

# Finding Optimal Flows Efficiently

Mehdi Mhalla<sup>1</sup> and Simon Perdrix<sup>2</sup>

<sup>1</sup> LIG, University of Grenoble, France  
mehdi.mhalla@imag.fr

<sup>2</sup> Oxford University Computing Laboratory, UK  
simon.perdrix@comlab.ox.ac.uk

**Abstract.** Among the models of quantum computation, the One-way Quantum Computer [11,12] is one of the most promising proposals of physical realisation [13], and opens new perspectives for parallelisation by taking advantage of quantum entanglement [2]. Since a One-way QC is based on quantum measurement, which is a fundamentally nondeterministic evolution, a sufficient condition of global determinism has been introduced in [4] as the existence of a *causal flow* in a graph that underlies the computation. A  $O(n^3)$ -algorithm has been introduced [6] for finding such a causal flow when the numbers of output and input vertices in the graph are equal, otherwise no polynomial time algorithm was known for deciding whether a graph has a causal flow or not. Our main contribution is to introduce a  $O(m)$ -algorithm for finding a causal flow (where  $m$  is the number of edges of the graph), if any, whatever the numbers of input and output vertices are. This answers the open question stated by Danos and Kashefi [4] and by de Beaudrap [6]. Moreover, we prove that our algorithm produces a flow of minimal depth.

Whereas the existence of a causal flow is a sufficient condition for determinism, it is not a necessary condition. A weaker version of the causal flow, called *gflow* (generalised flow) has been introduced in [3] and has been proved to be a necessary and sufficient condition for a family of deterministic computations. Moreover the depth of the quantum computation is upper bounded by the depth of the gflow. However the existence of a polynomial time algorithm that finds a gflow has been stated as an open question in [3]. In this paper we answer this positively with a polynomial time algorithm that outputs an optimal gflow of a given graph and thus finds an optimal correction strategy to the nondeterministic evolution due to measurements.

## 1 Introduction

A one-way quantum computation [11] consists in performing a sequence of one-qubit measurements on an initial entangled quantum state. This initial state, described by a graph, is a graph state [8], where some vertices correspond to the input qubits of the computation, others to the output qubits and the rest of the vertices correspond to auxiliary qubits measured during the computation. Since quantum measurements are nondeterministic, a one-way quantum computation requires corrections, making the basis of some measurements dependent

on the classical outcome of some other measurements. This corrections induce a dependency between the measurements, affecting the depth of the computation.

Moreover, in order to be implemented, such corrections impose the structure of the graph states that can be used for deterministic computation. Indeed, for some graph states, not all the sequences of one-qubit measurements represent a unitary embedding. The measurement calculus [5] is a formal framework for one-way quantum computations, where the dependencies between measurements and corrections are precisely identified. Using this formalism, Danos and Kashefi in [4] have proved that any one-way quantum computation translated from a quantum circuit is such that its underlying graph satisfies a causal flow condition (see section 2.) As a consequence, the existence of a causal flow is a sufficient condition for determinism.

In [7] a polynomial time algorithm in the size of the graph has been proposed for finding a causal flow when the numbers of outputs and inputs are equal, whereas the existence of a polynomial time algorithm in the general case, has been stated as an open question. We propose in this paper a faster and more general algorithm for finding a causal flow, whatever the numbers of inputs and outputs are.

It turns out that the existence of a causal flow is not a necessary condition for determinism. Indeed, a weaker flow condition, the gflow condition has been introduced as a necessary and sufficient condition for a class of uniformly deterministic computations (see [3] for a formal definition.) Here, we introduce a polynomial time algorithm for finding a gflow and thus checking whether a graph allows a uniformly deterministic computation, which gives substantially more relevance to the notion of gflow introduced in [3].

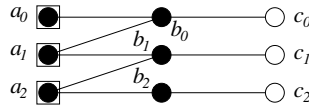
We also prove that the algorithms introduced in this paper produce minimal depth flows. This implies that the gflow algorithm gives a lower bound on the complexity of a correction strategy in a measurement-based setting for quantum computation.

## 2 Definitions

A graph with input and output vertices is called an open graph, and is defined as follows:

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

During a one-way quantum computation all non output qubits (represented as non output vertices in the corresponding open graph) are measured. Since quantum measurements are nondeterministic, for each qubit measurement, a correction consists in acting on some unmeasured non input qubits, depending on the classical outcome of the measurement, in order to make the computation deterministic. Thus, a correction strategy induces a sequential dependency between measurements. As a consequence, the depth of the quantum computation depends on the correction strategy.



**Fig. 1.** Example of open graph – squared vertices represent inputs, white vertices represent outputs – which has a causal flow  $(g, \prec)$ , where  $g(a_i) = b_i$ ,  $g(b_i) = c_i$  and  $a_0 \prec a_1 \prec a_2 \prec \{b_0, b_1, b_2\} \prec \{c_0, c_1, c_2\}$

Correction strategies are characterised by flows in open graphs. A flow  $(g, \prec)$  consists in a partial order  $\prec$  over the vertices ( $i \prec j$  if  $i$  is measured before  $j$ ) and a function  $g$  that associates with each vertex the vertices used for correcting its measurement (all non output qubits are measured.) Input qubits cannot be used for correction (see [5].)

Given an open graph, two kinds of flows are considered: the causal flow and the gflow (generalised flow.) The former has been introduced by Danos and Kashefi [4] and corresponds to the computation strategy that consists in correcting each qubit measurement by acting on a single neighbour of the measured qubit. For a given open graph, a causal flow is characterised by a function  $g$  which associates with each non output vertex a non input vertex used for the correction of its measurement. More formally:

**Definition 2 (causal flow).**  $(g, \prec)$  is a causal flow of  $(G, I, O)$ , where  $g : V(G) \setminus O \rightarrow V(G) \setminus I$  and  $\prec$  is a strict partial order over  $V(G)$ , if and only if

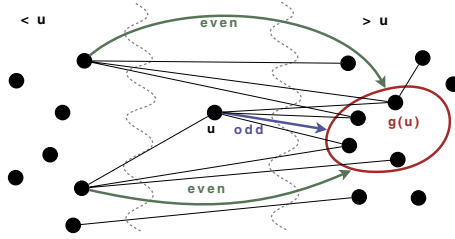
1.  $i \prec g(i)$
2. if  $j \in N(g(i))$  then  $j = i$  or  $i \prec j$ , where  $N(v)$  is the neighbourhood of  $v$
3.  $i \in N(g(i))$ .

An example of causal flow is given in Figure 1. Notice that if the numbers of input and output vertices are the same, a causal flow can be reduced to a path cover and then to a standard network flow [6]. This reduction was used to define an  $O(n^3)$ -algorithm for finding a causal flow in the case where the cardinalities of input and output qubits are the same [6].

The second type of flow considered, the generalised flow, *gflow*, has been introduced in [3] and corresponds to a more general correction strategy that associates with each non output vertex a set of vertices used for the corresponding correction (instead of a single vertex.) This generalisation not only leads to a reduction of the computational depth, but also provides a correction strategy to some open graphs which have no causal flow. Moreover, notice that the existence of a gflow is a necessary and sufficient condition for uniform, strong and stepwise deterministic computations [3].

For a given open graph, a gflow  $(g, \prec)$  is characterised by a function  $g$  which associates with each non output vertex, a set of non input vertices used for its correction, and a strict partial order  $\prec$ :

**Definition 3 (gflow).**  $(g, \prec)$  is a gflow of  $(G, I, O)$ , where  $g : V(G) \setminus O \rightarrow \wp(V(G) \setminus I)$  and  $\prec$  is a strict partial order over  $V(G)$ , if and only if



**Fig. 2.** Graphical interpretation of a gflow  $(g, \prec)$ : for a given vertex  $u$  all the vertices larger than  $u$  represent qubits that will be measured after the qubit  $u$ . The set  $g(u)$  has to be composed of qubits measured after  $u$ , and such that the following parity conditions are satisfied: there is an odd number of edges between  $g(u)$  and  $u$  and there is an even number of edges between  $g(u)$  and any vertex which is not larger  $u$ .

1. if  $j \in g(i)$  then  $i \prec j$
2. if  $j \in \text{Odd}(g(i))$  then  $j = i$  or  $i \prec j$
3.  $i \in \text{Odd}(g(i))$

Where  $\text{Odd}(K) = \{u, |N(u) \cap K| = 1 \pmod 2\}$  is the odd neighbourhood of  $K$ , i.e. the set of vertices which have an odd number of neighbours in  $K$ .

A graphical interpretation of the generalised flow is given in Figure 2.

A flow  $(g, \prec)$  of  $(G, I, O)$  induces a partition of the vertices of the open graph:

**Definition 4.** For a given open graph  $(G, I, O)$  and a given flow  $(g, \prec)$  of  $(G, I, O)$ , let

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

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

A causal flow or a gflow  $(g, \prec)$  of  $(G, I, O)$  leads to a correction strategy for the corresponding one-way quantum computation, which consists in measuring the non output qubits of each layer in parallel, from the layer  $V_{d^\prec}^\prec$  to the layer  $V_1^\prec$ . After the measurement of a layer  $V_k^\prec$ , with  $k > 0$ , corrections are realised according to the function  $g$  by acting on qubits in  $\cup_{i < k} V_i^\prec$  (see [3] for details.) The depth of such a one-way quantum computation is  $d^\prec$ .

**Definition 5.** For a given open graph  $(G, I, O)$  and two given causal flows (resp. gflows)  $(g, \prec)$  and  $(g', \prec')$  of  $(G, I, O)$ ,  $(g, \prec)$  is more delayed than  $(g', \prec')$  if  $\forall k, |\cup_{i=0 \dots k} V_k^\prec| \geq |\cup_{i=0 \dots k} V_k^{\prec'}|$  and there exists a  $k$  s.t. the inequality is strict. A causal flow (resp. gflow)  $(g, \prec)$  is maximally delayed if there exists no causal flow (resp. gflow) of the same open graph that is more delayed.

For instance, the flow  $(g, \prec)$  described in Figure 1 is a maximally delayed causal flow. However,  $(g, \prec)$  is not a maximally delayed gflow since  $(g', \prec')$  is a more

delayed gflow, where  $g'(a_0) = \{b_0, b_1, b_2\}, g'(a_1) = \{b_1, b_2\}, g'(a_2) = \{b_2\}, g'(b_0) = \{c_0\}, g'(b_1) = \{c_1\}, g'(b_2) = \{b_2\}$ , and  $\{a_0, a_1, a_2\} \prec' \{b_0, b_1, b_2\} \prec' \{c_0, c_1, c_2\}$ . One can prove that  $(g', \prec')$  is a maximally delayed gflow.

The following two lemmas are proved for both kinds of flows.

**Lemma 1.** *If  $(g, \prec)$  is a maximally delayed causal flow (resp. gflow) of  $(G, I, O)$  then  $V_0^\prec = O$ .*

*Proof.* Let  $(g, \prec)$  be a maximally delayed causal flow (resp. gflow) of  $(G, I, O)$ . Elements of  $V_0^\prec$  have no image under  $g$  because of condition 1 in both definitions thus  $V_0^\prec \subseteq O$ . Moreover, by contradiction, if  $O \setminus V_0^\prec \neq \emptyset$ , let  $\prec' = \prec \setminus (O \setminus V_0^\prec) \times V(G)$ .  $(g, \prec')$  is a causal flow (resp. gflow) of  $(G, I, O)$ : condition 1 of both definition is satisfied by  $\prec'$ , because the domain of  $g$  does not intersect  $O$ , so for any  $i$  in the domain of  $g, i \prec' j$  iff  $i \prec j$ ; conditions 2 and 3 of both definitions are satisfied in a same way. Thus,  $(g, \prec')$  is a causal flow (resp. gflow) of  $(G, I, O)$ . Moreover, for any  $k, \cup_{i=0\dots k} V_k^\prec \subseteq \cup_{i=0\dots k} V_k^{\prec'}$ , and  $|V_0^\prec| < |V_0^{\prec'}|$  thus  $(g, \prec')$  is more delayed than  $(g, \prec)$  which leads to a contradiction.  $\square$

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

*Proof.* First, one can prove that  $(\tilde{g}, \tilde{\prec})$  is a causal flow (resp. gflow) of  $(G, I, O \cup V_1^\prec)$ . Moreover, by contradiction, if there exists a causal flow (resp. gflow)  $(g', \prec')$  that is more delayed than  $(\tilde{g}, \tilde{\prec})$  then it could be extended to  $(g'', \prec'')$  where  $g''(u) = g'(u)$  if  $u \in V \setminus (V_0^\prec \cup V_1^\prec), g''(u) = g(u)$  if  $u \in V_1^\prec$  and  $\prec'' = \prec' \cup \{(u, v), u \in V_1^\prec \wedge u \prec v\}$ .  $(g'', \prec'')$  is then a more delayed causal flow (resp. gflow) of  $(G, I, O)$  than  $(g, \prec)$ , which leads to a contradiction.  $\square$

**Lemma 3.** *If  $(g, \prec)$  is a maximally delayed gflow, then  $V_1^\prec = \{u \in V \setminus O, \exists K \subseteq O, Odd(K) \cap (V \setminus O) = \{u\}\}$ .*

*Proof.* First, notice that if  $(g, \prec)$  is a maximally delayed gflow, then for any  $u \in V_1^\prec, g(u) \subseteq O$  since  $u \prec v$  if  $v \in g(u)$  (condition 1 of definition 3.) Furthermore, by definition of  $V_1^\prec$ , if  $u \prec v$  then  $v \in O$  thus conditions 2 and 3 of definition 3 imply that  $Odd(g(u)) \cap (V \setminus O) = \{u\}$ .

To prove that any  $u \in V \setminus O$  such that  $\exists K \subseteq O, Odd(K) \cap V \setminus O = \{u\}, u \in V_1^\prec$ , we proceed by contradiction. We prove that delaying the measurement of a vertex not in  $V_1^\prec$  satisfying the condition permits to create a more delayed gflow. Indeed, let  $(g, \prec)$  be a maximally delayed flow of  $(G, I, O)$  and let  $u_1 \in V \setminus V_0^\prec$  be such that  $\exists K \subseteq O, Odd(K) \cap V \setminus O = \{u_1\}$ . Let  $g'(u) = K$  if  $u = u_1$  and  $g'(u) = g(u)$  otherwise. Let  $\prec'$  be the strict partial order defined by  $u \prec' v$  if  $u \neq u_1$  and  $u \prec v$  or if  $u = u_1$  and  $v \in K$ . It leads to a contradiction since  $(g', \prec')$  is a more delayed gflow of  $(G, I, O)$  than  $(g, \prec)$ .  $\square$

In a similar way, one can prove that:

**Lemma 4.** *If  $(g, \prec)$  is a maximally delayed causal flow, then  $V_1^\prec = \{u \in V \setminus O, \exists v \in O, N(v) \cap V \setminus O = \{u\}\}$ .*

Lemmas 3 and 4 show that in a maximally delayed flow, all the elements that can be corrected at the last step are in the maximal layer of  $V \setminus O$  (i.e. in  $V_1^{\prec}$ .) Combined with the recursive structure of maximally delayed flow (lemma 2), this shows that the layers  $V_k^{\prec}$  of a maximally delayed flow can be iteratively constructed by finding elements that can be corrected starting from the output qubits. This gives rise to the polynomial time algorithms of the next sections.

### 3 Causal Flow Algorithm

The relation between causal flow and determinism is presented in [4]. The best known algorithm for finding a causal flow has been proposed in [6], and works only if the numbers of inputs and outputs are the same. The complexity of the algorithm is in  $O(nm)$  where  $n$  is the number of vertices and  $m$  the number of edges (more precisely  $O(km)$  where  $k$  is the number of inputs (outputs) [7].) We present here a more general and faster algorithm.

**Theorem 1.** *For a given open graph  $(G, I, O)$ , finding a causal flow can be done in  $O(m)$  operations where  $m = |E(G)|$  is the number of edges of the graph  $G$ .*

In order to prove Theorem 1, we introduce the algorithm 1 which decides whether a given open graph has a causal flow, and outputs a maximally delayed causal flow if one exists. This recursive algorithm is based on the recursive structure, pointed out in the previous section, of the maximally delayed causal flows. The algorithm recursively finds the layers  $(V_k^{\prec})_{k=0\dots d^{\prec}}$ : at the  $k^{th}$  call to **Flowaux**, the algorithm finds the set  $V_k^{\prec} = Out'$  (see algorithm 1 and figure 3.) To improve the complexity of the algorithm, a set  $C$  of potential correctors is maintained, we also maintain the number of non output neighbours of every output vertex that is not an input. The partial order  $\prec$  of the flow produced by the algorithm is defined via a labeling  $l$  which associates with each vertex, the index of its layer. As a consequence, for any two vertices  $u$  and  $v$ ,  $u \prec v$  iff  $l(u) > l(v)$ .

**Proof of Theorem 1:** The precondition for the call of **Flowaux** is:  $\{v \in Out \setminus In, |N(v) \cap (V \setminus Out)| = 1\} \subseteq C$  and  $\forall v \in Out \setminus In, Past(v) = |N(v) \cap (V \setminus Out)|$ .

By induction on the number of non output vertices, we prove that if the given open graph has a causal flow then the algorithm outputs a maximally delayed one. Assume that the given open graph has a causal flow. First, if there is no non output vertex, then no correction is needed: the empty flow  $(g, \emptyset)$  (where  $g$  is a function with an empty domain) is a maximally delayed gflow. Now suppose that there exist some non output vertices, according to Lemma 4 the elements of  $V_1^{\prec}$  have a neighbour in  $C$  and thus they are considered in the loop line 13. The test line 14 ensures that they are considered only once: if a vertex can be corrected by two vertices in  $C$  then only the first is considered. After the loop (line 27),  $Out := Out \cup V_1^{\prec}$ . At each modification of the output set (line 15) the preconditions are maintained. Indeed, the number of non output neighbours of this vertex is computed (line 17), and the number of non output neighbours of the potential correctors (vertices in  $In \setminus Out$ ) is updated (line 23). Notice that

```

input : An open graph
output: A causal flow
1 Flow ((V,N),In,Out) =
2 begin
3   C:=∅ ; Past:= 0n ; l := 0n ;
4   for all v ∈ Out \ In do
5     Past(v):=|N(v) ∩ (V \ Out)| ;
6     if Past(v) = 1 then C := C ∪ {v} ;
7   end
8   return Flowaux ((V,N),In,Out,C,Past,1);
9 end

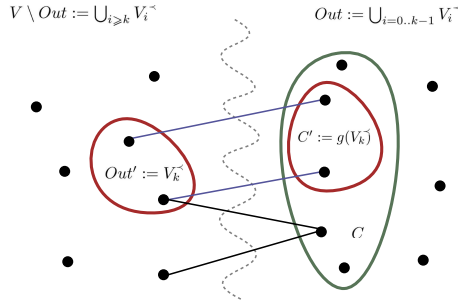
10 Flowaux ((V,N),In,Out,C,Past,k) = begin
11   C':=∅;
12   for all v ∈ C do
13     if |N(v) ∩ (V \ Out)| = 1 then
14       {u} := N(v) ∩ (V \ Out) ; g(u) := v ; l(u) := k ; Out := Out ∪ {u} ;
15       if u ∉ In then
16         Past(u) := |N(u) ∩ (V \ Out)| ;
17         if Past(u) = 1 then C' := C' ∪ {u} ;
18       end
19       for all w ∈ N(u) do
20         if Past(w) > 0 then
21           Past(w) := Past(w) - 1 ;
22           if Past(w) = 1 then C' := C' ∪ {w} ;
23         end
24       end
25     end
26   end
27   if C' = ∅ then
28     return (Out = V, (g, l)) ;
29   else
30     return Flowaux ((V,N),In,Out,C',Past,k + 1) ;
31   end
32 end

```

**Algorithm 1.** Causal flow

after the insertion of  $u$  in  $Out$ , the vertices that may become a potential corrector for the next recursive call are  $u$  and some of the neighbours of  $u$  (loop line 20). Thus,  $C'$  satisfies the precondition  $\{v \in Out \setminus In, |N(v) \cap (V \setminus Out)| = 1\} \subseteq C'$  for the recursive call (line 31).

Since the existence of a causal flow is assumed,  $V_1^{\prec}$  cannot be empty (every output vertex has to be corrected), thus the algorithm is called recursively. Lemma 2 ensures the existence of a causal flow in  $(G, I, O \cup V_1^{\prec})$ . The induction hypothesis ensures that the recursive calls output a maximally delayed causal



**Fig. 3.** Causal flow algorithm: At the  $k^{th}$  recursive call, the algorithm finds out the set  $V_k^{\prec}$  composed of the qubits that will be measured at the  $d^{\prec} - k + 1$  step of the one-way quantum computation, where  $d^{\prec}$  is the depth of the computation. At that step, all the qubits in  $Out := \bigcup_{i=0..k-1} V_i^{\prec}$  are not measured, whereas the qubits in  $\bigcup_{i > k} V_i^{\prec}$  are already measured. The correctors of the elements of  $V_k^{\prec}$  are in a set  $C \subseteq Out \setminus In$  of candidates composed of vertices  $u$  not already assigned to the correction of some future measurements ( $C = \{v, Past(v) \geq 1\}$ ). Since at each step, a maximum number of vertices are added to  $V_k^{\prec}$ , the causal flow, if it exists, produced by this algorithm is maximally delayed.

flow in  $(G, I, O \cup V_1^{\prec})$  and thus the produced causal flow  $(g, \prec)$  is a maximally delayed causal flow of  $(G, I, O)$ .

The termination of the algorithm is ensured by the fact that the set of output vertices strictly increases at each recursive call.

For a given open graph, if the algorithm outputs  $(true, (g, \prec))$ , then  $(g, \prec)$  is a valid causal flow since every output vertex has an image under  $g$ , moreover for any vertex  $i$ ,  $i \prec g(i)$ , and finally if  $j \in N(g(i))$  then  $j = i$  or  $i \prec j$ . Thus, if the given open graph has no flow, the algorithm returns false.

For the analysis of the complexity of the algorithm, we consider that testing whether a vertex is an input (resp. output) can be done in constant time. This can be achieved with an additional cost of  $n + |In|$  (resp.  $2n + |Out|$ ) by building a boolean array of size  $n$ . The additional  $n$  for the outputs comes from the fact that the output set has to be maintained as it changes during the algorithm. Each vertex  $v$  is inserted at most once in a set  $C$ . The cost associated with the vertex  $v$  consists in

- Finding the predecessor by  $g: \{u\} = N(v) \cap (V \setminus Out)$  which costs the degree  $\delta(v)$  of  $v$  assuming that the graph is given as an adjacency list ;
- Computing  $Past(v) = |N(v) \cap (V \setminus Out)|$  when inserting one of its neighbours in  $C$  which also costs  $\delta(v)$  ;
- Adding it to  $Out$  and  $C$  which has a constant cost ;
- Decreasing  $Past(v)$  and testing if  $Past(v) = 1$  which occurs at most  $\delta(v)$  times.

Thus, the total cost of the algorithm is upper bounded by  $O(\sum \delta(v)) = O(m)$ .  $\square$

This result improves the algorithm in [6] that decides, under the precondition  $|I| = |O|$ , whether an open graph  $(G, I, O)$  has a causal flow in  $O(km)$  operations,

where  $k = |O|$ . In [10], Pei and de Beaudrap have proved that an open graph which has a causal flow has at most  $(n - 1)k - \binom{k}{2}$  edges. According to this result, the algorithm in [6] can be transformed (see [10]) into a  $O(k^2n)$ -algorithm, whereas our algorithm becomes a  $O(\min(m, kn))$ -algorithm.

### 4 A Polynomial Algorithm for Gflow

In this section, we prove that the ideas of the algorithm 1 can be extended to derive a polynomial time algorithm in the more general case of the gflow, where each measurement is corrected by a set of qubits, instead of a single qubit. Since the existence of such a corrective strategy is sufficient and necessary for a large family of deterministic computations, the following algorithm decides whether a given one-way quantum computation is a member of such family of deterministic computation.

**Theorem 2.** *There exists a polynomial time algorithm that decides whether a given open graph has a gflow, and which outputs a gflow if it exists.*

*Proof.* In order to prove Theorem 2, we introduce a polynomial time algorithm which decides whether a given open graph has a gflow. Moreover, if a gflow exists, the algorithm outputs a maximally delayed gflow.

Let  $(G, I, O)$  be an open graph. The algorithm **gFlow** $(G, I, O)$  (Algorithm 2), where  $\Gamma$  is the adjacency matrix of  $G$ , finds a maximally delayed gflow  $(g, \prec)$  and returns  $(true, (g, \prec))$  if one exists and returns *false* otherwise. Given a set  $Y$  and a subset  $X \subseteq Y$ ,  $\mathbb{I}_X$  stands for a  $|Y|$ -dimensional vector defined by  $\mathbb{I}_X(i) = 1$  if  $i \in X$  and  $\mathbb{I}_X(i) = 0$  otherwise.

At the  $k^{th}$  recursive call, the set  $C$  found by the algorithm at the end of the loop at line 16 corresponds to the layer  $V_k^{\prec}$  of the partition induced by the returned strict partial order. At line 12, the columns of the sub-matrix  $\Gamma_{V \setminus Out, Out \setminus In}$  correspond to the vertices that can be used for correction (vertices in  $\cup_{i < k} V_i^{\prec} \setminus In$ ) and the rows to the candidates for the set  $V_k^{\prec}$ . A solution  $X_0$  in  $\mathbb{F}_2$  to  $\Gamma_{V \setminus Out, Out \setminus In} \mathbb{I}_X = \mathbb{I}_{\{u\}}$  is a subset of  $\cup_{i < k} V_i^{\prec} \setminus In$  that has only  $u$  as odd neighbourhood in  $\cup_{i \geq k} V_i^{\prec}$ , thus  $g(u) := X_0$  satisfies conditions 2 and 3 required by the definition of gflow (see Definition 3). Furthermore, line 11 of the algorithm ensures condition 1, thus if the algorithm returns a flow then it satisfies the definition of gflows.

Now suppose that the graph admits a gflow  $(g, \prec)$ , then it also admits a maximally delayed gflow  $(g', \prec')$ . The algorithm finds the set  $V_1^{\prec'}$  (in the loop at line 11), and by induction (similarly to the induction in the proof of Theorem 1) it also finds a maximally delayed gflow in  $(G, I, O \cup V_1^{\prec'})$  with the recursive call. Thus the algorithm finds a maximally delayed gflow. □

In order to analyse the complexity, notice that lines 11 to 16 consists in solving a system  $Ax = b_i$  for  $n - \ell$  different  $b_i$ s where  $n = |V|$ ,  $\ell = |Out|$  and  $A$  is a  $(n - \ell) \times \ell$  matrix. In order to solve these  $n - \ell$  systems, the  $(n - \ell) \times n$ -matrix  $M = [A|b_1 \dots b_{n-\ell}]$  is transformed into an upper triangular form within

```

input : An open graph
output: A generalised flow
1 gFlow ( $V, \Gamma, \text{In}, \text{Out}$ ) =
2 begin
3   for all  $v \in \text{Out}$  do
4      $l(v) := 0$  ;
5   end
6   return gFlowaux ( $V, \Gamma, \text{In}, \text{Out}, 1$ ) ;
7 end

8 gFlowaux ( $V, \Gamma, \text{In}, \text{Out}, k$ ) =
9 begin
10   $C := \emptyset$  ;
11  for all  $u \in V \setminus \text{Out}$  do
12    Solve in  $\mathbb{F}_2 : \Gamma_{V \setminus \text{Out}, \text{Out} \setminus \text{In}} \mathbb{I}X = \mathbb{I}\{u\}$  ;
13    if there is a solution  $X_0$  then
14       $C := C \cup \{u\}$  ;  $g(u) := X_0$  ;  $l(u) := k$  ;
15    end
16  end
17  if  $C = \emptyset$  then
18    return ( $\text{Out} = V, (g, l)$ ) ;
19  else
20    return gFlowaux ( $V, \Gamma, \text{In}, \text{Out} \cup C, k + 1$ ) ;
21  end
22 end

```

**Algorithm 2.** Generalised flow

$O(n^3)$  operations using gaussian eliminations for instance, then for each  $b_i$  a back substitution within  $O(n^2)$  operations is used to find  $x_i$ , if it exists, such that  $Ax_i = b_i$  (see [1]). The back substitutions cost  $O(n^3)$  operations at each call of the function. Since there are at most  $n$  recursive calls, the overall complexity is  $O(n^4)$ .

## 5 Depth Optimality

We consider in this section the depth of the flows produced by the algorithms. The depth of a given flow is nothing but the depth of any one-way quantum computation based on the correction strategy described by this flow, even if the preparation of the initial graph state is taken into account since any graph state can be prepared in a constant depth [9].

**Theorem 3.** *A maximally delayed causal flow (resp. gflow) has minimum depth.*

*Proof.* Let  $(g, \prec)$  be a minimum depth causal flow (resp. gflow) of a given open graph. If  $(g, \prec)$  is a maximally delayed causal flow (resp. gflow), then let  $(g', \prec') = (g, \prec)$ . Otherwise, let  $(g', \prec')$  be a maximally delayed causal flow (resp. gflow)

which is more delayed than  $(g, \prec)$ .  $(g', \prec')$  and  $(g, \prec)$  have the same depth. Indeed  $|\cup_{i=0\dots d^{\prec}} V_k^{\prec'}| \geq |\cup_{i=0\dots d^{\prec}} V_k^{\prec}| = |V|$ , thus  $\forall k > d^{\prec}, V_k^{\prec} = \emptyset$ , so  $d^{\prec} \geq d^{\prec'}$ . Since  $(g, \prec)$  has minimum depth  $d^{\prec} \leq d^{\prec'}$ , so  $d^{\prec} = d^{\prec'}$ . As a consequence  $(g', \prec')$  is a minimum-depth maximally delayed causal flow (resp. gflow). Moreover, even if a maximally delayed causal flow (resp. gflow) of a given open graph is not unique, one can prove, using Lemmas 2, 3, and 4, that all the maximally delayed causal flows (resp. gflows) of a given open graph induce the same partition of the vertices, and as a consequence, have the same depth. Thus, a maximally delayed causal flow (resp. gflow) has the same depth as  $(g', \prec')$  which is a minimum depth flow.  $\square$

Notice that the algorithms 1 and 2 produce maximally delayed flows, thus:

**Corollary 1.** *The previous algorithms find an optimal depth flow.*

The depth optimality of the flows found by the previous algorithms have several decisive implications in one-way quantum computation. First, the depth (optimal or not) of a flow is an upper bound on the depth of the corresponding deterministic one-way quantum computation. Moreover, if the one-way quantum computation is uniformly, stepwise and strongly deterministic (which mainly means that if the measurements are applied with an error in the angle which characterises the measurement, then the computation is still deterministic), then the correction strategy must be described by a gflow [3]. As a consequence the algorithm 2 produces the optimal correction strategy, and the depth of the gflow produced by the algorithm is a lower bound on the depth of a uniformly, stepwise and strongly deterministic one-way quantum computation.

## 6 Conclusion

Starting from quantum computational problems (determinism in one-way quantum computation), interesting graph problems have arisen as the property that the depth of correcting strategies for measurement-based quantum computation depends on flows in graphs. We have defined in this paper two algorithms for finding optimal causal flow and gflow. The key points are: the simplification of the structure of the gflows considering only the maximally delayed flows which have a nice recursive structure; a backward analysis (start from the outputs) which allows to take advantage of this structure and avoids backtracking.

From a complexity point of view, an important open question is: given a graph state and a fixed set of measurements (we relax the uniformity condition) what would be the depth of an optimal correction strategy. One direction to answer that question would be to define a weaker flow that is still polynomially computable. One can also consider the characterisation and the depth of computation in more generalised measurement-based models where other planes of measurements are allowed. Finally, these results open up new perspectives of depth optimisation in the more traditional model of quantum circuits: any circuit can be represented, in the one-way model, as an open graph that has a causal flow; moreover, the application on this open graph of the gflow algorithm

introduced in this paper produces a gflow of minimal depth. Investigating how such a one-way quantum computation of minimal depth can be translated back to a quantum circuit which has a smaller depth than the original circuit (but probably more ancillary qubits), should lead to a novel approach to reducing the depth complexity of quantum circuits.

## Acknowledgements

The authors would like to thank Elham Kashefi, Philippe Jorrand and Thierry Boy de la Tour for fruitful discussions.

## References

1. Bard, G.V.: Achieving a  $\log(n)$  Speed Up for Boolean Matrix Operations and Calculating the Complexity of the Dense Linear Algebra step of Algebraic Stream Cipher Attacks and of Integer Factorization Methods. Cryptology ePrint Archive, Report 2006/163 (2006)
2. Broadbent, A., Kashefi, E.: Parallelizing Quantum Circuits. arXiv, quant-ph/0704.1806 (2007)
3. Browne, D., Kashefi, E., Mhalla, M., Perdrix, S.: Generalized flow and determinism in measurement-based quantum computation. NJP 9, 250 (2007)
4. Danos, V., Kashefi, E.: Determinism in the one-way model PRA, 74 (2006)
5. Danos, V., Kashefi, E., Panangaden, P.: The measurement calculus. J. ACM 54(2) (2007)
6. de Beaudrap, N.: Finding flows in the one-way measurement model. Phys. Rev. A 77, 022328 (2008)
7. de Beaudrap, N.: Complete algorithm to find flows in the one-way measurement model (2006) arXiv, quant-ph/0603072
8. Hein, M., Dür, W., Eisert, J., Raussendorf, R., Van den Nest, M., Briegel, H.J.: Entanglement in graph states and its applications. In: Proc. of the Int. School of Physics Enrico Fermi on Quantum Computers, Algorithms and Chaos (July 2005) quant-ph/0602096
9. Høyer, P., Mhalla, M., Perdrix, S.: Resources required for preparing graph states. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288. Springer, Heidelberg (2006)
10. Pei, M., de Beaudrap, N.: An extremal result for geometries in the one-way measurement model. Quantum Information and Computation 8(5) (2008)
11. Raussendorf, R., Briegel, H.: A one-way quantum computer. PRL 86 (2001)
12. Raussendorf, R., Briegel, H.: Computational model underlying the one-way quantum computer. Quantum Information and Computation 2(6) (2002)
13. Walther, P., Resch, K., Rudolph, T., Schenck, E., Weinfurter, H., Vedral, V., Aspelmeyer, M., Zeilinger, A.: Experimental one-way quantum computing. Nature 434 (2005) (quant-ph/0503126)