Quantum Algorithms for One-Sided Crossing Minimization

Susanna Caroppo
Roma Tre University, Rome, Italy

Giordano Da Lozzo
Roma Tre University, Rome, Italy

Giuseppe Di Battista
Roma Tre University, Rome, Italy

Abstract

We present singly-exponential quantum algorithms for the ONE-SIDED CROSSING MINIMIZATION (OSCM) problem. We show that OSCM can be viewed as a set problem amenable for exact algorithms with a quantum speedup with respect to their classical counterparts. First, we exploit the quantum dynamic programming framework of Ambainis et al. [*Quantum Speedups for Exponential-Time Dynamic Programming Algorithms.* SODA 2019] to devise a QRAM-based algorithm that solves OSCM in $\mathcal{O}^*(1.728^n)$ time and space. Second, we use quantum divide and conquer to obtain an algorithm that solves OSCM without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph theory; Theory of computation → Quantum complexity theory

Keywords and phrases One-sided crossing minimization, quantum graph drawing, quantum dynamic programming, quantum divide and conquer, exact exponential algorithms

Digital Object Identifier 10.4230/LIPIcs.GD.2024.20

Related Version *Full Version*: `https://arxiv.org/abs/2409.01942` [3]

Funding This research was supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND).

Acknowledgements We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support.

1 Introduction and Preliminaries

We study, from the quantum perspective, the ONE-SIDE CROSSING MINIMIZATION (OSCM) problem, one of the most studied problems in Graph Drawing, which is defined below.

2-Level Drawings. In a 2-level drawing of a bipartite graph the vertices of the two sets of the bipartition are placed on two horizontal lines and the edges are drawn as straight-line segments. The number of crossings of the drawing is determined by the order of the vertices on the two lines. Formally, let $G = (U, V, E)$ be a bipartite graph, where U and V are the two parts of the vertex set of G and $E \subseteq U \times V$ is the edge set of G. In the following, we write n, n_U, and n_V for $|U \cup V|$, $|U|$, and $|V|$, respectively; also, for every integer h, we use the notation $[h]$ to refer to the set $\{1, \dots, h\}$. A *2-level drawing* of G is a pair (π_U, π_V), where $\pi_U : U \leftrightarrow \{1, \dots, |U|\}$ is a linear ordering of U and $\pi_V : V \leftrightarrow \{1, \dots, |V|\}$ is a linear ordering of V. We denote the vertices of U by u_i ($i \in [n_U]$), and the vertices of V by v_j ($j \in [n_V]$). Two edges (u_1, v_1) and (u_2, v_2) in E *cross* in (π_U, π_V) if: (i) $u_1 \neq u_2$ and $v_1 \neq v_2$ and (ii) either $\pi_U(u_1) < \pi_U(u_2)$ and $\pi_V(v_2) < \pi_V(v_1)$, or $\pi_U(u_2) < \pi_U(u_1)$ and $\pi_V(v_1) < \pi_V(v_2)$. The number of crossings of a 2-level drawing (π_U, π_V) is the number $cr(G, \pi_U, \pi_V)$ of distinct (unordered) pairs of edges that cross. Problem OSCM is defined as follows:

One-Sided Crossing Minimization (OSCM)

Input: A bipartite graph $G = (U, V, E)$ and a linear ordering $\pi_U : U \leftrightarrow [n_U]$.
Output: A linear ordering $\pi_V : V \leftrightarrow [n_V]$ such that $cr(G, \pi_U, \pi_V)$ is minimum.

State of the art. The importance of the OSCM problem, which is NP-complete [10] even for sparse graphs [19], in Graph Drawing was first put in evidence by Sugiyama in [22].

Exact solutions of OSCM have been searched with branch-and-cut techniques, see e.g. [16, 20, 24], and with FPT algorithms. The parameterized version of the problem, with respect to its natural parameter $k = \min_{\pi_V} cr(G, \pi_U, \pi_V)$, has been widely investigated. Dujmovic et al. [7, 8] were the first to show that OSCM can be solved in $f(k)n^{O(1)}$ time, with $f \in O(\psi^k)$, where $\psi \approx 1.6182$ is the golden ratio. Subsequently, Dujmovic and Whitesides [5, 6] improved the running time to $\mathcal{O}(1.4656^k + kn^2)$. Fernau et al. [11], exploiting a reduction to weighted FAST and the algorithm by Alon et al. [1], gave a subexponential parameterized algorithm with running time $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$. The reduction also gives a PTAS using [17]. Kobayashi and Tamaki [18] gave the current best FPT result with running time $\mathcal{O}(k2^{\sqrt{2k}} + n)$.

Quantum Graph Drawing has recently gained popularity. Caroppo et al. [4] applied Grover's search [14] to several Graph Drawing problems obtaining a quadratic speedup over classical exhaustive search. Fukuzawa et al. [12] studied how to apply quantum techniques for solving systems of linear equations [15] to Tutte's algorithm for drawing planar 3-connected graphs [23]. Recently, in a paper that pioneered *Quantum Dynamic Programming*, several vertex ordering problems related to Graph Drawing have been tackled by Ambainis et al. [2].

Our Results. First, we exploit the quantum dynamic programming framework of Ambainis et al. to devise an algorithm that solves OSCM in $\mathcal{O}^*(1.728^n)$ time and space. We compare the performance of our algorithm against the algorithm proposed in [18], based on the value of k. We have that the quantum algorithm performs asymptotically better than the FPT algorithm, when $k \in \Omega(n^2)$. Second, we use quantum divide and conquer to obtain an algorithm that solves OSCM using $\mathcal{O}^*(2^n)$ time and polynomial space. Both our algorithms improve the corresponding classical bounds in either time or space or both.

In our first result, we adopt the QRAM (quantum random access memory) model of computation [13], which allows (i) accessing quantum memory in superposition and (ii) invoking any T-time classical algorithm that uses a (classic) random access memory as a subroutine spending time $\mathcal{O}(T)$. In the second result we do not use the QRAM model of computation since we do not need to explicitly store the results obtained in partial computations.

Notation. For ease of notation, given positive integers a and b, we denote $\lceil \frac{a}{b} \rceil$ as $\frac{a}{b}$ and $\lceil \log a \rceil$ as $\log a$. If $f(n) = \mathcal{O}(n^c)$ for some constant c, we will write $f(n) = poly(n)$. In case $f(n) = d^n poly(n)$ for some constant d, we use the notation $f(n) = \mathcal{O}^*(d^n)$ (see, e.g., [25]).

Quantum Tools. We will use quantum search primitives, such as the one of Theorem 1, and exploit the fact that they can perform condition checking on data stored in QRAM.

▶ **Theorem 1** (Quantum Minimum Finding, QMF [9]). *Let $f : D \to C$ be a polynomial-time computable function, whose domain D has size N and whose codomain C is a totally ordered set (such as $\mathbb{N}$) and let $\mathcal{F}$ be a procedure that computes f. There exists a bounded-error quantum algorithm that finds $x \in D$ such that $f(x)$ is minimized using $\mathcal{O}(\sqrt{N})$ applications of $\mathcal{F}$.*

Because of space limitations, some proofs are sketched or omitted. They can be found in the full version of the paper [3].

Algorithm 1 Procedure `QuantumDP` is the algorithm of Lemma 2. Procedure `OPT` is a recursive procedure invoked by `QuantumDP`. Procedure `QMF` performs quantum minimum finding.

```
1:  procedure QUANTUMDP($X$)
2:      Input: Set $X$ of size $n$; Output: the value $OPT_{\mathcal{P}}(X)$.
3:      for all sets $W \subset X$ such that $|W| \leq (1-\alpha)n/4$ do     ▷ in order of increasing size
4:          Compute $OPT_{\mathcal{P}}(W)$ classically via dynamic programming     ▷ use Equation (1)
5:                                                                             with $k = |W| - 1$
6:          Store $OPT_{\mathcal{P}}(W)$ in QRAM
7:      end for
8:      return OPT(X)
9:  end procedure
10: procedure OPT($S$)
11:     Input: Subset $S \subseteq X$; Output: the value $OPT_{\mathcal{P}}(S)$.
12:     if $|S| \leq (1-\alpha)n/4$ then
13:         return value $OPT_{\mathcal{P}}(S)$ stored in QRAM
14:     else
15:         return the result of QMF over all $S \subset X$ to find
                $$\min_{W \subset S, |W| = \frac{|S|}{2}} \{\mathtt{OPT}(W) + \mathtt{OPT}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$
16:     end if
17: end procedure
```

2 Quantum Dynamic Programming for One-Sided Crossing Minimization

In this section, we first describe the quantum dynamic programming framework of Ambainis et al. [2], which is applicable to numerous optimization problems involving sets. Then, we show that OSCM is a set problem over V that falls within this framework. We use this fact to derive a quantum algorithm (Theorem 4) exhibiting a speedup over the corresponding classical singly-exponential algorithm (given in [3]) in both time and space complexity.

Quantum dynamic programming for set problems. The framework by Ambainis et al. is defined by the following lemma derivable from [2].

▶ **Lemma 2.** *Let $\mathcal{P}$ be an optimization problem (say a minimization problem) over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S|-1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\} \quad (1)$$

Then, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm that uses QRAM in $\mathcal{O}^*(1.728^n)$ time and space.

Proof sketch. The algorithm for the proof of the lemma is presented as Algorithm 1. The main idea of the algorithm is to precompute solutions for smaller subsets using classical dynamic programming and then recombine the results of the precomputation step to obtain the optimal solution for the whole set (recursively) applying QMF (see Theorem 1). ◀

Quantum dynamic programming for OSCM. In the following, let (G, π_U) be an instance of OSCM. For any $S \subseteq E$, consider the subgraph $H = (U, V, S)$ of G. For ease of notation, we denote $cr(H, \pi_U, \pi_V)$ simply as $cr_S(\pi_U, \pi_V)$. Also, let π_V be a linear ordering of the vertices in V and $V_1, V_2 \subseteq V$ be two subsets of the vertices of V such that $V_1 \cap V_2 = \emptyset$. We say that V_1 *precedes* V_2 in π_V, denoted as $V_1 \prec_{\pi_V} V_2$, if for any $v_1 \in V_1$ and $v_2 \in V_2$, it holds that $\pi_V(v_1) < \pi_V(v_2)$. Also, for a any $W \subseteq V$, we denote by $E(W)$ the subset of E defined a follows $E(W) := \{(u_a, v_b) : (u_a, v_b) \in E \wedge v_b \in W\}$. We will exploit the following.

▶ **Lemma 3.** *Let $G = (U, V, E)$ be a bipartite graph and let $\pi_U : U \leftrightarrow [n_U]$ be a linear ordering of the vertices of U. Also, let $V_1, V_2 \subseteq V$ be two subsets of the vertices of V such that $V_1 \cap V_2 = \emptyset$. Then, there exists a constant $\gamma(\pi_U, V_1, V_2)$ such that, for every linear ordering $\pi_V : V \leftrightarrow [n_V]$ with $V_1 \prec_{\pi_V} V_2$ we have that:*

$$\gamma(\pi_U, V_1, V_2) = cr_{E(V_1) \cup E(V_2)}(\pi_U, \pi_V) - cr_{E(V_1)}(\pi_U, \pi_V) - cr_{E(V_2)}(\pi_U, \pi_V) \tag{2}$$

Observe that, given an ordering π_V of V such that V_1 precedes V_2 in π_V, the value $\gamma(\pi_U, V_1, V_2)$ represents the number of crossings in a 2-level drawing (π_U, π_V) of G determined by pairs of edges, one belonging to $E(V_1)$ and the other belonging to $E(V_2)$.

We are now ready to derive our dynamic programming quantum algorithm for OSCM. To this aim, we argue next that the framework of Lemma 2 can be applied to the optimization problem associated with OSCM (i.e., computing the minimum number of crossings over all 2-level drawings (π_U, π_V) of G with π_U fixed), which we call MinOSCM. First, we have that MinOSCM is a set problem over V, whose optimal solution respects a recurrence of the same form as Equation (1). In fact, for a subset S of V, let $OPT(S)$ denote the minimum number of crossings in a 2-level drawing (π_U, π_S) of the graph $G_S = (U, S, E(S))$, where $\pi_S : S \leftrightarrow [|S|]$ is a linear ordering of the vertices of S. Then, by Lemma 3, we can compute $OPT(S)$ by means of the following recurrence for any $k \in [|S|-1]$:

$$OPT(S) = \min_{W \subset S, |W|=k} \{OPT(W) + OPT(S \setminus W) + \gamma(\pi_U, W, S \setminus W)\}$$

Clearly, $OPT(V)$ corresponds to the optimal solution for (G, π_U). Also, function γ plays the role of function $f_{\mathcal{P}}$ of Lemma 2. Second, we have that γ can be computed in $poly(n)$ time.

In [3], we show that Algorithm 1 applied to MinOSCM can also be adapted to compute an ordering π_V of V that yields a drawing with the minimum number of crossings.

Altogether, we have finally proved the following.

▶ **Theorem 4.** *There is a bounded-error quantum algorithm that solves OSCM in $O^*(1.728^{n_V})$ time and space.*

Comparing Theorem 4 against the current best FPT result [18] solving OSCM in $\mathcal{O}(k2^{\sqrt{2k}} + n)$ time, where k bounds the number of allowed crossings, we have the following.

▶ **Corollary 5.** *The algorithm of Theorem 4 is asymptotically more time-efficient than the FPT algorithm parameterized by the number k of crossings in [18] when $k \in \Omega(n^2)$.*

3 Quantum Divide and Conquer for One-Sided Crossing Minimization

Shimizu and Mori [21] used divided and conquer to obtain quantum exponential-time polynomial-space algorithms for coloring problems that do not rely on the use of QRAM. In this section, we first generalize their ideas to obtain a framework designed to speedup, without using QRAM, some classical exponential-time polynomial-space divide and conquer

Algorithm 2 The quantum algorithm of Lemma 6.

```
1: procedure QuantumDC($X$):
2:     Input: Set $X$ of size $n$; Output: the value $OPT_{\mathcal{P}}(X)$.
3:     if $|S| \leq c_{\mathcal{P}}$ then
4:         return $f_{\mathcal{P}}(S, \emptyset)$
5:     end if
6:     return the result of QMF over all $W \subset S$ with $|W| = \frac{|S|}{2}$ to find
           $\min_{W \subset S, |W| = \frac{|S|}{2}} \{\texttt{QuantumDC}(W) + \texttt{QuantumDC}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$
7: end procedure
```

algorithms for set problems. Then, we show that OSCM is a set problem over V that falls within this framework. We use this fact to derive a quantum algorithm (Theorem 7) that improves the time bounds of the corresponding classical singly-exponential algorithm (given in the full version of the paper [3]), while maintaining polynomial space complexity.

Quantum divide and conquer for set problems. In the remainder, we provide a general quantum framework, defined by the following lemma.

▶ **Lemma 6.** *Let $\mathcal{P}$ be an optimization problem (say a minimization problem) over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ and a constant $c_{\mathcal{P}}$ such that, for any $S \subseteq X$, it holds that:* **(i)** *If $|S| \leq c_{\mathcal{P}}$, then $OPT_{\mathcal{P}}(S) = f_{\mathcal{P}}(S, \emptyset)$;* **(ii)** *If $|S| > c_{\mathcal{P}}$, then*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = \frac{|S|}{2}} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\} \qquad (3)$$

We have that, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

Proof. The algorithm for the proof of the lemma is presented as Algorithm 2 and is based on the recurrence in Equation (3). It works recursively as follows. If the input set X is sufficiently small, i.e., $|X| \leq c_{\mathcal{P}}$, then the optimal value for X is computed directly as $f_{\mathcal{P}}(X, \emptyset)$. Otherwise, it uses QMF to find the optimal pair $(S, X \setminus S)$ that achieves $OPT_{\mathcal{P}}(X)$ according to Equation (3), where $OPT_{\mathcal{P}}(S)$ and $OPT_{\mathcal{P}}(X \setminus S)$ have been recursively computed.

The running time $Q(k)$ of Algorithm 2 when $|X| = k$ obeys the following recurrence:

$$Q(k) \leq \sqrt{\mathcal{O}\left(\binom{k}{k/2}\right)} \Big(Q(\lfloor k/2 \rfloor) + Q(k/2) + poly(k)\Big)$$

Hence, $Q(k) \leq 2^k poly(k)$, and the total running time of Algorithm 2 is bounded by $\mathcal{O}^*(2^n)$.

The space complexity of Algorithm 2 (procedure `QuantumDC`) can be proved polynomial as follows; see Figure 1. The execution of `QuantumDC` determines a rooted binary tree $\mathcal{T}$ whose nodes are associated with its recursive calls; see Figure 2. Each call corresponds to a circuit in Figure 1. We denote by `QDC(i,j)` the circuit, at the i^{th}-level of the recursion tree $\mathcal{T}$, with $i = 0, \dots, \log n - 1$, associated with the j^{th}-call, with $j \in 0, \dots, 2^i - 1$. The input to each of such circuits consists of a set of registers defined as follows. For $i = 0, 1, \dots, \log n - 1$ and $j = 0, 1, \dots, 2^i - 1$, there exists a register $A_{i,j}$ with $\frac{n}{2^i}$ qubits. It stores a superposition

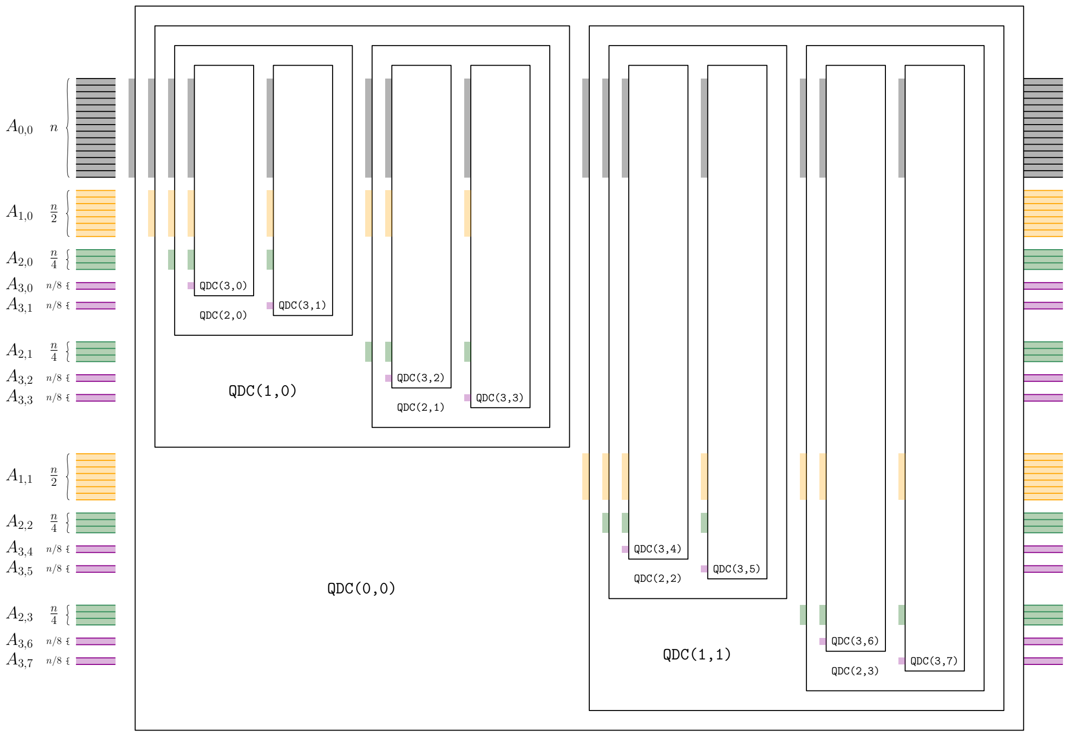

Figure 1 Schematic representation of the circuit realizing Algorithm 2 for a set X with $n = 16$. The qubits in $L_{i,j}$ in input to the circuit QDC(i,j) are incident to its left boundary.

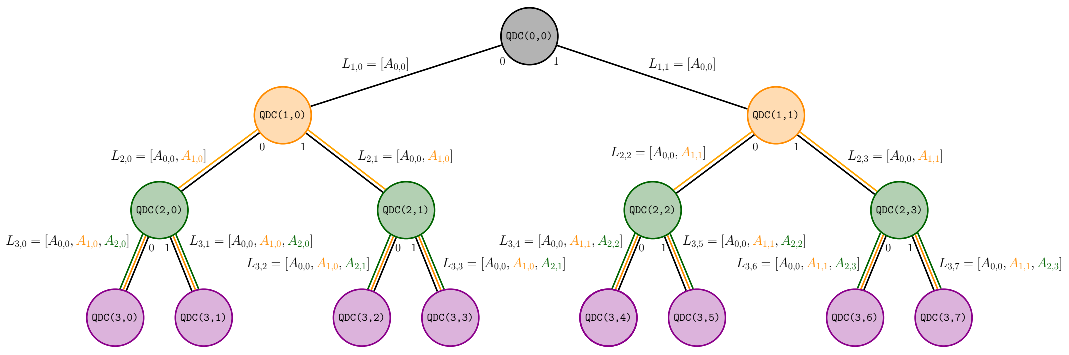
QDC(0,0)
$L_{1,0} = [A_{0,0}]$
$L_{1,1} = [A_{0,0}]$
QDC(1,0)
QDC(1,1)
$L_{2,0} = [A_{0,0}, A_{1,0}]$
$L_{2,1} = [A_{0,0}, A_{1,0}]$
$L_{2,2} = [A_{0,0}, A_{1,1}]$
$L_{2,3} = [A_{0,0}, A_{1,1}]$
QDC(2,0)
QDC(2,1)
QDC(2,2)
QDC(2,3)
$L_{3,0} = [A_{0,0}, A_{1,0}, A_{2,0}]$
$L_{3,1} = [A_{0,0}, A_{1,0}, A_{2,0}]$
$L_{3,2} = [A_{0,0}, A_{1,0}, A_{2,1}]$
$L_{3,3} = [A_{0,0}, A_{1,0}, A_{2,1}]$
$L_{3,4} = [A_{0,0}, A_{1,1}, A_{2,2}]$
$L_{3,5} = [A_{0,0}, A_{1,1}, A_{2,2}]$
$L_{3,6} = [A_{0,0}, A_{1,1}, A_{2,3}]$
$L_{3,7} = [A_{0,0}, A_{1,1}, A_{2,3}]$
0
1
QDC(3,0)
QDC(3,1)
QDC(3,2)
QDC(3,3)
QDC(3,4)
QDC(3,5)
QDC(3,6)
QDC(3,7)

Figure 2 The tree $\mathcal{T}$ whose nodes are associated with the recursive calls of Algorithm 2.

corresponding to a subset $S_{i,j}$ of X (to be defined later) of size $\frac{n}{2^i}$, which represents all possible ways of splitting the subset into two equal-sized subsets. Specifically, a status 0 for $A_{i,j}[k]$ corresponds to assigning the k^{th}-element of the subset associated with $A_{i,j}$ to one side of the split, while a status 1 of $A_{i,j}[k]$ corresponds to assigning the k^{th}-element of such a subset to the other side of the split. In Figure 2, we associate the split defined by the status-0 qubits (by the status-1 qubits) with the left (right) child of a node. Moreover, in Figure 2, each edge of $\mathcal{T}$ is labeled with the registers representing the corresponding split.

The input of QDC(i,j) is a set $L_{i,j}$ of $i+1$ registers of size n, $\frac{n}{2}$, $\frac{n}{4}$, $\ldots$, $\frac{n}{2^i}$, respectively; see Figure 1. The registers in input to QDC(i,j) can be recursively defined as follows. The register $A_{i-1,\lfloor j/2 \rfloor}$ belongs to $L_{i,j}$ and it is the smallest register in this set. Also, if $A_{c,d}$ with $c \geq 1$ belongs to $L_{i,j}$, then $A_{c-1,\lfloor d/2 \rfloor}$ also belong to $L_{i,j}$. In particular, observe that $L_{i,j}$

always contains $A_{0,0}$. The circuit `QDC(i,j)` solves problem $\mathcal{P}$ on a subset $S_{i,j}$ of X of size $\frac{n}{2^i}$, which is defined by the states of the registers in $L_{i,j}$. In particular, the set $S_{i,j}$ can be determined by following the path of $\mathcal{T}$ connecting `QDC(i,j)` to the root, and observing that the parity of j determines whether a node in the path is the left or right child of its parent.

We can finally bound the space complexity of Algorithm 2, in terms of both bits and qubits. Since our algorithm does not rely on external classic memory, we only need to bound the latter. Note that the number of circuits `QDC(i,j)` (which are in a bijection with the nodes of $\mathcal{T}$) is linear in n and that the number of qubits in $L_{i,j}$, which form the input of `QDC(i,j)`, is at most $\sum_{i=0}^{\log n} \frac{n}{2^i} = 2n$. Hence, the space complexity of Algorithm 2 is polynomial. ◀

Quantum divide and conquer for OSCM. We now describe a quantum divide and conquer algorithm for OSCM. We start by showing that the framework of Lemma 6 can be applied to MinOSCM (see Section 2). This can be done in a similar fashion as for the Lemma 2. In particular, the fact that the MinOSCM problem is a set problem over V immediately follows from the observation that Equation (3) is the restriction of Equation (1) to the case in which $k = |W| = \frac{|S|}{2}$. Moreover, recall that γ can be computed in *poly*(n) time. The execution of Algorithm 2 produces as output a superposition of the registers $A_{i,j}$ such that the state with the highest probability of being returned, if measured, corresponds to an ordering π_V of V that yields a drawing with the minimum number of crossings. In [3], we show how to obtain π_V from such a state. Altogether we have proved the following.

▶ **Theorem 7.** *There is a bounded-error quantum algorithm that solves OSCM in $\mathcal{O}^*(2^{n_V})$ time and polynomial space.*

4 Conclusions

We presented singly-exponential quantum algorithms for OSCM, exploiting both quantum dynamic programming and quantum divide and conquer. We believe that this research will spark further interest in the design of exact quantum algorithms for hard graph drawing problems. In [3], we highlight two meaningful applications of our results.

References

1 Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2009. `doi:10.1007/978-3-642-02927-1_6`.

2 Andris Ambainis, Kaspars Balodis, Janis Iraids, Martins Kokainis, Krisjanis Prusis, and Jevgenijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1783–1793. SIAM, 2019. `doi:10.1137/1.9781611975482.107`.

3 Susanna Caroppo, Giordano Da Lozzo, and Giuseppe Di Battista. Quantum algorithms for one-sided crossing minimization, 2024. `arXiv:2409.01942`.

4 Susanna Caroppo, Giordano Da Lozzo, and Giuseppe Di Battista. Quantum graph drawing. In Ryuhei Uehara, Katsuhisa Yamanaka, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation - 18th International Conference and Workshops on Algorithms and Computation, WALCOM 2024, Kanazawa, Japan, March 18-20, 2024, Proceedings*, volume 14549 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2024. `doi:10.1007/978-981-97-0566-5_4`.

5 Vida Dujmovic, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. In Giuseppe Liotta, editor, *Graph Drawing, 11th International Symposium, GD 2003, Perugia, Italy, September 21-24, 2003, Revised Papers*, volume 2912 of *Lecture Notes in Computer Science*, pages 332–344. Springer, 2003. doi:10.1007/978-3-540-24595-7_31.

6 Vida Dujmovic, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. Discrete Algorithms*, 6(2):313–323, 2008. doi:10.1016/J.JDA.2006.12.008.

7 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Graph Drawing, 10th International Symposium, GD 2002, Irvine, CA, USA, August 26-28, 2002, Revised Papers*, volume 2528 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2002. doi:10.1007/3-540-36151-0_12.

8 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, 2004. doi:10.1007/S00453-004-1093-2.

9 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *CoRR*, quant-ph/9607014, 1996. arXiv:quant-ph/9607014.

10 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.

11 Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Matthias Mnich, Geevarghese Philip, and Saket Saurabh. Ranking and drawing in subexponential time. In Costas S. Iliopoulos and William F. Smyth, editors, *Combinatorial Algorithms - 21st International Workshop, IWOCA 2010, London, UK, July 26-28, 2010, Revised Selected Papers*, volume 6460 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2010. doi:10.1007/978-3-642-19222-7_34.

12 Shion Fukuzawa, Michael T. Goodrich, and Sandy Irani. Quantum Tutte embeddings. *CoRR*, abs/2307.08851, 2023. doi:10.48550/arXiv.2307.08851.

13 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, April 2008. doi:10.1103/PhysRevLett.100.160501.

14 Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *STOC 1996*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866.

15 Aram W. Harrow. Quantum algorithms for systems of linear equations. In *Encyclopedia of Algorithms*, pages 1680–1683. Springer New York, 2016. doi:10.1007/978-1-4939-2864-4_771.

16 Michael Jünger and Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In Franz-Josef Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, volume 1027 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 1995. doi:10.1007/BFB0021817.

17 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 95–103. ACM, 2007. doi:10.1145/1250790.1250806.

18 Yasuaki Kobayashi and Hisao Tamaki. A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. *Algorithmica*, 72(3):778–790, 2015. doi:10.1007/S00453-014-9872-X.

19 Xavier Muñoz, Walter Unger, and Imrich Vrto. One sided crossing minimization is NP-hard for sparse graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001, Revised Papers*, volume 2265 of *Lecture Notes in Computer Science*, pages 115–123. Springer, 2001. doi:10.1007/3-540-45848-4_10.

20 Petra Mutzel and René Weiskircher. Two-layer planarization in graph drawing. In Kyung-Yong Chwa and Oscar H. Ibarra, editors, *Algorithms and Computation, 9th International Symposium, ISAAC '98, Taejon, Korea, December 14-16, 1998, Proceedings*, volume 1533 of *Lecture Notes in Computer Science*, pages 69–78. Springer, 1998. doi:10.1007/3-540-49381-6_9.

21 Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. *Algorithmica*, 84(12):3603–3621, 2022. doi:10.1007/S00453-022-00976-2.

22 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

23 William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.

24 Vicente Valls, Rafael Martí, and Pilar Lino. A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs. *European journal of operational research*, 90(2):303–319, 1996. doi:10.1016/0377-2217(95)00356-8.

25 Gerhard J. Woeginger. Open problems around exact algorithms. *Discret. Appl. Math.*, 156(3):397–405, 2008. doi:10.1016/J.DAM.2007.03.023.