Quantum Approach for Vertex Separator Problem in Directed Graphs

Ahmed Zaiou, Younès Bennani, Mohamed Hibti, Basarab Matei

Quantum Approach for Vertex Separator Problem in Directed Graphs

Ahmed ZAIOU[1,2,3] , Younès BENNANI[2,3]
Mohamed HIBTI[1] , Basarab MATEI[2,3]

[1] EDF R&D, Palaiseau, France
[2] LIPN - CNRS UMR 7030, Université Sorbonne Paris Nord, France
[3] LaMSN - La Maison des Sciences Numériques, France
{Firstname.Lastname}@{sorbonne-paris-nord}.fr

Abstract. The Vertex Separator Problem of a directed graph consists in finding all combinations of vertices which can disconnect the source and the terminal of the graph, these combinations are minimal if they contain only the minimal number of vertices. In this paper, we introduce a new quantum algorithm based on a movement strategy to find these separators in a quantum superposition with linear complexity. Our algorithm has been tested on small directed graphs using a real Quantum Computer made by IBM.

Quantum Computation, Vertex separators, Minimal cuts, Directed graphs.

1 Introduction

Quantum computing [24] has attracted enormous interest in recent years and attracted many researchers in different disciplines. This excitement followed the two main revolutionary algorithms introduced by Grover and Shor. The first algorithm introduced by Grover [10], manages to reduce the complexity of finding an element in an unstructured dataset of size N to $O(\sqrt{N})$, the second by Shor [28], which can break the RSA code in a polynomial time. One of the main goals of this new research area is to solve problems that can't be solved in the classical framework and to break the computational complexity of many hard problems and sometimes find good shortcuts and new approaches to solve them. In 2012, John Preskill introduced the term "quantum supremacy", a concept to describe that quantum computers can do some things that classical computers can't [26]. Various algorithms were then proposed to achieve this goal (sycamore [25], chinese [29]). Others have shown significant acceleration compared to the best results of classical computers. In graph theory, we can mention Quantum Max-Flow / Min-cut [5] by Shawn at al., which demonstrates that, unlike the classical case, the conjecture max-flow/min-cut quantum is not true in general. Under certain conditions, for example, when the capacity of each edge is a power of a fixed integer, it is proved that the quantum max-flow is equal to the

quantum min-cut. Also, they found connections of the quantum max-flow/min-cut with entanglement entropy and the quantum satisfiability problem. Also, the paper [14] gives a new upper bound on the quantum complexity of queries for deciding st-connectivity on some classes of planar graphs, and shows that this bound is sometimes exponential and also that the evaluation of Boolean formulas reduces to deciding connectivity on such a class of graphs. This paper gives for some classes of boolean formulas a quadratic acceleration with respect to the classical complexity. Kazuya and Ryuhei [27] provided an exponential-time quantum algorithm to calculate the chromatic number, and there are several other works published in these fields like [7] for the graph walk problems, which shows results that improve the best classical algorithms for Eulerian or Hamiltonian circuit problems, the traveling salesman problem and project scheduling. [23] provides two quantum algorithms to find a triangle in an undirected graph or to reject it if the graph is without triangle. The first algorithm uses combinatorial ideas with Grover Search and the second algorithm based on a concept of Ambainis [1]. In addition, we cite the following papers [2, 8].

In this paper, we address the problem of vertex partitioning of a directed graph that has not been solved yet by an accurate quantum algorithm. We provide a quantum algorithm to solve this problem and we also show that this algorithm is easily applicable in the existing frameworks of graph partitioning and that it is also computationally feasible. To show this, we make a study case where we use a small graph as an example, then we explain each iteration of our algorithm by indicating the results found in the quantum computer after each iteration.

This paper is organized as follows: In the next section, we give a formal description of the vertex separator problem. In section 3 we discuss our contribution and we define movement oracles and describe our algorithm and its complexity. In Section 4, we perform a comparative analysis on qualitative grounds and on the basis of some test cases. The paper is finally concluded in section 5.

2 Problem statement and state of the art

We model our problem by directed graphs (networks). In graph theory, the Vertex Separator Problem (VSP) consists of finding a subset of vertices (called a vertex separator) that allows the set of vertices in the graph to be divided into two related components. The VSP is NP-hard [4]. There are a number of algorithms that can find these vertex separators. We mention [15], which presents a heuristic method for partitioning arbitrary graphs that is both efficient in finding optimal partitions and fast enough to be practical in solving large problems. The paper [9] presents a linear time heuristic method for improving network partitions. Both papers [9, 15] are adapted by [3, 13] to generalize the methods and improve the runtime. According to [11], the vertex separator problem can be formulated by a bilinear quadratic programming problem. And, recently, several works [6, 12, 17] and [19] have combined the traditional combinatorial method

and the optimization-based method to improve the performance and quality of the separator. We mention the result of [18] who introduced a new hybrid algorithm for computing vertex separators in arbitrary graphs using computational optimization.

In the classical approach of the vertex separator problem for a planar graph with n vertices, Lipton and Tarjan [21] provided a polynomial time algorithm for finding a single vertex separator. This algorithm was improved in [22] for other families of graphs such as fixed genus graphs. These families of graphs include trees, 3D grids, and meshes that have small spacers. To obtain all minimal vertex separator of a graph, Kloks and Kratsch [16] provided an efficient algorithm listing all minimal vertex separator of an undirected graph. The algorithm requires a polynomial time for each single separator found. In this paper, we are interested in the vertex separator problem (VSP) in a directed graph that has a source s and a bound t, we search in this graph for all vertex separator that separate the source s and the bound t. In order to do this, let us first define a directed graph and a vertex separator:

Definition 1. *A directed graph or network is a graph in which the edges are oriented. More precisely, a directed graph is an ordered pair (V, E) including: (i) V a set of vertices and (ii) $E \subset \{(x, y)|(x, y) \in V \times V, x \neq y\}$ a set of oriented edges or arcs that are ordered and distinct pairs of vertices.*

Definition 2. *A vertex separator $s - t$ noted (S, C, T) is a partition of V such that $s \in S$ and $t \in T$. Then the $s-t$-cut for us is a division of the vertices of the graph into three independent subsets S, C and T, with the source s in the subset S, the terminal t in T and the subset C representing the vertex separator (the cut). The cut C is minimal, it means that the number of vertices existing in C is minimal, that is to say, if we remove only one vertex from C the remainder is not sufficient for a cut.*

For a single directed graph, we can find several minimal cuts between the source and the terminal. In the rest of this paper, we propose our quantum algorithm to determine all the minimal cuts of a directed graph.

3 Our contribution

In this section, we will describe our quantum algorithm for finding all the minimal cuts that can stop the flow between the source and the terminal of a directed graph. This algorithm is based on a movement strategy that uses movement oracles to construct a quantum superposition that contains all these minimal cuts. The first question that arrives here is how to represent all the sets of vertices with quantum qubits? For a graph with n vertices, we will find 2^n different subsets of these vertices. In the quantum framework, with n qubits, we can represent 2^n possible states (we cite this book for the basics of quantum computing [20]). In both cases with n elements we have 2^n possibilities, so we represent the subsets

by the quantum states of these n qubits. To do this, we use for each vertex of the graph a qubit, and each state of these qubits represents a subset of the vertices.

Before starting explaining the functioning of our algorithm, we start with the definition of a movement in a graph and how these movements are applied by quantum circuits.

Definition 3. *Let $G = (V, E)$, a directed graph, the movement of a vertex $v \in V$ corresponds to the move from v to its successors $Succ(v)$.*

$$Mov(v) = Succ(v) \tag{1}$$

The movement of a vertex $v \in \theta$ where $\theta \subset V$ is the substitution of v by its successors $Succ(v)$ in the set θ.

$$Mov_\theta(v) = \{\theta \setminus v\} \cup Succ(v) \tag{2}$$

In the quantum setting, to apply these moves, we use quantum oracles called movement oracles. Each vertex has a movement oracle, which allows to apply the movement if the vertex of the movement exists in the input subset and provide in the output the input and also the subsets after the movement.

For a vertex v and a given subset of vertices θ, such that $v \in \theta$. we assume that the subset θ is represented by the quantum state $|\psi_\theta\rangle$. To apply the movement of v, we give the quantum state $|\psi_\theta\rangle$ to the oracle as an input, and in the output of the oracle we find a superposition which contains two states $|\psi_\theta\rangle$ and the state $|\psi_{\theta'}\rangle$, with $\theta' = (\theta \setminus v) \cup Succ(v)$.

More generally, suppose that the input set is the union of two subsets $\theta = \theta_1 \cup \theta_2$ represented by the quantum superposition $|\psi_\theta\rangle = \alpha_1 |\theta_1\rangle + \alpha_2 |\theta_2\rangle$ where the state $|\theta_1\rangle$ represents the subset θ_1 and the state $|\theta_2\rangle$ represents the subset θ_2. Consider a vertex $w \in \theta_1$ and $w \notin \theta_2$. The output of the movement oracle of $w \in \theta$ is $|\psi_{out}\rangle = \frac{\alpha_1}{\sqrt{2}} |\theta_1\rangle + \alpha_2 |\theta_2\rangle + \frac{\alpha_1}{\sqrt{2}} |\theta_3\rangle$, where the state $|\theta_3\rangle$ represents the subset $\theta_3 = Mov_{\theta_1}(w) = \{\theta_1 \setminus w\} \cup Succ(w)$.

To provide a general formula for a movement oracle, let $v \in V$ be a vertex, O_v be the movement oracle of v and $|\psi_\theta\rangle = \sum_i \beta_i \psi_{\theta_i}$, $i = 1, \ldots, N$ is a quantum superposition which represents N subsets of vertices $\{i = 1, \ldots, N\}$. The general formula for the movement oracle of a vertex v is :

$$|\psi'_\theta\rangle = O_v |\psi_\theta\rangle = \sum \alpha_e f_v(e) |\psi_{\theta_e}\rangle \tag{3}$$

$$f_v(e) = \begin{cases} 1 & \text{if } \theta_e \in \{\theta_i\}_{i=1,\ldots,N} \\ 1 & \text{if } \theta_e \in \{Mov_{\theta_i}(v)\}_{i=1,\ldots,N} \\ 0 & \text{else} \end{cases} \text{ and } \begin{cases} \alpha_e = 0 & \text{if } f_v(e) = 0 \\ \sum_e \alpha_e = 1 \end{cases}$$

In order to represent an oracle with a simple quantum circuit, we need to add two additional control qubits $|c_0\rangle$ and $|c_1\rangle$. The qubit $|c_0\rangle$ is used to check if the

vertex of the movement exists in the input set and it will be in the state $|1\rangle$, if it exists in the input set, and in $|0\rangle$ otherwise. For this, we use the C-X gate with the qubit corresponding to the vertex of the movement as control and the $|c_0\rangle$ qubit as target. If the vertex is in the input set, we add another set to the collection of cuts. In other words, if the vertex qubit is in the $|1\rangle$ state, we add a new state to the input superposition. To do this, we use the C-H gate with the qubit $|c_0\rangle$ as control and the qubit of the vertex of the movement as target. Then, we use a C-X gate to apply the movement to the new set. To add all the successors of the movement vertex to the new set, we use the circuit of the figure 1 which allows to flip the qubit of the successor into the state $|1\rangle$ if it is in the state $|0\rangle$ and to do nothing if it is in the state $|1\rangle$.

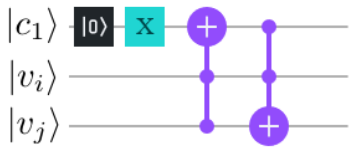


Fig. 1: The movement circuit from v_i to v_j, this circuit uses three qubits: $|v_i\rangle$, $|v_j\rangle$ the movement vertex and the successor respectively and $|c_1\rangle$ for the control.

3.1 Algorithm description

Let $G = (V, E)$ be a directed graph, with V being the set of vertices such that $|V| = n$ and E the set of edges, the source of the graph is the vertex s and the terminal is the vertex t. The first step of the algorithm consists in preparing the necessary number of qubits to represent all the possible subsets of vertices of the graph. The graph has n vertices, so we use n quits. These n qubits are initialised in the state $|0\rangle$, so the quantum state it initialized to $|\psi\rangle = |0 \dots 0\rangle$. With the first qubit corresponding to the source vertex s, the second qubit for the second vertex, until the last qubit for the last vertex (the sink vertex t). There are only zeros in the state $|\psi\rangle$, which means all the vertices are absent in the cut represented by $|\psi\rangle$.

$$|\psi\rangle = |tv_n \dots v_1 s\rangle = |00 \dots 00\rangle \iff \psi = \{\} \tag{4}$$

To start with a set ψ containing only the input vertex s, we apply the `not gate` on the first qubit, which gives us $|\psi\rangle = |0 \dots 01\rangle$ as a result, with the qubit $|s\rangle$ is in the state $|1\rangle$.

In the second step, for each vertex, we have an oracle of movement, so we call all these oracles of movement. The first oracle corresponding to the movement

of the vertex s to its successors:

$$|\psi_1\rangle = O_1 |\psi\rangle = \alpha_1 |\psi\rangle + \alpha_2 |Mov_\psi(s)\rangle \tag{5a}$$
$$= \alpha_1 |\psi\rangle + \alpha_2 |Succ(s)\rangle \tag{5b}$$

After this iteration, the oracle O_1 adds to the superposition the state $|Succ(s)\rangle$ which represent the first cut of the graph. After that, we apply all the remaining oracles:

$$|\psi_{fin}\rangle = O_n O_{n-1} \dots O_2 |\psi\rangle \tag{6}$$

Each one of these oracles adds a number of states to the superposition, which means its adds a number of cuts to the set of cuts represented by the superposition.

$$|\psi_{fin}\rangle = \alpha_1 |cut_1\rangle + \dots + \alpha_k |cut_k\rangle \tag{7}$$

After the n oracles, in the output superposition $|\psi_{fin}\rangle = \sum_i \alpha_i |cut_i\rangle$, we find all the possible minimal cuts represented by the states $|cut_i\rangle$. Finally, we use a simple classical filter to eliminate non-minimal cuts. The algorithm 1 represents the steps to generate the quantum circuit to find all the possible minimal cuts.

Algorithm 1: All Minimum cuts sets

Input : Graph $G = (V, E)$, with $n = |V|$ is the number of vertices of the graph, source vertex s, sink vertex t

Output: Min-cuts Cs

Start:

Initialized n qubits,

$$|\psi_0\rangle = |0 \dots 0\rangle$$

Apply the not gate X in the first qubit which represents the source s

$$|\psi_1\rangle = |0 \dots 01\rangle$$

Make the movement of s we apply the oracle O_s

$$|\psi_2\rangle = O_s |\psi_1\rangle$$

for *each* $v \in V$ *and* v *!=* s *and* t *is not a successor of* v **do**

$$|\psi_{i+1}\rangle = O_v |\psi_i\rangle$$

Cs = measured $|\psi_{n-1}\rangle$ and eliminate non-minimal cuts.

return Cs

3.2 Complexity Analysis

Suppose that we have a graph $G = (V, E)$, with V the set of vertices and E the set of edges such that $|V| = n$ and $|E| = m$. To build the circuit, we need n qubits to represent all the possible states of the graph and 2 auxiliary qubits

for the control. For n oracles of movements we need n gates C-H, $2n-2$ gates C-X, $m+2$ gates X and $2m$ gates CC-X. Therefore, our algorithm has a linear complexity either in terms of memory (number of qubits used), or in terms of computation (n oracles of movements).

4 Case study

In this section, we present the detailed version of the case study of our algorithm. For that, let us take the graph $G = (V, E)$ represented in the figure 2, where V is the set of vertices of size 9 ($n = |V| = 9$), which is labeled from v_0 to v_8 as follows: $V = \{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. And the set of edges between these vertices is noted E and is presented like the following: $E = \{(v_0, v_1), (v_0, v_2), (v_1, v_5), (v_1, v_7), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_6), (v_4, v_8), (v_5, v_6), (v_6, v_7), (v_7, v_8)\}$. We have also fixed the source s in the vertex v_0 and the terminal t in the vertex v_8.

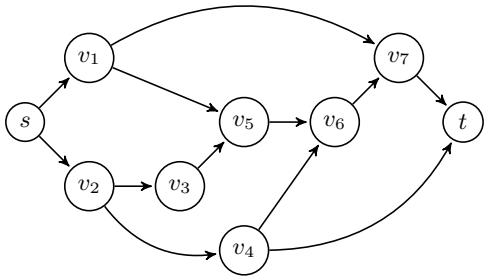
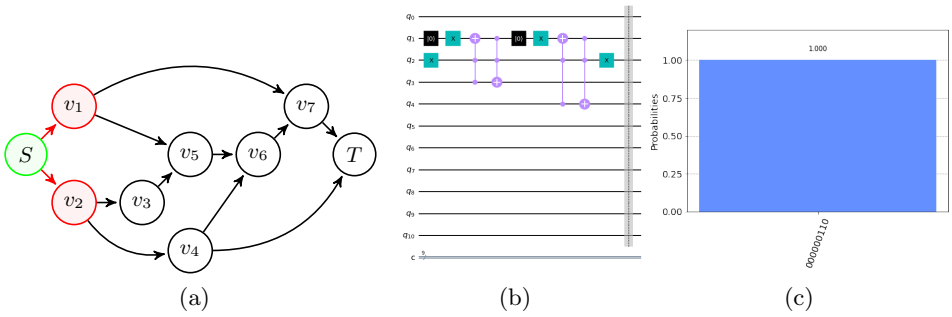


Fig. 2: Directed graph of 9 vertices

Visually, we can identify the set of vertex separators defining the minimal cuts : $Cs = \{C_1 = \{v_1, v_2\}, C_2 = \{v_2, v_7\}, C_3 = \{v_4, v_7\}, C_4 = \{v_1, v_4, v_5\}, C_5 = \{v_1, v_3, v_4\}, C_6 = \{v_1, v_4, v_6\}\}$, with each subset $C_i, i = 1, \ldots, 6$ is a vertex separators of the graph with a minimal number of vertices. In the following, we want to find this set of minimal cuts Cs by our quantum algorithm. To do this, we use IBM's quantum simulator to show the intermediate results of our algorithm. In the graph G we have $n = 9$ vertices, to represent all the possible combinations of these vertices we use 9 qubits. The source s is represented by the first qubit $|v_0\rangle$ and each vertex $v_i, i = 1, \ldots, 7$ is associated to the qubit $|v_i\rangle, i = 1, \ldots, 7$ and the sink t is associated to the last qubit $|v_8\rangle$. These 9 qubits $|v_i\rangle, i = 0, \ldots, i = 9$ can represent 2^9 possibles states, therefore, these qubits can represent the superposition $|v_8 v_7 v_6 v_5 v_4 v_3 v_2 v_1 v_0\rangle = \sum_{i=0}^{2^9-1} \alpha_i |C_i\rangle$, which represent the all possible subsets of vertices of the graph G. Each state $|C_i\rangle$ in the superposition $|v_8 v_7 v_6 v_5 v_4 v_3 v_2 v_1 v_0\rangle$ represent a single vertex separator, we say the vertex v_i belongs to the vertex separator $|C_i\rangle$ (or the minimal cuts $|C_i\rangle$) if

the corresponding qubit $|v_i\rangle$ in the state $|C_i\rangle$ is in the state $|1\rangle$. For example, the vertex separator $C = \{v_1, v_2\}$ can be encoded by the quantum state $|000000110\rangle$.

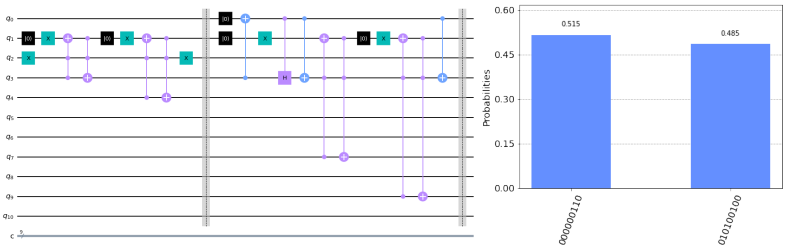
Suppose now that all the successors of the source $s = v_0$ are down, then there is no other way to go to the next vertices, so the subset of the successors of the vertex $s = v_0$ is a vertex separator, in addition, if one of these successors is in good condition, we will find a way to go to the next vertices. Therefore, the subset of successors of $s = v_0$ is a vertex separator with a minimal number of vertices, in other words a minimal cut. Then, to find this first minimal cut, we apply the first oracle of the movement O_S on the state $|\psi_0\rangle$. That is to say that we take $|\psi_0\rangle$ as the input state and we apply the movement of the vertex $v_0 = s$ to its successors v_1 and v_2, which gives us the output state $|\psi_1\rangle = |000000110\rangle$. The graph of the movement of the oracle s and the result in output are represented in the figure 3.



(a) (b) (c)

Fig. 3: The movement of s towards the two successors v_1 and v_2 represented in (a) can be managed by the oracle (b). (c) The result of the execution gives the state $|000000110\rangle$ which represents the first cut $\{v_1, v_2\}$.

Now, suppose that one of the successors v_1 and v_2 is in a good condition, then there is a way to go to the following vertices. For example, if vertex v_1 is in a good condition we can find a path to the terminal through the successors of v_1. If these successors of v_1 are out of order, we cannot find a path to the terminal. Therefore, the subset $Succ(v_1) \cup C_1 \setminus \{v_1\}$ is a cut. So, if we apply the movement of v_1 in the state $|\psi_1\rangle$, we find a new minimal cut containing the successors of v_1 and the vertex v_2. Here, the oracle uses the Hadamard gate to keep the first cut and add the new state corresponding to the new cut. The figure 4 presents the circuit with the second oracle and the execution results in the simulator. At step k, we apply the oracle O_k on the output superposition of step $k-1$, therefore, we apply the movement of the vertex v_k corresponding to the oracle O_k, which adds new states in the superposition $|\psi_k\rangle$, if the qubit corresponding



(a) (b)

Fig. 4: (a) The second oracle starts after the first barrier in the circuit. (b) The result of the run gives two states: $|000000110\rangle$ represents the input and $|010100100\rangle$ represents the new cut after the movement. Note that the second oracle in (a) uses the Hadamard gate to add the new state into the superposition.

to the vertex v_k is in the state $|1\rangle$ for each state of the superposition $|\psi_{k-1}\rangle$.

$$\begin{aligned} |\psi_k\rangle &= O_k |\psi_{k-1}\rangle \\ &= O_k \sum_j \alpha_j |C_j\rangle = \sum_j \alpha_j O_k |C_j\rangle \\ &= \sum_j \beta_j |C_j\rangle + \sum_j \beta_j Mov_{v_k}(|C_j\rangle) \end{aligned}$$

with $\sum_j \alpha_j = 1$ and $\sum_j \beta_j = 1$.

After all the possible movements, we found the superposition $|\psi_{final}\rangle$.

$$|\psi_{final}\rangle = \sum_i \alpha_i |v_i\rangle = \sum_i \alpha_i |v_{i_8} v_{i_7} v_{i_6} v_{i_5} v_{i_4} v_{i_3} v_{i_2} v_{i_1} v_{i_0}\rangle \tag{9}$$

where $\sum_i \alpha_i = 1$ and each state $|i\rangle = |v_{i8} v_{i7} v_{i6} v_{i5} v_{i4} v_{i3} v_{i2} v_{i1} v_{i0} c_{i_1} c_{i_0}\rangle$ of the state $|\psi_{final}\rangle$ represents a cut, with $|v_{ij}\rangle = |1\rangle$ if the vertex j is in the cut i and $|v_{ij}\rangle = |0\rangle$ in the otherwise.

In our graph example, after the execution of the circuit 5 in IBM's simulator and the quantum computer IBM Q 16 Melbourne, we present the results in the figure 6. Finally, we remove the non minimal cuts. For this, for each (i, j) we eliminate the cut C_j if $C_i \in C_j$. To verify the results, we visualized each state of the superposition 6 in an independent graph, with red color if the vertex qubit in the state $|1\rangle$ (present in the minimal cut) and black if it' s in the state $|0\rangle$.

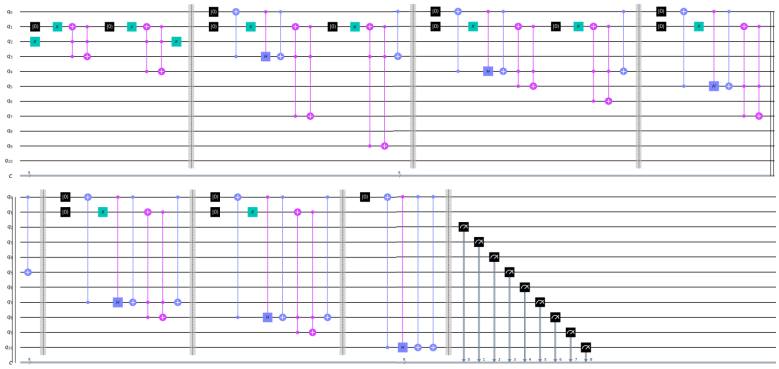


Fig. 5: The circuit uses 11 qubits: 9 to represent all possible subsets of vertices, 2 for the control. And also it uses 7 movement oracles separated by vertical separators. Each oracle represents the movement of a vertex. At the end of the circuit, we measure the 9 qubits to find the superposition which represents all the minimal cuts.

5 Conclusion

We have proposed a quantum algorithm to find all the minimal cuts of a directed graph. More precisely, we propose a quantum algorithm that uses movement oracles to generate as output a superposition of all states that represent the minimal cuts. In this paper, cuts are represented by a set of vertices, which can separate the source and the terminal of the graph, and they are minimal if they contain just the minimal number of vertices to represent a cut. Also, the complexity of our algorithm is linear, because: it uses only $n + 2$ qubits, n to represent all the possible combinations of vertices and 2 for the counter, and it uses n oracles of movements, n being the number of vertices of the graph.

Acknowledgements

This research work was partially funded by the European project NExt ApplicationS of Quantum Computing (NEASQC), supported by the Horizon 2020-FETFLAG (Grant Agreement 951821).

References

1. Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.
2. Simon Apers and Troy Lee. Quantum complexity of minimum cut. *arXiv preprint arXiv:2011.09823*, 2020.
3. Cleve Ashcraft and Joseph WH Liu. A partition improvement algorithm for generalized nested dissection. *Boeing Computer Services, Seattle, WA, Tech. Rep. BCSTECH-94-020*, 1994.

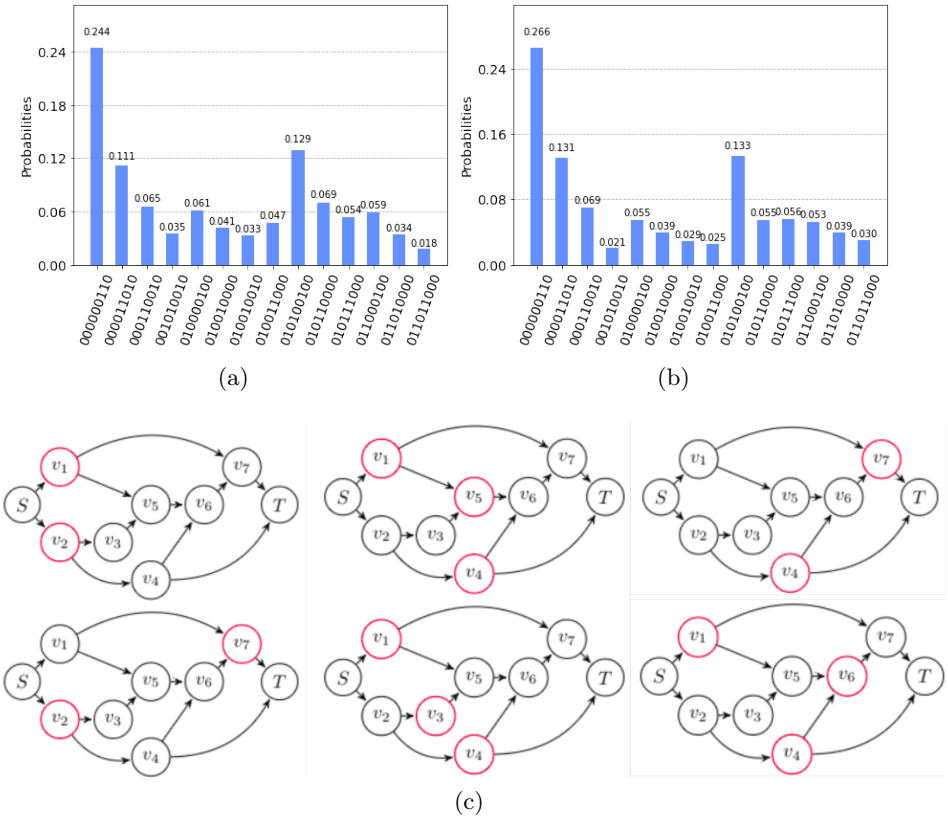


Fig. 6: The histograms represent the superposition of output $|\psi_{final}\rangle$. (a) The results of the execution in IBM's Qasm simulator. (b) The results of the execution in IBM Q 16 Melbourne quantum computer. (C) Results in the graph with red color represent the cuts.

4. Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.
5. Shawn X Cui, Michael H Freedman, Or Sattath, Richard Stong, and Greg Minton. Quantum max-flow/min-cut. *Journal of Mathematical Physics*, 57(6):062206, 2016.
6. Timothy A Davis, William W Hager, Scott P Kolodziej, and S Nuri Yeralan. Algorithm xxx: Mongoose, a graph coarsening and partitioning library. *ACM Trans. Math. Software*, 2019.
7. Sebastian Dörn. Quantum algorithms for graph traversals and related problems. In *Proceedings of CIE*, volume 7, pages 123–131. Citeseer, 2007.
8. Zachary Eldredge, Leo Zhou, Aniruddha Bapat, James R Garrison, Abhinav Deshpande, Frederic T Chong, and Alexey V Gorshkov. Entanglement bounds on the performance of quantum computing architectures. *Physical review research*, 2(3):033316, 2020.

9. Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *19th design automation conference*, pages 175–181. IEEE, 1982.
10. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
11. William W Hager and James T Hungerford. Continuous quadratic programming formulations of optimization problems on graphs. *European Journal of Operational Research*, 240(2):328–337, 2015.
12. William W Hager, James T Hungerford, and Ilya Safro. A multilevel bilinear programming algorithm for the vertex separator problem. *Computational Optimization and Applications*, 69(1):189–223, 2018.
13. Bruce Hendrickson and Edward Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM Journal on Scientific Computing*, 20(2):468–489, 1998.
14. Stacey Jeffery and Shelby Kimmel. Quantum algorithms for graph connectivity and formula evaluation. *Quantum*, 1:26, 2017.
15. Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.
16. Ton Kloks and Dieter Kratsch. Listing all minimal separators of a graph. *SIAM Journal on Computing*, 27(3):605–613, 1998.
17. S. Kolodziej and T. Davis. Vertex separators with mixed-integer linear optimization. *17th SIAM Conference on Parallel Processing for Scientific Computing*, 2016.
18. Scott P Kolodziej and Timothy A Davis. Generalized gains for hybrid vertex separator algorithms. In *2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing*, pages 96–105. SIAM, 2020.
19. Scott Parker Kolodziej. *Computational Optimization Techniques for Graph Partitioning*. PhD thesis, 2019.
20. Michel Le Bellac. *Introduction à l'information quantique*. Belin, 2005.
21. Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
22. Richard J Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM journal on computing*, 9(3):615–627, 1980.
23. Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007.
24. Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.
25. Edwin Pednault, John A Gunnels, Giacomo Nannicini, Lior Horesh, and Robert Wisnieff. Leveraging secondary storage to simulate deep 54-qubit sycamore circuits. *arXiv preprint arXiv:1910.09534*, 2019.
26. John Preskill. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.
27. Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In *Latin American Symposium on Theoretical Informatics*, pages 387–398. Springer, 2021.
28. Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
29. Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.