Span-Program-Based Quantum Algorithms for Graph Bipartiteness and Connectivity

Agnis Āriņš[✉]

University of Latvia, Raiņa Bulvāris 19, Riga 1586, Latvia
`agnis.arins@lu.lv`

Abstract. Span program is a linear-algebraic model of computation which can be used to design quantum algorithms. For any Boolean function there exists a span program that leads to a quantum algorithm with optimal quantum query complexity. In general, finding such span programs is not an easy task.

In this work, given a query access to the adjacency matrix of a simple graph G with n vertices, we provide two new span-program-based quantum algorithms:
- an algorithm for testing if the graph is bipartite that uses $O(n\sqrt{n})$ quantum queries;
- an algorithm for testing if the graph is connected that uses $O(n\sqrt{n})$ quantum queries.

1 Introduction

The concept of a span program as a linear-algebraic model of computation is not new. It was introduced by Karchmer and Wigderson in 1993 [9] and has many applications in classical complexity theory. Span programs can be used to evaluate decision problems. In 2008 Reichardt and Spalek [12] introduced a new complexity measure for span programs – witness size, which, as Reichardt showed later in [10,11], has strong connection with the quantum query complexity. There is a quantum algorithm for evaluating span programs [10] and these two complexity measures are essentially equivalent. The difficulty is to come up with a span program with a good witness size complexity.

In [12] authors dealt with bounded-size span programs evaluating Boolean functions each on $O(1)$ bits and posed an open question – do there exist interesting quantum algorithms based directly on asymptotically large span program? Belovs used span programs to construct learning graphs [3,4]. He also used span program approach for the matrix rank problem [2]. In [1] Ambainis et al. came up with a simple yet powerful span program for the graph collision problem.

In this paper, we extend the family of algorithms based on span programs. We present two new span-program-based quantum algorithms – an $O(n\sqrt{n})$ algorithm for the graph bipartiteness problem and an $O(n\sqrt{n})$ algorithm for the

graph connectivity problem. Both algorithms in the quantum query sense are optimal because the witness sizes match the quantum query complexity lower bounds [6,7] for these problems. Thus we demonstrate that span programs can be useful also for the problems with an asymptotically large input and possibly our algorithms could be building blocks for bigger span programs in the future.

The graph connectivity problem has been studied before [7] and there already exists a $O(n\sqrt{n})$ quantum query algorithm which requires $O(n)$ qubits of quantum memory. The advantage of our algorithm is that it uses only $O(\log n)$ qubits of quantum memory because the span program P_2 uses a vector space with $O(n^2)$ dimensions. Similarly for the graph bipartiteness problem. It can be solved with the breadth-first search method [8] which uses $O(n)$ qubits of quantum memory, but our approach with a span program requires $O(\log n)$ qubits of quantum memory.

2   Preliminaries

In this paper, we present algorithms which work on simple graphs, given in adjacency model. If the given graph has n vertices then the input size for an algorithm is $n \times n$ and we assume that the input variable $x_{i,j}$ corresponds to the value of entry in i-th row and j-th column of the adjacency matrix.

2.1   Span Programs

Definition 1 [1]. *A span program P is a tuple $P = (H, |t\rangle, V)$, where H is a finite-dimensional Hilbert space, $|t\rangle \in H$ is called the target vector, and $V = \{V_{i,b} | i \in [n], b \in \{0,1\}\}$, where each $V_{i,b} \subseteq H$ is a finite set of vectors.*

Denote by $V(x) = \bigcup \{V_{i,b} | i \in [n], x_i = b\}$. The span program is said to compute function $f : D \to \{0,1\}$, where the domain $D \subseteq \{0,1\}^n$, if for all $x \in D$,

$$f(x) = 1 \iff |t\rangle \in \mathrm{span}(V(x)).$$

Basically, what this definition says is that for each input variable x_i we have two sets of vectors (as the span program authors, we define these vectors in advance) – $V_{i,0}$ and $V_{i,1}$. If $x_i = b$ then we say that vectors from the set $V_{i,b}$ are available and vectors from the set $V_{i,1-b}$ are not available. If some vector is included in both sets $V_{i,0}$ and $V_{i,1}$ then we say that it is a free vector – it is always available.

The function returns 1 iff the target vector can be expressed as a linear combination of the available vectors, otherwise it returns 0.

Definition 2 [1].

(1) *A* positive witness *for $x \in f^{-1}(1)$ is a vector $w = (w_v), v \in V(x)$, such that $|t\rangle = \sum_{v \in V(x)} w_v v$. The* positive witness size *is*

$$\mathrm{wsize}_1(P) := \max_{x \in f^{-1}(1)} \min_{w:witness\ of\ x} \|w\|^2.$$

(2) A negative witness *for $x \in f^{-1}(0)$ is a vector $w \in H$, such that $\langle t|w \rangle = 1$ and for all $v \in V(x)$: $\langle v|w \rangle = 0$. The* negative witness size *is*

$$\text{wsize}_0(P) := \max_{x \in f^{-1}(0)} \min_{w:\text{witness of } x} \sum_{v \in V} \langle v|w \rangle^2.$$

(3) The witness size of a program P *is*

$$\text{wsize}(P) := \sqrt{\text{wsize}_0(P) \cdot \text{wsize}_1(P)}.$$

(4) The witness size of a function f *denoted by* $\text{wsize}(f)$ *is the minimum witness size of a span program that computes f.*

Theorem 1 [10, 11]**.** $Q(f)$ *and* $\text{wsize}(f)$ *coincide up to a constant factor. That is, there exists a constant $c > 1$ which does not depend on n or f such that $\frac{1}{c}\text{wsize}(f) \le Q(f) \le c \cdot \text{wsize}(f)$.*

3   Span Program for Testing Graph Bipartiteness

A bipartite graph is a graph whose vertices can be divided into two disjoint sets such that there is no edge that connects vertices of the same set. An undirected graph is bipartite iff it has no odd cycles.

Algorithm 1. *There exists a span program P which for a graph with n vertices detects if the graph is bipartite with $wsize(P) = O(n\sqrt{n})$.*

Proof. We will make a span program which detects if a graph has an odd cycle.

Let $n = |G|$ be a number of vertices in the given graph G. Then the span program is as follows:

Span program P_1 for testing graph bipartiteness

– H is a $(2n^2 + 1)$ dimensional vector space with basis vectors $\{|0\rangle\} \cup \{|v_{k,b}\rangle \,|\, v, k \in [1..n], b \in \{0, 1\}\}$.
– The target vector is $|0\rangle$.
– For every $k \in [1..n]$ make available the free vector $|0\rangle + |k_{k,0}\rangle + |k_{k,1}\rangle$.
– For every $k \in [1..n]$, for every edge $u - v$ (where input $x_{u,v} = 1$), make available the vectors $|u_{k,0}\rangle + |v_{k,1}\rangle$ and $|u_{k,1}\rangle + |v_{k,0}\rangle$.

The states in the span program P_1 are mostly in the form $|v_{k,b}\rangle$ where v is vertex index, k represents vertex from which we started our search for an odd length cycle and b represents the parity of the current path length. The first subindex k in state $|v_{k,b}\rangle$ can also be considered as the subspace index for the subspace $V_k = \text{span}(\{|v_{k,b}\rangle \,|\, v \in [1..n], b \in \{0, 1\}\})$. Vectors corresponding to

edges are in the form $|u_{k,b}\rangle + |v_{k,1-b}\rangle$ consisting from sum of two states which both belong to same subspace V_k.

In the span program P_1 the target vector $|0\rangle$ can only be expressed as a linear combination of the available vectors if at least one of the vectors in the form $|k_{k,0}\rangle + |k_{k,1}\rangle$ can be expressed. Without loss of generality, if there is an odd length cycle $v_1 - v_2 - \cdots - v_{(2j+1)} - v_1$ then the target vector can be expressed by taking the vectors corresponding to the edges of this cycle, alternatingly with plus and minus sign

$$|0\rangle = (|0\rangle + |1_{1,0}\rangle + |1_{1,1}\rangle) - (|1_{1,0}\rangle + |2_{1,1}\rangle) + \cdots - (|(2j+1)_{1,0}\rangle + |1_{1,1}\rangle)$$

therefore the span program P_1 will always return 1 when the given graph is not bipartite.

From the other side, if there is no odd length cycle then none of the vectors in the form of $|k_{k,0}\rangle + |k_{k,1}\rangle$ can be expressed using the available vectors from P_1. To cancel out the state $|k_{k,0}\rangle$ we should be using a vector $|k_{k,0}\rangle + |v_{k,1}\rangle$ corresponding to some edge $k - v$ where v is some vertex adjacent to k because no other vector contains the state $|k_{k,0}\rangle$. By doing so we move from the state $|k_{k,0}\rangle$ to the state $|v_{k,1}\rangle$ (possibly with some coefficient other than 1) which has the parity bit flipped. Similarly, to cancel out the state $|v_{k,1}\rangle$ we should be using a vector corresponding to some edge going out from vertex v. To stop this process we need to reach the state $|k_{k,1}\rangle$. It can be done only if there is an odd cycle because the path must be closed and the parity bit restricts it to odd length. When there is no odd cycle, span program P_1 will always return 0.

We can conclude that P_1 indeed computes the expected function. It remains to calculate the witness size of P_1.

For the case when there is an odd cycle we need to calculate the positive witness size. If there is an odd cycle $v_1 - v_2 - \cdots - v_d - v_1$ with length d then the target vector can be expressed in this way

$$|0\rangle = 1 \cdot (|0\rangle + |1_{1,0}\rangle + |1_{1,1}\rangle) + (-1) \cdot (|1_{1,0}\rangle + |2_{1,1}\rangle) + \cdots + (-1) \cdot (|d_{1,0}\rangle + |1_{1,1}\rangle)$$

and the positive witness w here consists only from $d + 1$ entries ± 1 therefore $\|w\|^2 = d + 1$.

If $v_1 - v_2 - \cdots - v_d - v_1$ is a cycle then also $v_2 - v_3 - \cdots - v_d - v_1 - v_2$ is a cycle and therefore the target vector can also be expressed in this way

$$|0\rangle = (|0\rangle + |2_{2,0}\rangle + |2_{2,1}\rangle) - (|2_{2,0}\rangle + |3_{2,1}\rangle) + \cdots - (|1_{2,0}\rangle + |2_{2,1}\rangle)$$

the same follows for all d vertices in this cycle and the target vector therefore can be expressed in atleast d different ways. We can combine these d ways each taken with coefficient $1/d$ and then we get that the positive witness size

$$\text{wsize}_1(P_1) \leq d * (1/d)^2 * (d + 1) < 2 \tag{1}$$

To estimate the negative witness size we must find a negative witness w'. We derive w' by defining how it acts on basis vectors. From definition $\langle w'|0\rangle = 1$.

For every k we must have $\langle w'|\left(|0\rangle + |k_{k,0}\rangle + |k_{k,1}\rangle\right)\rangle = 0$ therefore lets pick w' in such a way that $\langle w'|k_{k,0}\rangle = 0$ and $\langle w'|k_{k,1}\rangle = -1$. Now repeat the following steps until no changes happen:

– for every available vector $|u_{k,0}\rangle + |v_{k,1}\rangle$ if $\langle w'|u_{k,0}\rangle$ is defined then define $\langle w'|v_{k,1}\rangle = -\langle w'|u_{k,0}\rangle$.
– for every available vector $|u_{k,1}\rangle + |v_{k,0}\rangle$ if $\langle w'|u_{k,1}\rangle$ is defined then define $\langle w'|v_{k,0}\rangle = -\langle w'|u_{k,1}\rangle$.

For all not yet defined $\langle w'|v_{k,b}\rangle$ define $\langle w'|v_{k,b}\rangle = 0$.

For any given vector v in span program P_1 the value $\langle w'|v\rangle^2 \leq 1$. The total number of vectors does not exceed $n + n^3$ therefore the negative witness size is

$$\text{wsize}_0(P_1) \leq 1 \cdot (n + n^3) \tag{2}$$

Combining positive and negative witness sizes we obtain the upper bound for witness size which also corresponds to quantum query complexity

$$\text{wsize}(P_1) = \sqrt{\text{wsize}_0(P_1) \cdot \text{wsize}_1(P_1)} = O\left(n\sqrt{n}\right) \tag{3}$$

□

4   Span Program for Testing Graph Connectivity

A graph is said to be connected if every pair of vertices in the graph is connected. If in an undirected graph one vertex is connected to all other vertices then by transitivity the graph is connected.

Algorithm 2. *There exists a span program P which for a graph with n vertices detects if the graph is connected with $wsize(P) = O(n\sqrt{n})$.*

Proof. Let $n = |G|$ be a number of vertices in the given graph G. Then the span program is as follows:

Span program P_2 for testing graph connectivity

– H is a $n^2 - 1$ dimensional vector space with basis vectors $\{|v_k\rangle \,|\, v \in [0..n], k \in [2..n]\}$.
– The target vector is $|t\rangle = |0_2\rangle + |0_3\rangle + \cdots + |0_n\rangle$.
– For every $k \in [2..n]$ make available the free vector $|0_k\rangle + |1_k\rangle - |k_k\rangle$.
– For every $k \in [2..n]$, for every edge $u - v$ (where $u, v \in [1..n]$ and input $x_{u,v} = 1$), make available the vector $|u_k\rangle - |v_k\rangle$.

If all vertices are reachable from vertex with index 1 then the given graph is connected. Here we use Belov's [5] span program for *s-t connectivity* as subroutine. This subroutine checks if in a given graph there is a path from the vertex s to the vertex t. The span program for it has the target vector $|s\rangle - |t\rangle$ and for each edge $i - j$ (input $x_{i,j} = 1$) we can use the vector $|i\rangle - |j\rangle$.

In span program P_2, by using this subroutine $n-1$ times, we check if all other vertices are connected to vertex with index 1. We create a separate subspace $V_k = \mathrm{span}(\{|v_k\rangle \, |v \in [0..n]\})$ for each such subroutine call to avoid any interference between them, which is a common technique [10] how to compose span programs. The span program returns 1 when all vertices are connected, but otherwise it returns 0.

For the case when the given graph is connected we need to calculate the positive witness size. In each *s-t* subroutine the shortest path length from the vertex s to the vertex t can not be larger than $n - 1$. Threfore each vector from the set $\{|0_k\rangle \, |k \in [2..n]\}$ requires no more than n vectors to express it. There are $n - 1$ such subroutines. The positive witness size is

$$wsize_1(P_2) \leq n \cdot (n-1) \leq n^2 \tag{4}$$

To estimate the negative witness size we must find a negative witness w'. We derive w' by defining how it acts on the basis vectors. From definition $\langle w'|t\rangle = 1$. We need to talk about negative witness only when some vertex v is not connected to vertex with index 1. Then the vertex v belongs to different connected component than vertex with index 1. Lets name this connected component C_v and let the count of vertices in this connected component be d_v. Lets pick w' in such a way that for each vertex $v_k \in C_v$ set $\langle w'|0_k\rangle = 1/d_v$ and for each vertex $v_z \notin C_v$ set $\langle w'|0_z\rangle = 0$.

For $k \in [2..n]$ we must have $\langle w'|(|0_k\rangle + |1_k\rangle - |k_k\rangle)\rangle = 0$ therefore set $\langle w'|1_k\rangle = -\langle w'|0_k\rangle$ and $\langle w'|k_k\rangle = 0$. Now repeat the following step until no changes happen: for every available vector $|u_k\rangle - |v_k\rangle$ if $\langle w'|u_k\rangle$ is defined then define $\langle w'|v_k\rangle = \langle w'|u_k\rangle$. For all other not yet defined basis vectors $|v_k\rangle$ set $\langle w'|v_k\rangle = 0$.

With such negative witness w' choice the overall negative witness size will only get increased by vectors which correspond to nonexistent graph edges which connects C_v with other connected components in graph - i.e. border edges. An edge $u-v$ is a border edge if $u \in C_v$ and $v \notin C_v$. To a border edge $u-v$ correspond the vectors $|u_k\rangle - |v_k\rangle$ where $k \in [2..n]$ but only d_v from these vectors will have $\langle w'|u_k\rangle \neq \langle w'|v_k\rangle$ and each such vector increases the negative witness size by value $(1/d_v)^2$. For C_v there are at most $d_v \cdot (n - 1)$ border edges therefore the overall negative witness size is

$$\mathrm{wsize}_0(P_2) \leq d_v^2 \cdot (n-1) \cdot (1/d_v)^2 \leq n \tag{5}$$

Combining the positive and negative witness sizes we obtain the upper bound for the witness size which also corresponds to the quantum query complexity

$$\mathrm{wsize}(P_2) = \sqrt{\mathrm{wsize}_0(P_2) \cdot \mathrm{wsize}_1(P_2)} = O\left(n\sqrt{n}\right) \tag{6}$$

□

References

1. Ambainis, A., Balodis, K., Iraids, J., Ozols, R., Smotrovs, J.: Parameterized quantum query complexity of graph collision. CoRR abs/1305.1021 (2013). http://arxiv.org/abs/1305.1021

2. Belovs, A.: Span-program-based quantum algorithm for the rank problem. CoRR abs/1103.0842 (2011). http://arxiv.org/abs/1103.0842

3. Belovs, A.: Learning-graph-based quantum algorithm for k-distinctness. In: IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 207–216. IEEE (2012). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6375298

4. Belovs, A.: Span programs for functions with constant-sized 1-certificates. In: Proceedings of the 44th Symposium on Theory of Computing, pp. 77–84. ACM (2012). http://dl.acm.org/citation.cfm?id=2213985

5. Belovs, A., Reichardt, B.W.: Span programs and quantum algorithms for st-connectivity and claw detection. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 193–204. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-33090-2_18

6. Berzina, A., Dubrovsky, A., Freivalds, R., Lace, L., Scegulnaja, O.: Quantum query complexity for some graph problems. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 140–150. Springer, Heidelberg (2004). http://dx.doi.org/10.1007/978-3-540-24618-3_11

7. Dürr, C., Heiligman, M., Høyer, P., Mhalla, M.: Quantum query complexity of some graph problems. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 481–493. Springer, Heidelberg (2004). http://dx.doi.org/10.1007/978-3-540-27836-8_42

8. Furrow, B.: A panoply of quantum algorithms. Quantum Info. Comput. **8**(8), 834–859 (2008). http://dl.acm.org/citation.cfm?id=2017011.2017022

9. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of the Eighth Annual Structure in Complexity Theory Conference, pp. 102–111. IEEE (1993). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=336536

10. Reichardt, B.W.: Span programs and quantum query complexity: the general adversary bound is nearly tight for every boolean function. In: 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 544–551. IEEE (2009). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5438598

11. Reichardt, B.W.: Reflections for quantum query algorithms. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 560–569. SIAM (2011). http://dl.acm.org/citation.cfm?id=2133080

12. Reichardt, B.W., Spalek, R.: Span-program-based quantum algorithm for evaluating formulas. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 103–112. ACM, New York (2008). http://doi.acm.org/10.1145/1374376.1374394