq |O|$.

Moreover the link between signal shifted flow and maximally delayed gflow opens a new direction to unify further the "flow" structure and fully characterise the constructions behind the parallel power of MBQC. The techniques presented in this work have already been used to translate computations in the 1WQC model to circuit model without increasing the depth of the computation and introducing any auxiliary qubits [9]. Further application of the different techniques introduced in this paper to known quantum algorithms, as well as a full comparison with other known optimisation methods beyond MBQC, constitute an interesting subject to be explored in the future.

References

1. Vincent Danos and Elham Kashefi. Determinism in the one-way model. *Physical Review A*, 74(5):052310, 2006.
2. Daniel Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9:250, August 2007.
3. Mehdi Mhalla and Simon Perdrix. Finding Optimal Flows Efficiently. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I (ICALP'08)*, pages 857–868, Berlin, Heidelberg, 2008.
4. Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *Journal of the ACM*, 54(2), April 2007.
5. Anne Broadbent and Elham Kashefi. Parallelizing quantum circuits. *Theoretical Computer Science*, 410(26):2489–2510, 2009.
6. Dan E Browne, Elham Kashefi, and Simon Perdrix. Computational depth complexity of measurement-based quantum computation. In *Proceeding of the Fifth Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC'10)*, pages 35–46, December 2010.
7. Ross Duncan and Simon Perdrix. Rewriting Measurement-Based Quantum Computations with Generalised Flow. *Automata, Languages and Programming*, pages 285–296, 2010.
8. Raphael Dias da Silva and Ernesto F Galvão. Compact quantum circuits from one-way quantum computation. *Physical Review A*, 88(1):012319, July 2013.
9. Jisho Miyazaki, Michal Hajdušek, and Mio Murao. Translating measurement-based quantum computation with gflow into quantum circuit. *arXiv:1310.4043v2*, November 2013.
10. Damian Markham and Elham Kashefi. Entanglement, Flow and Classical Simulatability in Measurement Based Quantum Computation. *arXiv:13113610v1*, November 2013.
11. Niel de Beaudrap. Finding flows in the one-way measurement model. *Physical Review A*, 77(2):022328, January 2008.
12. Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal Blind Quantum Computation. *IEEE Annual Symposium on Foundations of Computer Science*, 0:517–526, 2009.
13. Elham Kashefi, Damian Markham, Mehdi Mhalla, and Simon Perdrix. Information Flow in Secret Sharing Protocols. *Electronic Proceedings in Theoretical Computer Science*, 9:87–97, November 2009.
14. Bobby Antonio, Damian Markham, and Janet Anders. Trade-off between computation time and

Hamiltonian degree in adiabatic graph-state quantum computation. *arXiv:13091443v1*, September 2013.
15. Theodoros Kapourniotis, Elham Kashefi, and Animesh Datta. Verified Delegated Quantum Computing with One Pure Qubit. *arXiv:1403.1438v2*, March 2014.
16. Daniel Gottesman and Isaac Chuang. Quantum Teleportation is a Universal Computational Primitive. *Nature*, 402:390–393, 1999.
17. Michael Nielsen. Universal quantum computation using only projective measurement, quantum memory, and preparation of the 0 state. *Physics Letters A*, 2-3(308):96–100, February 2003.
18. Debbie Leung. Two-qubit Projective Measurements are Universal for Quantum Computation. *arXiv:quant-ph/0111122v2*, 2002.
19. Susana F Huelga, Martin B Plenio, and Joan A Vaccaro. Remote control of restricted sets of operations: Teleportation of Angles. *Physical Review A*, 65(042316), 2002.
20. Susana F Huelga, Joan A Vaccaro, Anthony Chefles, and Martin B Plenio. Quantum Remote Control: Teleportation of Unitary Operations. *Physical Review A*, 63(042303):5, 2001.
21. Richard Jozsa. An introduction to measurement based quantum computation. *arXiv:quant-ph/0508124v2*, page 22, January 2006.
22. Robert Raussendorf, Hans Briegel, and Daniel Browne. The one-way quantum computer - a non-network model of quantum computation. *Journal of Modern Optics*, 49(8):1299–1306, 2002.
23. Robert Raussendorf, Daniel Browne, and Hans Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(022312), August 2003.
24. Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, Marteen Van den Nest, and Hans J Briegel. Entanglement in Graph States and its Applications. In *Proceedings of the International School of Physics "Enrico Fermi"*, pages 115–218, February 2006.
25. David Gross, Jens Eisert, and Steven T Flammia. Most quantum states are too entangled to be useful as computational resources. *Physical Review Letters*, 102(190501):5, 2009.
26. Maarten Van Den Nest, Akimasa Miyake, Wolfgang Dür, and Hans Briegel. Universal Resources for Measurement-Based Quantum Computation. *Physical Review Letters*, 97(15):150504, October 2006.
27. Caterina E Mora, Marco Piani, Akimasa Miyake, Marteen Van den Nest, Wolfgang Dür, and H J Briegel. Universal resources for approximate and stochastic measurement-based quantum computation. *Physical Review A*, 81(4):042315, April 2010.
28. Tzu-Chieh Wei, Ian Affleck, and Robert Raussendorf. Affleck-Kennedy-Lieb-Tasaki State on a Honeycomb Lattice is a Universal Quantum Computational Resource. *Physical Review Letters*, 106(7):070501, February 2011.
29. Mehdi Mhalla, Mio Murao, Simon Perdrix, Masato Someya, and Peter S Turner. Which graph states are useful for quantum information processing? In *Proceeding of the 6th Conference on the Theory of Quantum Computation, Communication and Cryptography*, 2010.