Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 29, no. 2, pp. 3–47 (2025)
DOI: 10.7155/jgaa.v29i2.3039

Quantum Graph Drawing

Susanna Caroppo *Giordano Da Lozzo* *Giuseppe Di Battista*

Department of Engineering, Roma Tre University, Rome, Italy

Submitted: May 2024 Accepted: April 2025 Published: May 2025

Article type: Regular Communicated by: R. Uehara, K. Yamanaka, and H.-C. Yen

Abstract. In this paper, we initiate the study of quantum algorithms in the Graph Drawing research area. We focus on two foundational drawing standards: *2-level drawings* and *book layouts*. Concerning 2-level drawings, we consider the problems of obtaining drawings with the minimum number of crossings, k-planar drawings, quasi-planar drawings, and the problem of removing the minimum number of edges to obtain a 2-level planar graph. Concerning book layouts, we consider the problems of obtaining 1-page book layouts with the minimum number of crossings, book embeddings with the minimum number of pages, and the problem of removing the minimum number of edges to obtain an outerplanar graph. We explore both the quantum circuit and the quantum annealing models of computation. In the *quantum circuit model*, we provide an algorithmic framework based on Grover's quantum search, which allows us to obtain, ignoring polynomial terms, a quadratic speedup on the best known classical exact algorithms for all the considered problems. In the *quantum annealing model*, we perform experiments on the quantum processing unit provided by D-Wave, focusing on the classical 2-level crossing minimization problem, demonstrating that quantum annealing is competitive with respect to classical algorithms.

Keywords: Quantum complexity, Grover's algorithm, QUBO, Quantum annealing, 2-Level drawings, Book layouts

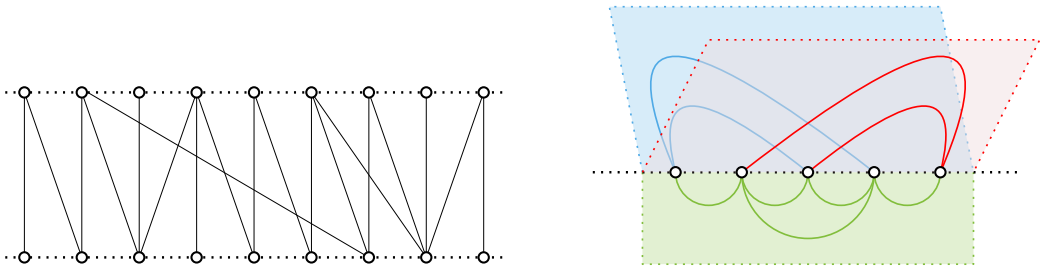
Research partially funded by the European Union, Next Generation EU, Mission 4, Component 1, CUP C53D23003680006 PRIN project no. 2022TS4Y3N "EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data", and CUP J53D23007130006 PRIN project no. 2022ME9Z78 "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization". We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support. A preliminary version of this research appeared in [13].

E-mail addresses: susanna.caroppo@uniroma3.it (Susanna Caroppo) giordano.dalozzo@uniroma3.it (Giordano Da Lozzo) giuseppe.dibattista@uniroma3.it (Giuseppe Di Battista)

1 Introduction

We initiate the study of quantum algorithms in the Graph Drawing research area[1]. The exploration of quantum approaches to address graph-drawing problems that are computationally challenging for classical methods is appealing from both theoretical and practical perspectives. On the one hand, leveraging quantum speedups, such as those provided by Grover's search, could enable the analysis of larger and more complex graphs within reasonable timeframes. On the other hand, the Graph Drawing area offers new opportunities to apply emerging and practical quantum technologies, such as Quantum Annealing, to combinatorial optimization problems in both geometric and topological graph theory.

The problems. We focus on two foundational graph drawing standards: *2-level drawings* and *book layouts*. In a *2-level drawing*, the graph is bipartite, the vertices are placed on two horizontal lines, and the edges are drawn as y-monotone curves; see Fig. 1a. In this drawing standard, we consider the search version of the Two-Level Crossing Minimization (TLCM) problem, where given an integer ρ we seek a 2-level drawing with at most ρ crossings, and of the Two-Level Skewness (TLS) problem, where given an integer σ we seek to determine a set of σ edges whose removal yields a *2-level planar graph*, i.e., a forest of caterpillars [19]. The minimum value of σ is the *2-level skewness* of the considered graph. We also consider the Two-Level Quasi Planarity (TLQP) problem, where we seek a drawing in which no three edges pairwise cross, i.e., a *quasi-planar* drawing, and the Two-Level k-Planarity (TLKP) problem, where we seek a drawing in which each edge participates in at most k crossings, i.e., a *k-planar* drawing. In a *book layout*, the drawing is constructed using a collection of half-planes, called *pages*, all having the same line, called *spine*, as their boundary; see Fig. 1b. The vertices lie on the spine and each edge is drawn on a page. In this drawing standard, we consider the search version of the One-Page Crossing Minimization (OPCM) problem, where given an integer ρ we seek a 1-page layout with at most ρ crossings; the Book Thickness (BT) problem, where we search a τ-page layout where the edges in the same page do not cross, i.e., a *τ-page book embedding*; and the Book Skewness (BS) problem, where given an integer σ we seek a set of σ edges whose removal yields a graph admitting a *1-page book embedding*, i.e., it is outerplanar [9]. The minimum value of σ is the *book skewness* of the considered graph.



(a) (b)

Figure 1: Examples of graph drawing standards: (a) a 2-level drawing and (b) a 3-page book embedding.

[1] Simultaneously to this research [13], Fukuzawa, Goodrich, and Irani [21] have formulated a model for quantum graph drawing in the circuit model of computation, showing how to use Harrow's quantum algorithm [26] for solving systems of linear equations to compute a so-called Tutte embedding of a 3-connected planar graph.

The models. We delve into both the *quantum circuit* [35, 37] and the *quantum annealing* [33] models of computation. In the former, quantum gates are used to compose a circuit that transforms an input superposition of qubits into an output superposition. The circuit design depends on both the problem and the specific instance being processed. The output superposition is eventually measured, obtaining the solution with a certain probability. The quality of the circuit is measured in terms of its *circuit complexity*, i.e., the number of elementary gates it contains, of its *depth*, i.e., the maximum length of a chain of elementary gates from the input to the output, and of its *width*, i.e., the maximum number of elementary gates "along a cut" separating the input from the output. It is natural to upper bound the time complexity of the execution of a quantum circuit either by its depth, assuming the gates at each layer can be executed in parallel, or by its circuit complexity, assuming the gates are executed sequentially. The width estimates the desired level of parallelism. In the quantum annealing model of computation, the quantum annealing processors, in general quite different from those designed for the quantum circuit model, consist of a fixed-topology network, whose vertices correspond to qubits and whose edges correspond to possible interactions between qubits. A problem is mapped to an embedding on such a topology. During the computation, the solution space of a problem is explored, searching for minimum-energy states, which correspond to, in general approximate, solutions.

Our contributions. In the quantum circuit model, we first show that the above graph drawing problems can be described by means of quantum circuits. The first problem that we have to solve is choosing an effective representation for inputs and outputs. Since all problems we tackle require the selection of a permutation of n vertices, a tempting idea is to have as inputs-outputs binary variables explicitly representing the precedence between pairs of vertices. However, this requires to represent the superposition of a quadratic, in n, number of qubits. Hence, we use as inputs-outputs the vertex coordinates, which implicitly represent a permutation and require just $n \log n$ qubits. On the other hand, to solve the above problems, we have to transform the coordinates into orderings. Thus, the first contribution of the paper is a set of efficient quantum methods, that can be of general usage in *Quantum Graph Drawing,* that allow to transform coordinates of vertices into precedence between vertices and vice versa. These methods use a small number of "ancilla" qubits. Second, we present an algorithmic framework based on Grover's quantum search [23]. This framework enables us to achieve, ignoring polynomial terms, a quadratic speedup compared to the best *known*[2] exact classical algorithms for all the problems under consideration. Table 1 summarizes our complexity results and compares them with classical algorithms. Within this framework we introduce quantum phase inversion methods composed by building blocks suitable to be combined to solve several types of graph drawing problems.

In the quantum annealing model, we focus on the actual processing unit provided by D-Wave, which allows us to perform hybrid computations, which are partly classical and partly quantum. We first show that it is relatively easy to use D-Wave for implementing heuristics for the above problems. Second, we focus on the classical TLCM problem. Through experiments, we demonstrate that quantum annealing exhibits competitiveness when compared to current classical algorithms. Table 3 and Fig. 30 show our experimental findings. To ensure reproducibility, the codebase and datasets used in the conducted experiments have been made available at [14].

[2]We compare our running time with a classical algorithm that enumerates and checks all possible candidate solution, which is, to the best of our knowledge, the current state of the art for all of the considered problems.

Table 1: Results presented in this paper and comparison with exact algorithms. FPT algorithms are mentioned, if any, only with respect to the natural parameter. CC stands for Circuit Complexity. M denotes the number of solutions[3].

Problem	Classic Algorithm Running Time	Upperbound for m	FPT Time	Quantum Oracle Calls	Oracles (Lemma 4)		
					CC	Depth	Width
TLCM	$2^{n \log n} O(n^2)$	$O(\sqrt[3]{\rho \cdot n^2})$ [3, 4]	$2^{O(\rho)} + n^{O(1)}$ [31]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n}}{M}}$	$O(m^2)$	$O(n^2)$	$O(m^2)$
TLKP	$2^{n \log n} O(n^2)$	$O(\sqrt{k} \cdot n)$ [3, 4]	-	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n}}{M}}$	$O(m^2)$	$O(m \log^2 m)$	$O(m)$
TLQP	$2^{n \log n} O(n)$	$O(n)$ [4]	Para-NP-hard [2]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n}}{M}}$	$O(m^6)$	$O(m^4)$	$O(m^2)$
TLS	$O(m^\sigma n)$	$O(n + \sigma)$	$2^{O(\sigma^3)} n$ [18]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}}$	$O(m^2)$	$O(m)$	$O(m)$
OPCM	$2^{n \log n} O(n^2)$	$O(\sqrt[3]{\rho \cdot n^2})$ [38]	Courcelle's Th. [6, 7]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n}}{M}}$	$O(n^8)$	$O(n^6)$	$O(m^2)$
BT	$2^{n \log n + m \log \tau} O(\tau n)$	$O(\tau \cdot n)$	Para-NP-hard [32]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n + m \log \tau}}{M}}$	$O(n^8)$	$O(n^6)$	$O(m)$
BS	$O(m^\sigma n)$	$O(n + \sigma)$	Courcelle's Th. [15, 16]	$\frac{\pi}{4}\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}}$	$O(n^8)$	$O(n^6)$	$O(m)$

State of the art. We now provide an overview of the complexity status of each of the considered problems, together with the existence of FPT algorithms with respect to the corresponding natural parameter (total number of crossings ρ, number of crossings per edge k, maximum number of allowed mutually crossing edges, number of pages τ, and number of edges to be removed σ), density bounds, and exact algorithms. Let n and m denote the number of vertices and edges of an input graph, respectively.

TLCM is probably the most studied among the above problems (see, e.g., [18, 28]). It is NP-complete [22], and it remains NP-complete even when the order on one level is prescribed [20]. Kobayashi and Tamaki [31] combined the kernelization result in [30] and an enumeration technique to devise a fixed-parameter tractable (FPT) algorithm with running time in $2^{O(\rho)} + n^{O(1)}$. Since the number of crossings may be quadratic in m, such an FPT result yields an algorithm whose running time is $2^{O(m^2)}$. On the other hand, a trivial $2^{n \log n} O(n^2)$-time exact algorithm for TLCM can be obtained by iteratively considering each of the possible $n! \in \Theta(2^{n \log n})$ vertex orderings, and by verifying whether the considered ordering yields less than ρ crossings (which can be done in $O(n^2)$ time [34]). To the best of our knowledge, however, no faster exact algorithm is known for this problem that performs asymptotically better than the simple one mentioned above. Note that for positive instances of TLCM, m is upper bounded by $\sqrt[3]{\frac{15625 \cdot \rho \cdot n^2}{4608}}$ [4].

TLKP has not been proved to be NP-complete, and no FPT algorithm parameterized by k is known for this problem. A trivial $2^{n \log n} O(n^2)$-time exact algorithm for TLKP can be devised analogously to the one for TLCM. Observe that for positive instances of TLKP and for $k > 5$, m is upper bounded by $\frac{125}{96}\sqrt{k} \cdot n$ [4].

TLQP is NP-complete [2]. If we assume that its natural parameter is the maximum number of allowed mutually crossing edges, then no FPT algorithm exists for it parameterized by this parameter (unless P=NP). A trivial $2^{n \log n} O(n)$-time exact algorithm for TLQP can be obtained

[3] A solution for TLCM (resp. OPCM) consists of a permutation of the vertices of the input graph that yield a 2-level drawing (resp. 1-page book layout) with at most ρ crossings. A solution for TLS (resp. BS) is a subset of the edges of size at most σ whose removal results in a caterpillar (resp. outerplanar graph). A solution for TLQP (resp. TLKP) consists of a permutation of the vertices of the input graph that yields a 2-level quasi-planar drawing (resp. a 2-level k-planar drawing). Finally, a solution for BT consists of a permutation of the vertices and a partition of the edges of the input graph in τ pages that yield a τ-page book embedding.

by iteratively considering each of the possible $n! \in \Theta(2^{n \log n})$ vertex orderings, and by verifying whether the considered ordering yields three pairwise crossings edges. The latter test can be performed in $O(n)$ time, since it reduces to testing planarity of the graph obtained by augmenting the input level graph with a cycle that first traverses all the vertices of the first level and then traverses all the vertices in the second level, according to the considered vertex ordering. Observe that for positive instances of TLQP, m is upper bounded by $2n - 4$ [4].

TLS is NP-complete [40]. Dujmovic et al. gave an FPT algortihm for TLS with $2^{O(\sigma^3)}n$ running time [18]. A trivial $O(m^\sigma n)$-time exact algorithm for TLS performs a guess of σ edges to be removed. This yields $\binom{m}{\sigma} \in O(m^\sigma)$ possible choices. For each of them, a linear-time algorithm to test if the input is a forest of caterpillars (and, thus, it admits a 2-level planar drawing [19]) is invoked. Since caterpillars have at most $n - 1$ edges, we have that for positive instances of TLS, m is upper bounded by $n - 1 + \sigma$.

OPCM is NP-complete [32]. However, the optimum value of crossings can be approximated with an approximation ratio of $O(\log^2 n)$ [38]. Bannister and Eppstein [6, 7] showed that OPCM is fixed-parameter tractable parameterized by ρ. To this aim, they exploit Courcelle's Theorem [15, 16], which provides a super-exponential dependency of the running time in this parameter. A trivial $2^{n \log n} O(n^2)$-time exact algorithm for OPCM can be devised analogous to the one for TLCM. For positive instances of OPCM, m is upper bounded by $\sqrt[3]{37 \cdot \rho \cdot n^2}$ [38].

BT is NP-complete, even when $\tau = 2$ [41], in which case it coincides with the problem of testing whether the input graph is sub-Hamiltonian. This negative result implies that the problem does not admit FPT algorithms parameterized by τ (unless P=NP). A trivial $2^{n \log n + m \log \tau} O(\tau n)$-time exact algorithm for BT can be obtained by iteratively considering each of the possible choices of a permutation for the vertex order and of an assignment of the edges to the τ pages. This yields $n! \cdot \tau^m \in \Theta(2^{n \log n + m \log \tau})$ possible choices. For each of these choices, τ calls to a linear-time algorithm to test whether a graph can be laid out outerplanar with a prescribed vertex order [24, 25], one for each of the graphs induced by the τ pages, are performed to decide whether the considered choice defines a τ-page book embedding. Since outerplanar graphs have at most $2n-3$ edges and by the definition of BT, we have that for positive instances of BT, m is upper bounded by $\tau(2n - 3)$.

BS is NP-complete [42]. Since the treewidth of a YES-instance of BS is at most $2+\sigma$ and since the existence of σ edges whose removal makes the input graph outerplanar can be expressed as an MSO_2 formula whose length depends on σ, Courcelle's Theorem implies that BS is FPT parameterized by σ [15, 16]. A trivial $O(m^\sigma n)$-time exact algorithm for BS can be devised analogously to the one for TLS. By the density of outerplanar graphs and by the definition of BS, we have that for positive instances of BS, m is upper bounded by $2n - 3 + \sigma$.

2 Preliminaries

For basic concepts related to graphs and their drawings, we refer the reader, e.g., to [17, 39]. For the standard notation we adopt to represent quantum gates and circuits, and for basic concepts about quantum computation, we refer the reader, e.g., to [35, 37].

Notation. Let k and h be a positive integers. To ease the description, we will denote the value $\lceil \log_2 k \rceil$ simply as $\log k$, the value $\lceil \frac{k}{h} \rceil$ simply as $\frac{k}{h}$, and the set $\{0, \ldots, k-1\}$ as $[k]$. We refer to any of the permutations of the integers in $[k]$ as a *k-permutation*. A *k-set* is a set of size k.

We denote the set of binary values $\{0, 1\}$ by $\mathbb{B}$. Consider a binary string s of length $a \cdot b$, for some $a, b \in \mathbb{N}$, i.e., $s \in \mathbb{B}^{a \cdot b}$. We often regard s as a sequence of a integers, each represented with

b bits (where the specific a and b will always be clarified in the considered context). For $i \in [a]$, the i-th number in s, which we denote by $s[i]$, is given by the substring of s formed by the bits $s[b \cdot i][b \cdot i + 1] \dots s[b \cdot i + b - 1]$. Moreover, for $j \in [b]$, we denote by $s[i][j]$ the j-th digit of $s[i]$, where $s[i][0]$ is the least significant bit of $s[i]$. We use the binary operator $=$ to compare binary strings; when applied to two binary strings s' and s'' the resulting expression evaluates to 1 if s' and s'' coincide, and to 0 otherwise.

Graph drawings. A *drawing* of a graph maps each vertex to a point in the plane and each edge to a Jordan arc connecting its end-vertices, such that the arc does not contain the image of any other vertex. In this paper, we only consider graph drawings that are *simple*, i.e., every two edges cross at most once and no edge crosses itself. A graph is *planar* if it can be drawn in the plane such that no two edges cross, i.e., it admits a *planar drawing*. A drawing of a graph is *k-planar* (with $k \geq 0$) if it contains no edge crossed more than k times, and it is *quasi-planar* if it contains no three edges that pairwise cross.

Let G be a graph. A *τ-page book layout* of G consists of a linear ordering $\prec$ of the vertices of G along a line, called the *spine*, and of a partition $\{E_1, \dots, E_\tau\}$ of the edges of G into τ sets, called *pages*. A *τ-page book embedding* of G is a τ-page book layout such that no two edges of the same page *cross*. That is, there exist no two edges (u, v) and (w, z) in the same page E_i such that $u \prec v$, $v \prec w$, $u \prec z$, and $z \prec v$. The *book thickness* of G is the minimum integer τ for which G has a τ-page book embedding. The *book skewness* of G is the minimum number of edges that need to be removed from G so that the resulting graph has book thickness 1, that is, it is outerplanar [9].

Let $G = (U, V, E)$ be a bipartite graph, where U and V denote the two subsets of the vertex set of G, and E denotes the edge set of G. A *2-level drawing* of G maps each vertex $u \in U$ to a point on a horizontal line ℓ_u, which we call the *u-layer*, each vertex $v \in V$ to a point on a horizontal line ℓ_v (distinct from ℓ_u), which we call the *v-layer*, and each edge in E to a y-monotone curve between its endpoints. Observe that, from a combinatorial standpoint, a 2-level drawing Γ of G is completely specified by the linear ordering in which the vertices in U and the vertices in V appear along ℓ_u and ℓ_v, respectively. The *2-level skewness* of G is the minimum number of edges that need to be removed from G so that the resulting graph admits a 2-level planar drawing, that is, it is a forest of caterpillars [19].

Next, we provide the definitions of the search problems we study concerning 2-level drawings of graphs.

Two-level Crossing Minimization (TLCM)

Input:	A bipartite graph G and a positive integer ρ.
Output:	A 2-level drawing of G with at most ρ crossings, if one exists.

Two-level k-planarity (TLKP)

Input:	A bipartite graph G and a positive integer k.
Output:	A 2-level k-planar drawing of G, if one exists.

Two-level Quasi Planarity (TLQP)

Input:	A bipartite graph G.
Output:	A 2-level quasi-planar drawing of G, if one exists.

Two-level Skewness (TLS)

Input: A bipartite graph G and an integer σ.
Output: A set $S \subseteq E(G)$ such that $|S| \leq \sigma$ and the graph $G' = (V, E(G) \setminus S)$ is a forest of caterpillars, if one exists.

Finally, we provide the definitions of the search problems we study concerning book embeddings of graphs.

One-Page Crossing Minimization (OPCM)

Input: A graph G and a positive integer ρ.
Output: A 1-page book layout of G with at most ρ crossings, if one exists.

Book Thickness (BT)

Input: A graph G and an integer τ.
Output: A τ-page book embedding of G, if one exists.

Book Skewness (BS)

Input: A graph G and an integer σ.
Output: A set $S \subseteq E(G)$ such that $|S| \leq \sigma$ and the graph $G' = (V, E(G) \setminus S)$ is outerplanar, if one exists.

Partitions. A *k-subset* of a ground set W is a subset of W fo size k. Baranyai's Theorem [8] states the following.

Theorem 1 (Baranyai's Theorem)**.** *Let W be a set and let k be an integer that divides $|W|$. Then, the set of all k-subsets of W may be partitioned into $\binom{|W|-1}{k-1}$ disjoint classes each of size $\frac{|W|}{k}$.*

In order to prove the depth bounds of our circuits we will exploit the following, which can easily be derived from Theorem 1.

Corollary 1.1. *Let X be a set and let k be an integer. Then, the set of all k-subsets of X may be partitioned into at most $\binom{|X|+k-2}{k-1}$ classes each of size at most $\frac{|X|+k-1}{k}$.*

Proof:
We extend X to a set U by adding a dummy elements, where $a = \left((\lfloor \frac{|X|}{k} \rfloor + 1) \cdot k - |X|\right) \mod k$, so that $|U|$ is a multiple of k. By Theorem 1, the set C_U of all k-subsets of U can be partitioned into $\binom{|U|-1}{k-1} = \binom{|X|+a-1}{k-1}$ disjoint classes, each of size $\frac{|U|}{k} = \frac{|X|+a}{k}$. Let C_X be the set of all k-subsets of X. By removing, from the partition of C_U into $\binom{|X|+a-1}{k-1}$ disjoint classes of size $\frac{|X|+a}{k}$, all k-subsets in $C_U \setminus C_X$, we obtain a partition of C_X into at most $\binom{|X|+a-1}{k-1}$ disjoint classes each of size at most $\frac{|X|+a}{k}$. Since $a < k$, such a partition of C_X has at most $\binom{|X|+k-2}{k-1}$ classes, each of size at most $\frac{|X|+k-1}{k}$. □

We remark that, in our subsequent uses of Corollary 1.1 to partition the set of all k-subsets of a ground set X into disjoint classes, we will always have that $k \ll |X|$. Therefore, the number of classes and their size will be in $O(|X|^{k-1})$ and $O(\frac{|X|}{k})$, respectively.

Mathematical formulations. We introduce the mathematical formulations used in the D-Wave[4] quantum annealing platform.

A *constrained binary optimization* (CBO) is a mathematical formulation of an optimization problem, in which the variables are binary. Note that, both the objective function and the constraints may have an arbitrary degree. In some cases, we focus on CBO formulations in which the objective function is not defined, and we aim at verifying whether a problem instance satisfies the given constraints.

A *quadratic unconstrained binary optimization* (QUBO) is a mathematical formulation of an optimization problem, in which the variables are binary, the optimization function is quadratic, and there are no constraints. Specifically, let Q be an upper triangular matrix $Q \in \mathbb{R}^{k\times k}$. Using Q, we can define a quadratic function $f_Q : \mathbb{B}^k \to \mathbb{R}$ that assigns a real value to a k-length binary vector. Namely, we let $f_Q(x) = x^T Q x = \sum_{i=1}^{k} \sum_{j=1}^{k} Q_{ij} x_i x_j$. The QUBO formulation for f_Q asks for a binary vector x^* that minimizes f_Q, i.e., $x^* = \arg\min_{x\in\mathbb{B}^k} f_Q(x)$.

Quantum circuits. Quantum circuits are obtained by composing quantum gates, which receive and output quantum states on the same number of qubits. A quantum gate implements a linear transformation of the input quantum state, and thus a state that is a superposition of other states is mapped to the superposition of the images of such states. In particular, any such a transformation U must be *unitary*, that is, a linear transformation such that $I = U^\dagger U = U U^\dagger$. Any quantum computation is *reversible* in the following sense. As long as no measurement is performed on the output quantum state $|\phi\rangle$ obtained by applying U to an initial quantum state $|\psi\rangle$, such an initial quantum state can be recovered by applying $U^{-1} = U^\dagger$ to $|\phi\rangle$.

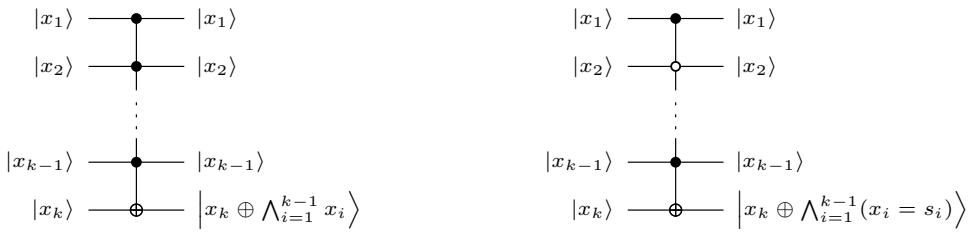
In Dirac's notation, a *ket* such as $|v\rangle$, where v is an arbitrary label, refers to a vector in the complex Hilbert space representing a state of a quantum system. We denote by $|0_k\rangle$ the quantum basis state composed of k qubits set to $|0\rangle$.

A reversible version of any classical circuit can be obtained by composing reversible gates, known as *Toffoli gates*. The Toffoli gate has k input and output qubits; refer to Fig. 2a. The first $k-1$ qubits $|x_1, \ldots, x_{k-1}\rangle$ are *control qubits*, whereas the last qubit is the *target qubit* $|x_k\rangle$. When provided with the input superposition $|x_1, \ldots, x_k\rangle$, the Toffoli gate produces the output superposition $\left|x_1, \ldots, x_{k-1}, x_k \oplus \bigwedge_{i=1}^{k-1} x_i\right\rangle$. We will exploit a more general version of the Toffoli gate, which is defined with respect to a binary length-$k-1$ string s; Fig. 2b. Such a Toffoli gate, when provided with the input superposition $|x_1, \ldots, x_k\rangle$, produces the output superposition $\left|x_1, \ldots, x_{k-1}, x_k \oplus \bigwedge_{i=1}^{k-1}(x_i = s_i)\right\rangle$. Observe that a standard Toffoli gate corresponds to the general version we adopt, defined with respect to the binary string s consisting of $k-1$ bits with value 1. Following a common standard, we mark with a black-filled (resp. white-filled) dot the control qubit corresponding to the bits of s with value 1 (resp. 0). In the remainder, we will refer to the general version of this gate as a Toffoli gate.

We make extensive use of the Hadamard gate H, whose unitary transformation matrix is $\frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$. In particular, we will exploit the Hadamard gate to create an equal superposition of the states $|0\rangle$ and $|1\rangle$, when provided with the input state $|0\rangle$.

Let D be a vertex-weighted directed acyclic graph (DAG). The *depth* of D is the number of vertices in a longest path of D. Two vertices u and v of D are *incomparable* if there exists no directed path from u to v, or vice versa. An *anti-chain* of D is a maximal set of incomparable vertices. The *weight* of an anti-chain is the sum of its vertex weights. The *width* of D is the

[4]D-Wave quantum annealing platform system documentation (https://docs.dwavesys.com/docs/latest/index.html)



(a) Standard Toffoli gate. (b) General Toffoli gate.

Figure 2: The circuit representation of a Toffoli gate.

maximum weight of an anti-chain of D. A quantum circuit Q can be modeled as a vertex-weighted directed acyclic graph D_Q, whose vertices correspond to the gates of Q and whose directed edges represent qubit input-output dependencies. Moreover, the weight of a vertex representing a gate U corresponds to the number of elementary gates [35, 37] needed to build U.

Note that the size of a circuit corresponds to the total number of operations that must be performed to execute the circuit, the depth of a circuit corresponds to the number of distinct time steps at which gates are applied, and the width of a circuit corresponds to the maximum number of operations that can be performed "in parallel". Therefore, it is natural to upper bound the time complexity of executing the circuit either by its depth, assuming the gates at each layer can be executed in parallel, or by its circuit complexity, assuming the gates are executed sequentially.

Next, we provide lemmas that describe the input-output behaviour of some gates that will be used in the following sections and present their circuit complexity, depth and width. These gates can be obtained by combining elementary quantum circuits such as Toffoli, Half-Adder, and Full-Adder Gates. In [5, 10, 29], implementations of the simple quantum circuits we will use, or slightly-similar ones, can be found. Let $\phi[i]$ and $\phi[j]$ be binary strings of length $\log t$, which we interpret as binary integers represented with $\log t$ bits. Also, let $|\phi[i]\rangle$ and $|\phi[j]\rangle$ be the basis states corresponding to $\phi[i]$ and $\phi[j]$, respectively.

First, we focus on gate $U_=$ that, given integers $\phi[i]$ and $\phi[j]$, verifies if $\phi[i]$ is equal to $\phi[j]$.

Lemma 1 ([5])**.** *There exists a gate $U_=$ that, when provided with the input superposition $|\phi[i]\rangle\,|\phi[j]\rangle\,|0_{\log t}\rangle\,|0\rangle$, produces the output superposition $|\phi[i]\rangle\,|\phi[j]\rangle\,|0_{\log t}\rangle\,|\phi[i] = \phi[j]\rangle$. Gate $U_=$ has $O(\log t)$ circuit complexity, depth, and width.*

Second, we focus on gate $U_<$ that, given integers $\phi[i]$ and $\phi[j]$ represented with $\log t$ bits, verifies if $\phi[i]$ is less than $\phi[j]$.

Lemma 2 ([5])**.** *There exists a gate $U_<$ that, when provided with the input superposition $|\phi[i]\rangle\,|\phi[j]\rangle\,|0_{\log t}\rangle\,|0\rangle$, produces the output superposition $|\phi[i]\rangle\,|\phi[j]\rangle\,|0_{\log t}\rangle\,|\phi[i] < \phi[j]\rangle$. Gate $U_<$ has $O(\log t)$ circuit complexity, depth, and width.*

Third, we focus on gate U_{1s} that, given a binary string b of length t, counts how many bits set to 1 are in it, that is, the Hamming weight of b. For simplicity, we assume that t is a power of 2. If not, we can always append to b the smallest number of 0s such that this property holds. Observe that, in the worst case, the length of the string may double, which does not alter the asymptotic bounds of the next lemma.

Lemma 3 ([5, 10, 29])**.** *There exists a gate U_{1s} that, when provided with the input superposition $|b_0 \dots b_{t-1}\rangle\,|0_h\rangle\,|0_k\rangle$, where t is a power of 2, $k = 1 + \log t$, and $h = 4t - 2\log t - 4$, produces the*

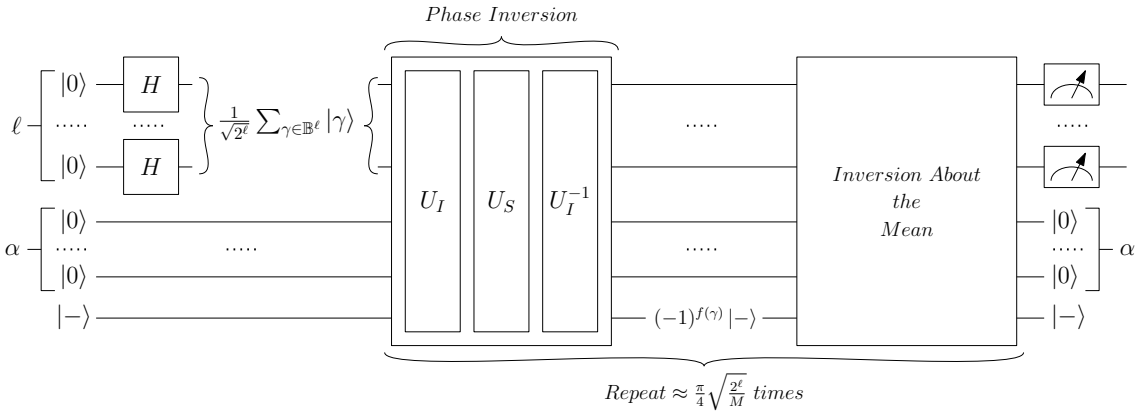


Figure 3: Overview of the quantum graph drawing framework based on Grover's approach.

output superposition $|b_0 \dots b_{t-1}\rangle |0_h\rangle |s\rangle$*, where* s *is the binary representation, in* $\log t + 1$ *bits, of the total number* $\sum_{i=0}^{t-1} b_i$ *of qubits set to* 1 *in* b*. Gate* U_{1s} *has* $O(t)$ *circuit complexity,* $O(\log^2 t)$ *depth, and* $O(t)$ *width.*

3 A Quantum Framework for Graph Drawing Problems

In this section, we establish a framework for dealing with several NP-complete graph drawing search problems based on linear orderings; refer to Fig. 3. A *search problem* P is defined by a binary relation $R \subseteq I \times S$. The elements of I are the *instances* of P, whereas the elements of S are the *solutions* of P. For every instance $\Lambda \in I$, the set $Sol_P(\Lambda) = \{s \in S : (\Lambda, s) \in R\}$ is the set of *solutions* of P for Λ. The number of solution of P for Λ is $|Sol_P(\Lambda)|$. The framework is based on the Grover's approach for quantum search [23], which builds upon three circuits. The first circuit consist of multiple Hadamard gates that, when provided with a collection of qubits set to $|0\rangle$, builds a uniform superposition of all basis states of a collection of qubits. Such a superposition includes basis states that correspond to encodings of all the candidate solutions to the problem, as well as basis states that may not correspond to well-formed encodings (according to the selected encoding scheme) of candidate solutions to the problem. The second circuit exploits an *oracle* to perform the so-called Phase Inversion. The third circuit executes the so-called Inversion about the Mean. The second and the third circuit are executed a number of times which guarantees that a final measure outputs a solution, if any exists, with high probability.

Theorem 2 (Grover's search [1, 23])**.** *Let* P *be a search problem whose solutions can be encoded using* ℓ *bits and suppose that there exists a* Phase Inversion *circuit for* P *with* $c(\ell)$ *circuit complexity and* $d(\ell)$ *depth. Assume that* $c(\ell)$ *and* $d(\ell)$ *are* $\Omega(\log \ell)$*. Then, there exists a quantum circuit that, given an instance* Λ *of* P*, outputs a solution of* P *for* Λ*, if any exists, with* $\frac{\pi}{4}\sqrt{\frac{2^\ell}{M}} \cdot O(c(\ell))$ *circuit complexity and* $\frac{\pi}{4}\sqrt{\frac{2^\ell}{M}} \cdot O(d(\ell))$ *depth, where* M *is the number of solutions of* P *for* Λ*.*

Let n and m be the number of vertices and edges of an input graph G, respectively. Observe that, in all the problems we consider, G admits the sought layout if and only if each of its connected components does. Hence, in the following, we assume that G is connected, and therefore

$m \geq n-1$. During the computation, we will manage a superposition $|\Gamma\rangle = |\Phi\rangle\,|\Psi\rangle\,|\Theta\rangle$, where $|\Phi\rangle$ is a superposition of $n \log n$ qubits, $|\Psi\rangle$ is a superposition of $m \log \tau$ qubits, and $|\Theta\rangle$ is a superposition of $\sigma \log m$ qubits. In particular, for some of the problems, $|\Psi\rangle$ and/or $|\Theta\rangle$ might not be present. We denote by ℓ the value $(n \log n) + (m \log \tau) + (\sigma \log m)$, where the second and/or third terms might be missing.

The superposition $|\Phi\rangle = \sum_{\phi \in \mathbb{B}^{n \log n}} c_\phi\,|\phi\rangle$ represents the superposition of all sequences of n natural numbers with values in $[n]$, each represented by a binary string ϕ of length $n \log n$ (according to notation defined in Sect. 2 to represent sequences of integers). The digits corresponding to each natural number contained in ϕ form a consecutive sequence of length $\log n$. In particular, we denote by $\phi[i]$ the i-th natural number contained in ϕ. The purpose of $|\phi\rangle$ is to represent the position of each vertex of G in a total order. To this aim, observe that, within the superposition $|\Phi\rangle$, all possible states corresponding to assignments of positions from 0 to $n-1$ for each vertex in G are included.

The superposition $|\Psi\rangle = \sum_{\psi \in \mathbb{B}^{m \log \tau}} c_\psi\,|\psi\rangle$ represents the superposition of all sequences of m natural numbers with values in $[\tau]$, each represented by a binary string ψ of length $m \log \tau$. The digits corresponding to each natural number contained in ψ form a consecutive sequence of length $\log \tau$. In particular, we denote by $\psi[i]$ the i-th natural number contained in ψ. The purpose of $|\psi\rangle$ is to represent a coloring of the edges of G with color in $[\tau]$. To this aim, observe that, within the superposition $|\Psi\rangle$, all possible states corresponding to an assignment of integers from 0 to $\tau-1$ for each edge in G are included.

The superposition $|\Theta\rangle = \sum_{\theta \in \mathbb{B}^{\sigma \log m}} c_\theta\,|\theta\rangle$ represents the superposition of all sequences of σ natural numbers with values in $[m]$, each represented by a binary string θ of length $\sigma \log m$. The digits corresponding to each natural number contained in θ form a consecutive sequence of length $\log m$. In particular, we denote by $\theta[i]$ the i-th natural number contained in θ. The purpose of $|\theta\rangle$ is to represent a subset of the edges of G of size at most σ, each labeled with an integer in $[m]$. To this aim, observe that, within the superposition $|\Theta\rangle$, all possible states corresponding to a selection of σ edges of G, where each edge is indexed with an integer from 0 to $m-1$, are included.

For problems TLCM, TLKP, TLQP, and OPCM, we have that $|\Gamma\rangle = |\Phi\rangle$. For problem BT, we have that $|\Gamma\rangle = |\Phi\rangle\,|\Psi\rangle$. Finally, for problems TLS and BS, we have that $|\Gamma\rangle = |\Phi\rangle\,|\Theta\rangle$.

Next, we present an overview of how the superposition $|\Gamma\rangle$ evolves within the three main circuits of the framework; refer to Fig. 3.

First, in all problems we study, ℓ qubits set to $|0\rangle$ enter a Hadamard gate that outputs the uniform superposition $|\Gamma\rangle = H^{\otimes \ell}\,|0_\ell\rangle = \frac{1}{\sqrt{2^\ell}} \sum_{\gamma \in \mathbb{B}^\ell} |\gamma\rangle$. Observe that, such a superposition, corresponds to the tensor product of the uniform superpositions $|\Phi\rangle = H^{\otimes n \log n}\,|0_{n \log n}\rangle$, $|\Psi\rangle = H^{\otimes m \log \tau}\,|0_{m \log \tau}\rangle$, and $|\Theta\rangle = H^{\otimes \sigma \log m}\,|0_{\sigma \log m}\rangle$, where possibly Ψ and/or Θ might be not present. Also, observe that, within $|\Gamma\rangle$, all possible encodings of solutions of the considered problems are included, if any exist.

Second, we focus on the Phase Inversion circuit. In the first iteration, it receives as input **(i)** the uniform superposition $|\Gamma\rangle = H^{\otimes \ell}\,|0_\ell\rangle$, **(ii)** α ancilla qubits set to $|0\rangle$, whose number depends on the type of problem we are addressing, and **(iii)** a qubit set to $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Namely, it receives as input the superposition $|\Gamma\rangle\,|0_\alpha\rangle\,|-\rangle$, where $|\Gamma\rangle = \frac{1}{\sqrt{2^\ell}} \sum_{\gamma \in \mathbb{B}^\ell} |\gamma\rangle$. It outputs the superposition $\frac{1}{\sqrt{2^\ell}} \sum_{\gamma \in \mathbb{B}^\ell} (-1)^{f(\gamma)}\,|\gamma\rangle\,|0_\alpha\rangle\,|-\rangle$, where $f(\gamma) = 1$ if and only if γ represents a solution to the considered problem. In general, the Phase Inversion circuit receives in input the superposition $|\Gamma\rangle\,|0_\alpha\rangle\,|-\rangle$, where $|\Gamma\rangle = \sum_{\gamma \in \mathbb{B}^\ell} c_\gamma\,|\gamma\rangle$. It outputs the superposition $\sum_{\gamma \in \mathbb{B}^\ell} (-1)^{f(\gamma)} c_\gamma\,|\gamma\rangle\,|0_\alpha\rangle\,|-\rangle$. To implement this phase inversion transformation, we exploit the following property of the $|-\rangle$ state. Applying the NOT gate (also called the X gate) to $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ yields $X\,|-\rangle = \frac{1}{\sqrt{2}}(-\,|0\rangle + |1\rangle) = -\,|-\rangle$. This means that flipping the $|-\rangle$ state introduces a *global* phase of -1 to the entire state.

We use this property to implement the phase inversion in the Grover framework. Mathematically, the transformation can be written as $\sum_{\gamma\in\mathbb{B}^\ell} c_\gamma |\gamma\rangle |0_\alpha\rangle (-1)^{f(\gamma)} |-\rangle = \sum_{\gamma\in\mathbb{B}^\ell} (-1)^{f(\gamma)} c_\gamma |\gamma\rangle |0_\alpha\rangle |-\rangle$.

We remark that the values of the complex coefficients c_γ depend on the iteration. Third, in Grover's approach, the INVERSION ABOUT THE MEAN circuit does not depend on the specific problem, but it is standard gate common to all applications of the Grover's framework. The aim of this circuit is to adjust the amplitudes of the quantum states in the superposition by reflecting them around their average value. This reflection increases the amplitudes of states with values below the mean while decreasing those above it, thereby redistributing amplitude to enhance the probability of measuring the solution state, whose amplitudes have been placed below the mean by the phase inversion circuit. Therefore, the main challenge in exploiting Grover's framework is to provide a specific phase inversion circuit that can *efficiently* evaluate the function $f(\gamma)$.

For each problem we consider, we provide a specific PHASE INVERSION circuit. All such circuits consist of three circuits (see Fig. 3), the first is called INPUT TRANSDUCER and is denoted by U_I, the second is called SOLUTION DETECTOR and is denoted by U_S, and the third is the inverse U_I^{-1} of the INPUT TRANSDUCER. The purpose of the INPUT TRANSDUCER circuits is to "filter out" the states of $|\Gamma\rangle$ that do not correspond to well-formed encodings of candidate solutions. The purpose of the SOLUTION DETECTOR circuits is to invert the amplitude of the states of $|\Gamma\rangle$ that correspond to encodings of solutions, if any. The purpose of U_I^{-1} is to restore the state of the ancilla qubits to $|0\rangle$ so that they may be employed in the subsequent iterations of the amplitude-amplification process.

The INPUT TRANSDUCER circuits are described in Sect. 4. The SOLUTION DETECTOR circuits are described in Sect. 5. In that section, we also combine the INPUT TRANSDUCER circuits and the SOLUTION DETECTOR circuits to prove the following lemma; refer also to Table 1.

Lemma 4. *The TLCM, TLKP, TLQP, TLS, OPCM, BT, and BS problems admit* PHASE INVERSION *circuits whose circuit complexity, depth, and width are bounded as follows:*

TLCM *Circuit complexity:* $O(m^2)$. *Depth:* $O(n^2)$. *Width* $O(m^2)$.

TLKP *Circuit complexity:* $O(m^2)$. *Depth:* $O(m \log^2 m)$. *Width* $O(m)$.

TLQP *Circuit complexity:* $O(m^6)$. *Depth:* $O(m^4)$. *Width* $O(m^2)$.

TLS *Circuit complexity:* $O(m^2)$. *Depth:* $O(m)$. *Width* $O(m)$.

OPCM *Circuit complexity:* $O(n^8)$. *Depth:* $O(n^6)$. *Width* $O(m^2)$.

BT *Circuit complexity:* $O(n^8)$. *Depth:* $O(n^6)$. *Width* $O(m)$.

BS *Circuit complexity:* $O(n^8)$. *Depth:* $O(n^6)$. *Width* $O(m)$.

Theorem 2 and Lemma 4 imply the following.

Theorem 3. *In the quantum circuit model of computation, the TLCM, TLKP, TLQP, TLS, OPCM, BT, and BS problems can be solved with the following sequential and parallel time bounds (where M denotes the number of solutions to the problem):*

TLCM *Sequential:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(m^2)$. *Parallel:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(n^2)$.

TLKP *Sequential:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(m^2)$. *Parallel:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(m \log^2 m)$.

TLQP *Sequential:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(m^6)$. *Parallel:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(m^4)$.

TLS *Sequential:* $\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}} \cdot O(m^2)$. *Parallel:* $\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}} \cdot O(m)$.

OPCM *Sequential:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(n^8)$. *Parallel:* $\sqrt{\frac{2^{n \log n}}{M}} \cdot O(n^6)$.

BT *Sequential:* $\sqrt{\frac{2^{n \log n + m \log \tau}}{M}} \cdot O(n^8)$. *Parallel:* $\sqrt{\frac{2^{n \log n + m \log \tau}}{M}} \cdot O(n^6)$.

BS *Sequential:* $\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}} \cdot O(n^8)$. *Parallel:* $\sqrt{\frac{2^{n \log n + \sigma \log m}}{M}} \cdot O(n^6)$.

4 Input Transducer Circuits

We use two different versions of circuit U_I, depending on the considered problem. Namely, for all problems but TLS and BS, circuit U_I consists of just one circuit U_{OT}, called Order-Transducer (refer to Fig. 4). For problems TLS and BS, circuit U_I executes, in parallel to U_{OT}, another circuit U_{ST}, called Skewness-Transducer (refer to Fig. 7).

4.1 Order Transducer

Let ϕ be a binary string of length $n \log n$, which we interpret as a sequence of n binary integers, each consisting of $\log n$ bits. Recall that we denote by $\phi[i]$ the i-th binary integer contained in ϕ. Also, let $|\phi\rangle$ be the basis state corresponding to ϕ. This subsection is devoted to proving the following lemma.

Lemma 5. *There exists a gate U_{OT} that, when provided with the input superposition $|\phi\rangle |0_\alpha\rangle$, where $\alpha = \frac{n}{2}(n - 1 + \log n) + 1$, produces the output superposition*

$$|\phi\rangle |f(\phi)\rangle |x_{0,1}\rangle \dots |x_{0,n-1}\rangle \dots |x_{i,j}\rangle \dots |x_{n-2,n-1}\rangle \left|0_{\frac{n}{2} \log n}\right\rangle,$$

such that $|x_{i,j}\rangle = 1$ if and only if $\phi[i] < \phi[j]$ and $f(\phi) = 1$ if and only if ϕ represents an n-permutation. U_{OT} has $O(n^2 \log n)$ circuit complexity, and $O(n \log n)$ depth and width.

Proof of Lemma 5. The input of U_{OT} is composed of $n \log n + \alpha$ qubits, the first $\ell = n \log n$ form a superposition $|\phi\rangle$ and the rest are set to $|0\rangle$. First, $|\phi\rangle$ and $\frac{n}{2}(\log n + 3) + 1$ qubits set to $|0\rangle$ enter a gate U_C, called Collision Detector. The purpose of U_C is to compute the superposition $|\phi\rangle |f(\phi)\rangle \left|0_{\frac{n}{2}(\log n + 3)}\right\rangle$, where $f(\phi) = 1$ if and only if $\phi[i] \neq \phi[j]$ for each $i \neq j$. It has $O(n^2 \log n)$ circuit complexity, and it has $O(n \log n)$ depth and width. Second, $|\phi\rangle$ and $\frac{n}{2} \log n + \frac{n(n-1)}{2}$ qubits set to $|0\rangle$ enter a gate U_P called Precedence Constructor. The purpose of U_P is to compute a superposition $|\phi\rangle |x\rangle \left|0_{\frac{n}{2} \log n}\right\rangle$, where $|x\rangle = |x_{0,1}\rangle \dots |x_{0,n-1}\rangle \dots |x_{i,j}\rangle \dots |x_{n-2,n-1}\rangle$ and $x_{i,j} = 1$ if and only if $\phi[i] < \phi[j]$. It has circuit complexity $O(n^2 \log n)$, and it has $O(n \log n)$ depth and width. Gate U_{OT} has $O(n^2 \log n)$ circuit complexity, and it has $O(n \log n)$ depth and width.

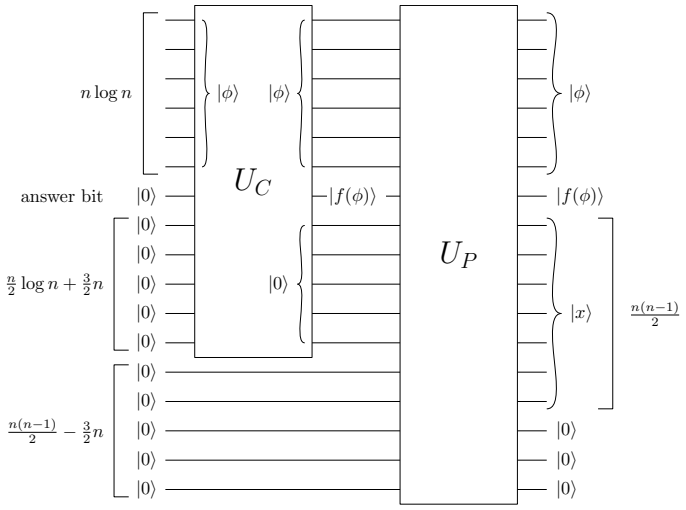


Figure 4: The Order Transducer gate U_{OT}.

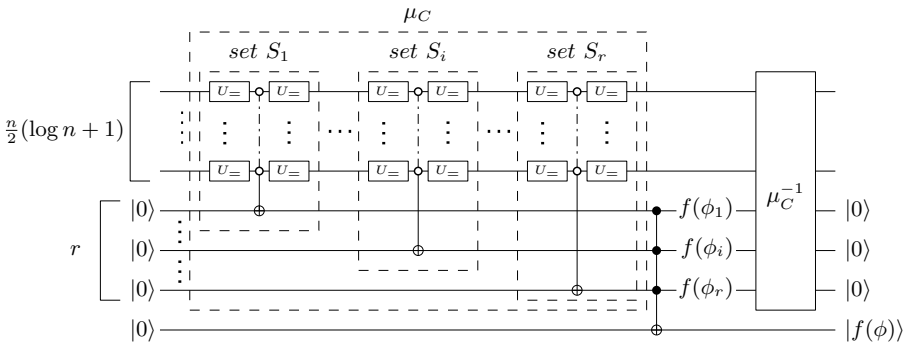


Figure 5: The gate U_C.

Collision-detector. Gate U_C works as follows. Refer to Fig. 5.

It executes two gates μ_C and μ_C^{-1} and in between such gates it executes a Toffoli gate. The input of μ_C is the superposition $|\phi[0]\rangle \dots |\phi[n-1]\rangle \left|0_{\frac{n}{2}(\log n+1)}\right\rangle |0_n\rangle$. The output of μ_C is the superposition $|\phi[0]\rangle \dots |\phi[n-1]\rangle \left|0_{\frac{n}{2}(\log n+1)}\right\rangle |f(\phi_0)\rangle \dots |f(\phi_i)\rangle \dots |f(\phi_{n-1})\rangle$. In gate μ_C, we compare the unordered pairs of numbers $\phi[i]$ and $\phi[j]$ in parallel as follows. Consider that if two numbers are compared, none of the two can be compared with another number at the same time. Hence, we partition the pairs using Corollary 1.1 (with $k = 2$ and $|X| = n$) into $r = O(n)$ classes $S_1, \dots, S_r$ each containing at most $\frac{n}{2}$ disjoint pairs.

For each pair (i, j) of S_1 (refer to Fig. 5) we use a $U_=$ gate to compare $\phi[i]$ and $\phi[j]$. Recall that the gate $U_=$ outputs a superposition $|\phi[i]\rangle |\phi[j]\rangle |0\rangle \dots |0\rangle |\phi[i] == \phi[j]\rangle$. All the last output qubits of the $U_=$ gates for S_1 enter a Toffoli gate with $|S_i| + 1$ inputs and outputs, which computes the qubit $|f(\phi_0)\rangle$ such that $\phi_0 = 1$ if and only if all of the first $|S_i|$ input qubits are equal to $|0\rangle$, i.e., all pairs correspond to different numbers.

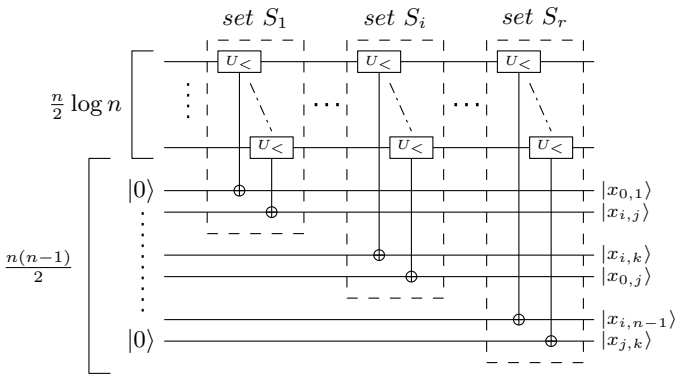


Figure 6: The gate U_P.

After dealing with S_1, we deal with S_2 with the same technique and keep on dealing with the S_i sets until S_r is reached.

All the last output qubits of the $U_=$ gates for S_i enter a Toffoli gate that outputs a qubit $|f(\phi_i)\rangle$ such that $f(\phi_1) = 1$ if and only if all of them are equal to $|0\rangle$, i.e., if does not exist pair (j, k) of S_i where $\phi[j] = \phi[k]$. In order to allow the reuse of the ancilla qubits, except for the qubit $|f(\phi_i)\rangle$, gate U_C executes in parallel a gate $U_=^{-1}$ for each pair in S_i.

All the qubits $|f(\phi_i)\rangle$ and the qubit $|\phi\rangle$ enter a Toffoli gate with $r + 1$ inputs and outputs. The first r qubits are control qubits, the target qubit is $|f(\phi)\rangle$, which is initialized to $|0\rangle$. The target qubit is set to $|\bigwedge_{i=1}^{r} \phi_i\rangle$. In order to allow the reuse of the ancilla qubits, we apply to the entire circuit preceding the Toffoli gate its inverse gate. Recall that by Lemma 1, gate $U_=$ has $O(\log n)$ circuit complexity, depth, and width. Therefore, gate U_C has $O(n^2 \log n)$ circuit complexity, and it has $O(n \log n)$ depth and width.

Precedence Constructor. Gate U_P works as follows. Refer to Fig. 6.

For each pair i, j ($i, j \in [n]$ and $i \neq j$) U_P exploits $U_<$ that outputs a qubit $|x_{i,j}\rangle$ such that $x_{i,j} = \phi[i] < \phi[j]$. Using several gates $U_<$, we compare the ordered pairs of numbers $\phi[i]$ and $\phi[j]$ in parallel as follows. As for gate U_C, if two numbers are compared, none of the two can be compared with another number at the same time. Hence, we partition the pairs using Corollary 1.1 (with $k = 2$ and $|X| = n$) into $r \in O(n)$ classes $S_1, \ldots, S_r$ each containing at most $\frac{n}{2}$ disjoint pairs.

For each pair (i, j) of S_1 (refer to Fig. 6) we use a $U_<$ gate to compare $\phi[i]$ and $\phi[j]$. Recall that the gate $U_<$ outputs a superposition $|\phi[i]\rangle\,|\phi[j]\rangle\,|0\rangle \ldots |0\rangle\,|\phi[i] < \phi[j]\rangle$. In order to allow the reuse of the ancilla qubit different from $|x_{i,j}\rangle$ for each pair $(i, j) \in S_1$, we use a symmetric circuit that transforms the ancilla output qubits into a sequence of $|0\rangle$ qubits, that will be re-used in the following step.

After dealing with S_1, we deal with S_2 with the same technique and keep on dealing with the S_i sets until S_r is reached.

Recall that by Lemma 2, gate $U_<$ has $O(\log n)$ circuit complexity, depth, and width. Therefore, gate U_P has $O(n^2 \log n)$ circuit complexity, and it has $O(n \log n)$ depth and width. □

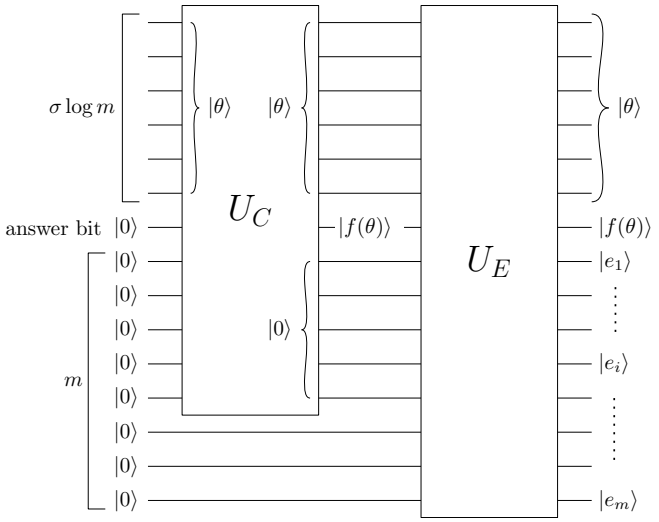


Figure 7: The Skewness Transducer gate U_{ST}.

4.2 Skewness Transducer

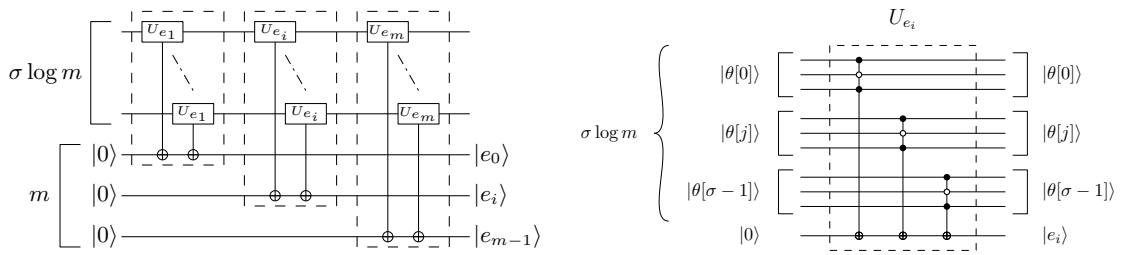
Let θ be a binary string of length $\sigma \log m$, which we interpret as a sequence of σ binary integers, each consisting of $\log m$ bits. Recall that we denote by $\theta[i]$ the i-th binary integer contained in θ. Also, let $|\theta\rangle$ be the basis state corresponding to θ. This subsection is devoted to proving the following lemma.

Lemma 6. *There exists a gate U_{ST} that, when provided with the input superposition $|\theta\rangle |0\rangle |0_m\rangle$, produces the output superposition $|\theta\rangle |f(\theta)\rangle |e_0\rangle \dots |e_i\rangle \dots |e_{m-1}\rangle$, such that $f(\theta) = 1$ if and only if θ represents a subset of size σ of the set $[m]$, and when $f(\theta) = 1$ it holds that $e_i = 1$ if and only if there exists $j \in [\sigma]$ such that $\theta[j]$ coincides with (the binary representation of) the integer i. Gate U_{ST} has $O(\sigma m \log m)$ circuit complexity and depth, and $O(\sigma \log m)$ width.*

Proof of Lemma 6. The input of U_{ST} is composed of $\sigma \log m + 1 + m$ qubits, the first $\ell = \sigma \log m$ qubits form a superposition θ and the rest are set to $|0\rangle$. First, $|\theta\rangle$, $|0\rangle$, and $h = \frac{\sigma}{2}(\log m + 3)$ qubits set to $|0\rangle$ enter an instance of the Collision Detector gate U_C used in the proof of Lemma 5, where the qubits of the superposition $|\theta\rangle |0\rangle |0_h\rangle$ play the role of the qubits of the superposition $|\phi\rangle |0\rangle \left|0_{\frac{n}{2}(\log n+3)}\right\rangle$. The purpose of this instance of U_C is to compute the superposition $|\theta\rangle |f(\theta)\rangle |0_h\rangle$, where $f(\theta) = 1$ if and only if $\theta[a] \neq \theta[b]$ for each $0 \leq a < b \leq \sigma - 1$. It has $O(\sigma^2 \log m)$ circuit complexity and it has $O(\sigma \log(m))$ depth and width. Second, the superpositions $|\theta\rangle$ and $|0_m\rangle$ enter a gate U_E, called Edge Constructor; refer to Fig. 8a. The purpose of U_E is to compute the superposition $|\theta\rangle |e\rangle$, where $|e\rangle = |e_0\rangle \dots |e_i\rangle \dots |e_{m-1}\rangle$ and, when $f(\theta) = 1$, it holds that $e_i = 1$ if and only if there exists a $j \in [\sigma]$ such that $\theta[j]$ coincides with (the binary representation of) the integer i. It has $O(\sigma m \log m)$ circuit complexity, $O(\sigma m)$ depth, and $O(\log m)$ width. Recall that $\sigma < m$. Thus, we have that gate U_{ST} has $O(\sigma m \log m)$ circuit complexity, $O(\sigma m)$ depth, and $O(\sigma \log m)$ width.

Edge Constructor. The gate U_E exploits instances of the auxiliary gate U_{e_i}, defined for each edge $e_i \in E$ as follows. Refer to Fig. 8b. When provided with the input superposition $|\theta\rangle\,|0\rangle$, the gate U_{e_i} produces the output superposition $|\theta\rangle\,|e_i\rangle$, where – provided that θ represents a subset of size σ of the set $[m]$, i.e., $f(\theta) = 1$ – it holds that $e_i = 1$ if and only if there exists $j \in [\sigma]$ such that $\theta[j]$ coincides with the integer i. The gate U_{e_i} contains σ Toffoli gates, each with $\log(m) + 1$ inputs and outputs. All such Toffoli gates share the same target qubit. The control qubits of the first Toffoli gate T_0 are $|\theta[0][0]\rangle \dots |\theta[0][\log(m) - 1]\rangle$ and its target qubit is initialized to $|0\rangle$. It turns the target qubit into $|1\rangle$ if and only if $\theta[0][0] \dots \theta[0][\log(m) - 1]$ coincides with the integer i. For $j = 1, \dots, \sigma - 1$, gate U_{e_i} contains a Toffoli gate T_j that takes in input the superposition $|\theta[j][0]\rangle \dots |\theta[j][\log(m) - 1]\rangle$ and the target qubit that has been output by T_{j-1}. It flips the value of the target qubit if and only if $\theta[j][0] \dots \theta[j][\log(m) - 1]$ coincides with (the binary representation of) the integer i. Thus, if at most one of the j integers composing θ coincides with the integer i, then $e_i = 1$ if and only if there exists $j \in [\sigma]$ such that $\theta[j]$ coincides with (the binary representation of) the integer i. Gate U_{e_i} has $O(\sigma \log m)$ circuit complexity, $O(\sigma)$ depth, and $O(\log m)$ width.

The gate U_E computes $|e\rangle = |e_0\rangle \dots |e_i\rangle \dots |e_{m-1}\rangle$ as follows. For $i \in [m]$, it computes $|e_i\rangle$ by applying the gate U_{e_i} to the superposition $|\theta\rangle\,|0\rangle$, where the last qubit is the i-th qubit of the m qubits compositing the superposition $|0_m\rangle$, which is provided in input to the gate U_E. Gate U_E has circuit complexity $O(\sigma m \log m)$, depth $O(\sigma m)$, and width $O(\log m)$. □



(a) Gate U_E (b) Gate U_{e_i}

Figure 8: Illustration for the construction of gate U_E.

5 Solution Detector Circuits

In this section, we present the details of the Solution Detector circuits U_S for the problems we consider.

Recall that the Order Transducer circuit produces the output superposition

$$|\phi\rangle\,|f(\phi)\rangle\,|x_{0,1}\rangle \dots |x_{0,n-1}\rangle \dots |x_{i,j}\rangle \dots |x_{n-2,n-1}\rangle\,\left|0_{\frac{n}{2}\log n}\right\rangle,$$

such that $x_{i,j} = 1$ if and only if $\phi[i] < \phi[j]$, and $f(\phi) = 1$ if and only if ϕ represents an n-permutation. In the following, for simplicity, we denote the superposition $|x_{0,1}\rangle \dots |x_{0,n-1}\rangle \dots |x_{i,j}\rangle \dots |x_{n-2,n-1}\rangle$ as $|x\rangle$. We interpret the values $x_{0,1}, \dots, x_{0,n-1}, \dots, x_{i,j}, \dots, x_{n-2,n-1}$ as the entries above the main diagonal of a square binary matrix X, where $X[i][j] = x_{i,j}$, and whose entries along and below the main diagonal are undefined. We use such entries to represent the precedence between vertices in a graph drawing. Let x be the string obtained by concatenating $x_{0,1}, \dots, x_{0,n-1}, \dots, x_{i,j}, \dots, x_{n-2,n-1}$. We will use x both for book layouts and for 2-level drawings as follows.

Consider book layouts of a graph $G = (V, E)$. We denote by $\Pi(x)$ the vertex order along the spine of a book layout of G defined as follows. We have that, if $x_{i,j} = 1$, then vertex v_i precedes vertex v_j in $\Pi(x)$. Conversely, if $x_{i,j} = 0$, then vertex v_j precedes vertex v_i in $\Pi(x)$. Consider now a 2-level drawing of a graph $G = (U, V, E)$. We assume that the vertices in U are labeled as $u_0, \ldots, u_{|U|-1}$, and the vertices of V are labeled as $v_{|U|}, \ldots, v_{|U|+|V|-1}$. We denote by $D(x)$ the 2-level drawing of G defined as follows. Let w_i and w_j be two vertices of G. Suppose that $w_i, w_j \in U$. If $x_{i,j} = 1$, then $w_i \prec w_j$ along the horizontal line ℓ_u, otherwise, $w_j \prec w_i$ along ℓ_u. Suppose that $w_i, w_j \in V$. If $x_{i,j} = 1$, then $w_i \prec w_j$ along the horizontal line ℓ_v, otherwise, $w_j \prec w_i$ along ℓ_v. Suppose now that $w_i \in U$ and $w_j \in V$. Then, we assume that $x_{i,j} = 0$, which we interpret as the absence of a precedence relation between such vertices.

We remark that, if ϕ is not an n-permutation, then $\Pi(x)$ and $D(x)$ do not correspond to actual spine orders and 2-level drawings, respectively. In this case, we say that they are *degenerate*.

We will exploit $|x\rangle$ to compute a superposition $|\chi_{0,1}\rangle \ldots |\chi_{0,m-1}\rangle \ldots |\chi_{i,j}\rangle \ldots |\chi_{m-2,m-1}\rangle$, which we will denote for simplicity by $|\chi\rangle$. We interpret the values $\chi_{0,1}, \ldots, \chi_{0,m-1}, \ldots, \chi_{i,j}, \ldots, \chi_{m-2,m-1}$ as the entries of a square binary matrix C, where $C[i][j] = \chi_{i,j}$ and whose entries along and below the main diagonal are undefined. We use such entries to represent the existence of crossings between pairs of edges in a graph drawing. Namely, $\chi_{i,j} = 1$ if e_i and e_j belong to E and cross, and $\chi_{i,j} = 0$ if either e_i and e_j belong to E and do not cross or at least one of e_i and e_j does not belong to E. Let χ be the string obtained by concatenating $\chi_{0,1}, \ldots, \chi_{0,m-1}, \ldots, \chi_{i,j}, \ldots, \chi_{m-2,m-1}$. The values of χ are completely determined by x and by whether the considered layout is a book layout or a 2-level drawing. For every $0 \leq a < b \leq m-1$, consider the value $\chi_{a,b}$, where $e_a = (v_i, v_k)$ and $e_b = (v_j, v_\ell)$. In a book layout of G in which the vertex order is $\Pi(x)$, we have that $\chi_{a,b} = 1$ if e_a and e_b belong to E and cross (refer to the conditions in Fig. 29), and $\chi_{a,b} = 0$ if either e_a and e_b belong to E and do not cross or at least one of e_a and e_b does not belong to E. If $D(x)$ corresponds to a 2-level drawing of G, then we have that $\chi_{a,b} = 1$ if e_a and e_b belong to E and cross (i.e., $x_{i,j} \neq x_{k,\ell}$), and $\chi_{a,b} = 0$ if either e_a and e_b belong to E and do not cross (i.e., $x_{i,j} = x_{k,\ell}$) or at least one of e_a and e_b does not belong to E.

Recall that the output of the Skewness Transducer circuit is the superposition $|\theta\rangle |f(\theta)\rangle |e_0\rangle \ldots |e_i\rangle \ldots |e_{m-1}\rangle$ such that, when $f(\theta) = 1$, it holds that $e_i = 1$ if and only if there exists a $j \in [\sigma]$ such that $\theta[j]$ coincides with (the binary representation of) the integer i. In the following, for simplicity, we denote the superposition $|e_0\rangle \ldots |e_i\rangle \ldots |e_{m-1}\rangle$ as $|e\rangle$. Observe that, during the computation for problems TLS and BS, we manage the superposition $|\Theta\rangle = \sum_{\theta \in \mathbb{B}^{\sigma \log m}} c_\theta |\theta\rangle$, which includes all possible states corresponding to a selection of σ edges of G. Specifically, consider any basis state θ that appears in $|\Theta\rangle$, which represents a (multi)subset $N(\theta)$ of size σ of the set $[m]$. Recall that the integers contained in θ are the labels of the edges of G. We denote by $K(\theta)$ the subset of the edges of G whose indices appear in $N(\theta)$. Observe that, if $N(\theta)$ does not contain repeated entries, then $K(\theta)$ is a subset of σ edges of G (with no repeated edges); this occurs exactly when $f(\theta) = 1$. If $N(\theta)$ contains repeated entries, then we say that it is *degenerate*.

5.1 Problem TLCM

We call TLCM the Solution Detector circuit for problem TLCM. Recall that for the TLCM problem, we denote by ρ the maximum number of crossings allowed in the sought 2-level drawing of G.

Lemma 7. *There exists a gate TLCM that, when provided with the input superposition* $|f(\phi)\rangle |x\rangle |0_h\rangle |-\rangle$, *where* $h = 5\frac{m(m-1)}{2} - \log m - \log(m-1) - 2$, *produces the output superposition* $(-1)^{g(x)f(\phi)} |f(\phi)\rangle |x\rangle |0_h\rangle |-\rangle$, *where* $g(x) = 1$ *if* $D(x)$ *is not degenerate and the 2-level*

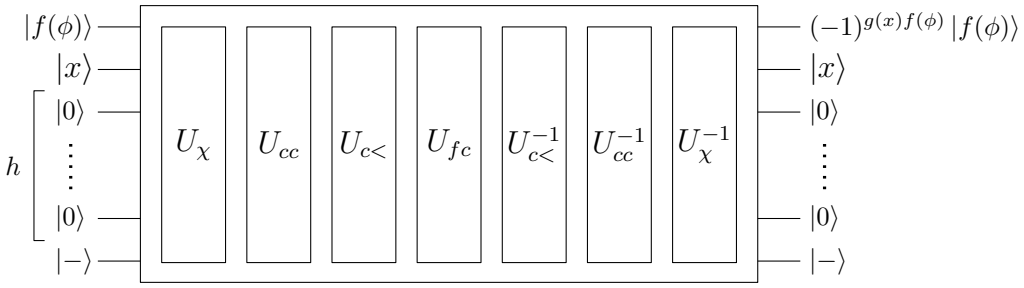
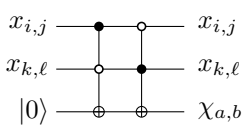


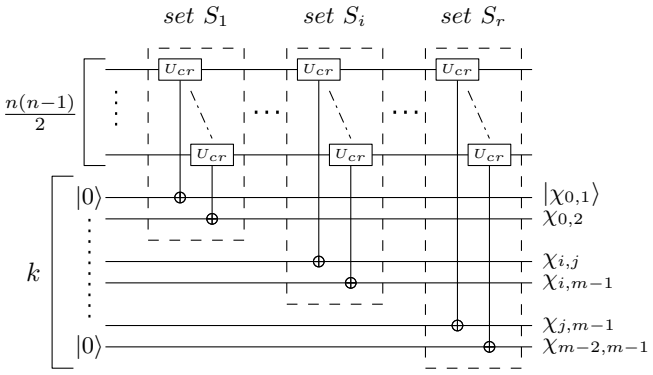
Figure 9: TLCM Oracle Pipeline.

drawing $D(x)$ of G has at most ρ crossings. TLCM has $O(m^2)$ circuit complexity, $O(n^2)$ depth, and $O(m^2)$ width.

Proof of Lemma 7. Gate TLCM uses four gates: TL-Cross Finder U_χ, Cross Counter U_{cc}, Cross Comparator $U_{c<}$, and Final Check U_{fc}, followed by the inverse gates $U_{c<}^{-1}$, U_{cc}^{-1}, and U_χ^{-1}. Refer to Fig. 9.



(a) Gate U_{cr}



(b) Gate U_χ

Figure 10: The gate U_{cr} (a) and gate U_χ (b). In (b), it holds $k = \frac{m(m-1)}{2}$.

TL-cross finder. The purpose of U_χ is to compute the crossings in $D(x)$ (under the assumption that $D(x)$ is not degenerate), determined by the vertex order corresponding to x; refer to Fig. 10b. When provided with the input superposition $|x\rangle|0_k\rangle$, where $k = \frac{m(m-1)}{2}$, the gate U_χ produces the output superposition $|x\rangle|\chi\rangle$.

The gate U_χ exploits the auxiliary gate U_{cr}, whose purpose is to check if two edges cross; refer to Fig. 10a. When provided with the input superposition $|x_{i,j}\rangle|x_{k,\ell}\rangle|0\rangle$, the gate produces the output superposition $|x_{i,j}\rangle|x_{k,\ell}\rangle|\chi_{a,b}\rangle$, where $e_a = (u_i, v_k)$, $e_b = (u_j, v_\ell)$, and $\chi_{a,b} = x_{i,j} \oplus x_{k,\ell}$ (which is 1 if and only if e_a and e_b cross in $D(x)$). It is implemented using two Toffoli gates with three inputs and outputs. The first one is activated when the qubit $|x_{i,j}\rangle$ is equal to $|1\rangle$ and the qubit $|x_{k,\ell}\rangle$ is equal to $|0\rangle$. The second one is activated when the qubit $|x_{i,j}\rangle$ is equal to $|0\rangle$ and the qubit $|x_{k,\ell}\rangle$ is equal to $|1\rangle$. U_{cr} has $O(1)$ circuit complexity, depth, and width.

The gate U_χ works as follows. Consider that if two variables $x_{i,j}$ and $x_{k,\ell}$ are compared to determine whether the edges (u_i, v_k) and (u_j, v_ℓ) cross, none of these variables can be compared with another variable at the same time. Therefore, we partition the pairs of such variables using Corollary 1.1 (with $k = 2$ and $|X| = \frac{n(n-1)}{2}$) into $r \in O(n^2)$ classes $S_1, \ldots, S_r$ each containing at most $\frac{n(n-1)}{4}$ disjoint pairs. For $i = 1, \ldots, r$, the gate U_χ executes in parallel a U_{cr} gate, for each pair $(x_{i,j}, x_{k,\ell})$ in S_i (refer to Fig. 10b), in order to output the qubit $|\chi_{a,b}\rangle$. U_χ has $O(n^4)$ circuit complexity, $O(n^2)$ depth, and $O(n^2)$ width.

CROSS COUNTER. The purpose of gate U_{cc} is to count the total number of crossings in the drawing $D(x)$. When provided with the input superposition $|\chi\rangle\,|0_h\rangle\,|0_k\rangle$, where $h = \log m + \log(m-1)$ and $k = 2m(m-1) - 2(\log m + \log(m-1)) - 2$, the gate U_{cc} produces the output superposition $|\chi\rangle\,|\sigma(x)\rangle\,|0_h\rangle$, where $\sigma(x) = \sum_{e_i,e_j \in E} \chi_{i,j}$ is a binary integer of length $\log m + \log(m-1)$ representing the total number of crossings. The gate U_{cc} is an instance of the gate U_{1s} (refer to Lemma 3), where the qubits of the superposition $|\chi\rangle = |\chi_{0,1}\rangle \ldots |\chi_{0,m-1}\rangle \ldots |\chi_{i,j}\rangle \ldots |\chi_{m-2,m-1}\rangle$ play the role of the qubits of the superposition $|b_0 \ldots b_{t-1}\rangle$, where $t = m$, which forms part of the input of U_{1s}. Observe that the number of crossings in $D(x)$ is at most $\frac{m(m-1)}{2}$. Therefore, the number $\sigma(x)$ of crossings in $D(x)$ can be represented by a binary string of length $h \leq \log m + \log(m-1)$. By Lemma 3, replacing $t = \frac{m(m-1)}{2}$, we get that the parameter $k = 2m(m-1) - 2(\log m + \log(m-1)) - 2$. By Lemma 3, gate U_{cc} has $O(m^2)$ circuit complexity, $O(\log^2 m)$ depth, and $O(m^2)$ width.

CROSS COMPARATOR. The purpose of gate $U_{c<}$ is to verify if the total number of crossings $\sigma(x)$ in $D(x)$ computed by gate U_{cc} is less than the allowed number of crossings ρ. When provided with the input superposition $|\sigma(x)\rangle\,|\rho\rangle\,|0_h\rangle\,|0\rangle$, where $h = \log m + \log(m-1)$, the gate $U_{c<}$ produces the output superposition $|\sigma(x)\rangle\,|\rho\rangle\,|0_h\rangle\,|g(x)\rangle$, where $g(x) = 1$ if $D(x)$ is not degenerate and $\sigma(x) < \rho$. The gate $U_{c<}$ is an instance of the gate $U_<$ (refer to Lemma 2), where the $h = \log m + \log(m-1)$ qubits of the superposition $|\sigma(x)\rangle$ play the role of the qubits of the superposition $|\phi[i]\rangle$ and where h qubits initialized to the binary representation of ρ play the role of $|\phi[j]\rangle$. By Lemma 2, gate $U_{c<}$ has $O(\log m)$ circuit complexity, depth, and width.

FINAL CHECK. The purpose of gate U_{fc} is to check whether the current basis state is the encoding of an admissible solution, i.e., whether the 2-level drawing $D(x)$ of G is not degenerate and it has at most ρ crossings. Refer to Fig. 11. When provided with the input superposition $|f(\phi)\rangle\,|g(x)\rangle\,|-\rangle$, the gate U_{fc} produces the outputs superposition $|f(\phi)\rangle\,|g(x)\rangle\,(-1)^{g(x)f(\phi)}\,|-\rangle$. U_{fc} exploits a Toffoli gate with three inputs and outputs. The control qubits are $|f(\phi)\rangle$ and $|g(x)\rangle$, and the target qubit is $|-\rangle$. When $f(\phi) = g(x) = 1$, the target qubit is transformed into the qubit $-|-\rangle$. Otherwise it leaves unchanged. Gate U_{fc} has $O(1)$ circuit complexity, depth, and width.

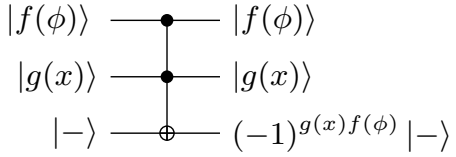


Figure 11: Gate U_{fc}.

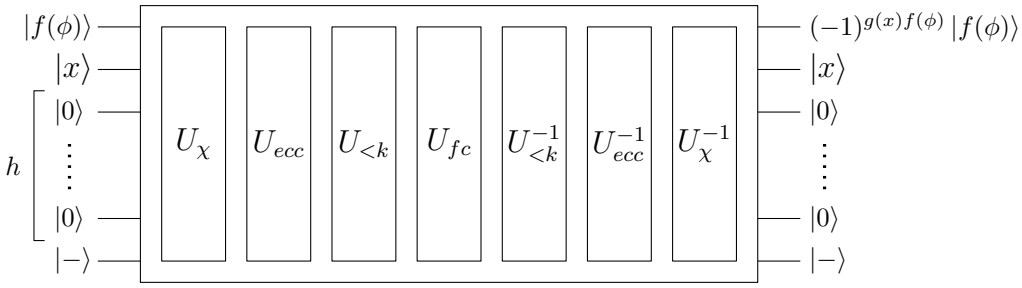


Figure 12: TLKP Oracle Pipeline.

The inverse circuits. The purpose of circuits $U_{c<}^{-1}$, U_{cc}^{-1}, and U_χ^{-1} is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 7, observe that the gates U_χ, U_{cc}, $U_{c<}$, and U_{fc} verify all the necessary conditions for which $D(x)$ has at most ρ crossings, under the assumption that $D(x)$ is not degenerate. Therefore, the sign of the output superposition of gate TLCM, which is determined by the expression $(-1)^{g(x)f(\phi)}$, is positive when either $D(x)$ is degenerate or $D(x)$ is not degenerate and the number of crossings $\sigma(x)$ in $D(x)$ is larger than ρ, and it is negative when $D(x)$ is not degenerate and the number of crossings $\sigma(x)$ in $D(x)$ is smaller than ρ. The bound on the circuit complexity descends from the circuit complexity of the gate U_{cc}, the bound on the depth descends from the depth of the gate U_χ, and the bound on the width descends from the width of U_{cc}. □

5.2 Problem TLKP

We call TLKP the Solution Detector circuit for problem TLKP. Recall that for TLKP problem, we denote by k the maximum number of crossings allowed for each edge in the sought 2-level drawing of G.

Lemma 8. *There exists a gate TLKP that, when provided with the input superposition* $|f(\phi)\rangle|x\rangle|0_h\rangle|-\rangle$, *where* $h = \frac{m(m-1)}{2} + 4m - 2\log m - 4 + m(\log m + 1)$, *outputs the superposition* $(-1)^{g(x)f(\phi)}|f(\phi)\rangle|x\rangle|0_h\rangle|-\rangle$, *where* $g(x) = 1$ *if* $D(x)$ *is not degenerate and each edge of the 2-level drawing* $D(x)$ *of* G *has at most* k *crossings. TLKP has* $O(m^2)$ *circuit complexity,* $O(m\log^2 m)$ *depth and* $O(m)$ *width.*

Proof of Lemma 8. Gate TLKP uses four gates: tl-cross finder U_χ, edge cross counter U_{ecc}, edge cross comparator $U_{<k}$, and final check U_{fc}, followed by the inverse gates U_χ^{-1}, U_{ecc}^{-1}, and $U_{<k}^{-1}$. Refer to Fig. 12.

tl-cross finder. For the definition of gate U_χ, refer to the proof of Lemma 7. Recall that the purpose of U_χ is to compute the crossings in $D(x)$ (under the assumption that $D(x)$ is not degenerate), determined by the vertex order corresponding to x. Also Recall that when provided with the input superposition $|x\rangle|0_k\rangle$, the gate U_χ produces the output superposition $|x\rangle|\chi\rangle$.

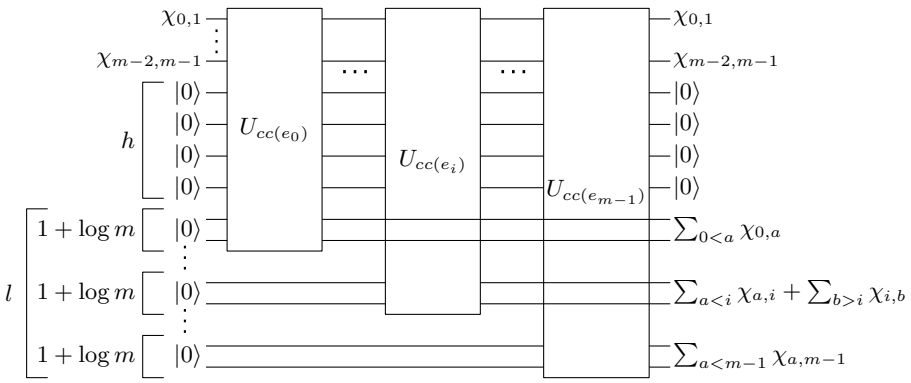


Figure 13: The gate U_{ecc}.

EDGE CROSS COUNTER. The purpose of gate U_{ecc} is to count, for each edge $e_i \in E$, the total number of crossings of each edge e_i in the drawing $D(x)$. When provided with the input superposition $|\chi\rangle\,|0_h\rangle\,|0_l\rangle$, where $h = 4m-2\log m-4$ and $l = m(1+\log m)$, the gate U_{ecc} produces the output superposition $|\chi\rangle\,|0_h\rangle\,|\sigma_{e_0}(x)\rangle \dots |\sigma_{e_i}(x)\rangle \dots |\sigma_{e_{m-1}}(x)\rangle$, where $\sigma_{e_i}(x) = \sum_{a<i} \chi_{a,i} + \sum_{b>i} \chi_{i,b}$ is a binary integer of length $1+\log m$ representing the total number of crossings of the edge e_i in $D(x)$. The gate U_{ecc} exploits the auxiliary gate $U_{cc(e_i)}$, whose purpose, for each edge e_i, is to compute the binary integer $\sigma_{e_i}(x)$. When provided with the input superposition $|\chi_{0,i}\rangle \dots |\chi_{i,i+1}\rangle \dots |\chi_{i,m-1}\rangle\,|0_h\rangle\,|0_{1+\log m}\rangle$, the gate $U_{cc(e_i)}$ produces the output superposition $|\chi_{0,i}\rangle \dots |\chi_{i,i+1}\rangle \dots |\chi_{i,m-1}\rangle\,|0_h\rangle\,|\sigma_{e_i}(x)\rangle$. The gate $U_{cc(e_i)}$ is an instance of the gate U_{1s} (refer to Lemma 3), where the qubits of the superposition $|\chi_{0,i}\rangle \dots |\chi_{i,i+1}\rangle \dots |\chi_{i,m-1}\rangle$ play the role of the qubits of the superposition $|b_0 \dots b_{t-1}\rangle$, with $t = m$, which forms part of the input of U_{1s}. Observe that, for each edge e_i, the number of crossings of e_i in $D(x)$ is at most $m-1$. Therefore, $\sigma_{e_i}(x)$ can be represented by a binary string of length $1+\log m$. By Lemma 3, gate $U_{cc(e_i)}$ has $O(m)$ circuit complexity, $O(\log^2 m)$ depth , and $O(m)$ width . Gate U_{ecc} works as follows; refer to Fig. 13. Gate U_{ecc} executes in sequence gates $U_{cc(e_0)}$, $U_{cc(e_1)}, \dots, U_{cc(e_{m-1})}$. Gate U_{ecc} has circuit complexity $O(m^2)$, depth $O(m\log^2 m)$, and width $O(m)$.

EDGE CROSS COMPARATOR. The purpose of $U_{<k}$ is to verify, for each edge $e_i \in E$, if the total number of crossings $\sigma_{e_i}(x)$ of e_i in $D(x)$ computed by U_{ecc} is less than the allowed number of crossings for each edge k; refer to Fig. 14. When provided with the input superposition $|\sigma_E(x)\rangle\,|k\rangle\,|0_h\rangle\,|0_m\rangle\,|0\rangle$, where $\sigma_E(x) = \sigma_{e_0}(x) \dots \sigma_{e_i}(x) \dots \sigma_{e_{m-1}}(x)$ and $h = \log m + 1$, the gate $U_{<k}$ produces the output superposition $|\sigma_E(x)\rangle\,|k\rangle\,|0_h\rangle\,|0_m\rangle\,|g(x)\rangle$, where $g(x) = 1$ if $D(x)$ is not degenerate and $\sigma_{e_i}(x) < k$ for each $e_i \in E$. The gate $U_{<k}$ exploits m instances of gate $U_<$ (refer to Lemma 2), where the qubits of the superposition $|\sigma_{e_i}(x)\rangle$ play the role of the qubits of the superposition $|\phi[i]\rangle$ and the qubits initialized to the binary representation of k play the role of $|\phi[j]\rangle$. Recall that by Lemma 2, gate $U_<$ has $O(\log m)$ circuit complexity, depth, and width. Each of the m gates $U_<$ provides an answer qubit ($|\kappa_i\rangle$), which is equal to 1 if and only if $\sigma_{e_i}(x) < k$ for the considered edge e_i. At the end of the m-th computation a Toffoli gate with $m+1$ inputs and outputs is applied to check if, for each $e_i \in E$, $\sigma_{e_i}(x) < k$. Gate $U_{<k}$ has $O(m\log m)$ circuit complexity, $O(m\log m)$ depth, and $O(m)$ width.

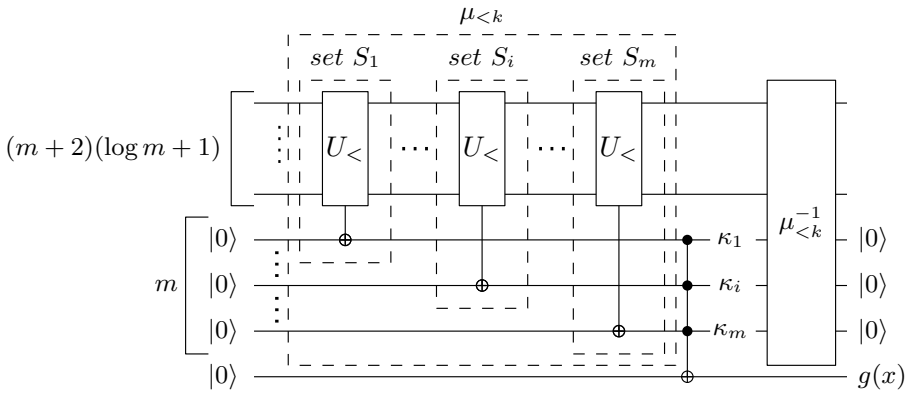


Figure 14: $U_{<k}$.

FINAL CHECK. The purpose of gate U_{fc} is to check wether the current basis state is the encoding of an admissible solution, i.e., whether the 2-level drawing $D(x)$ of G is not degenerate and each edge has at most k crossings. Refer to Fig. 11. When provided with the input superposition $|f(\phi)\rangle\,|g(x)\rangle\,|-\rangle$, the gate U_{fc} produces the outputs superposition $|f(\phi)\rangle\,|g(x)\rangle\,(-1)^{f(\phi)g(x)}\,|-\rangle$. U_{fc} exploits a Toffoli gate with three inputs and outputs. The control qubits are $|f(\phi)\rangle$ and $|g(x)\rangle$, and the target qubit is $|-\rangle$. When at least one of $f(\phi)$ and $g(x)$ are equal to 0, the target qubit leaves unchanged. On the other hand, when $f(\phi)=g(x)=1$, the target qubits is transformed into the qubit $-\,|-\rangle$. Gate U_{fc} has $O(1)$ circuit complexity, depth and width.

The inverse circuits. The purpose of circuits $U_{<k}^{-1}$, $U_{cc(E)}^{-1}$, and U_{χ}^{-1} is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 8, observe that the gates U_{χ}, $U_{cc(E)}$, $U_{<k}$, and U_{fc} verify all the necessary conditions for which $D(x)$, for each edge of G, has at most k crossings, under the assumption that $D(x)$ is not degenerate. Therefore, the sign of the output superposition of gate TLKP, which is determined by the expression $(-1)^{g(x)f(\phi)}$, is positive when either $D(x)$ is degenerate or $D(x)$ is not degenerate and the number of crossings $\sigma_{e_i}(x)$ in $D(x)$ is larger than k, for some edge $e_i \in E$, and it is negative when $D(x)$ is not degenerate and the number of crossings $\sigma_{e_i}(x)$ in $D(x)$ is smaller than k, for each edge $e_i \in E$. The bounds on the circuit complexity, depth, and width descend from the circuit complexity, depth, and width of the gate $U_{cc(E)}$. □

5.3 Problem TLQP

We call TLQP the SOLUTION DETECTOR circuit for problem TLQP.

Lemma 9. *There exists a gate TLQP that, when provided with the input superposition* $|f(\phi)\rangle\,|x\rangle\,|0_h\rangle\,|-\rangle$, *where* $h \in O(m^4)$, *produces the output superposition* $(-1)^{f(\phi)g(x)}\,|f(\phi)\rangle\,|x\rangle\,|0_h\rangle\,|-\rangle$, *and* $g(x)=1$ *if* $D(x)$ *is not degenerate and the 2-level drawing* $D(x)$ *of* G *is quasi-planar. TLQP has* $O(m^6)$ *circuit complexity,* $O(m^4)$ *depth, and* $O(m^2)$ *width.*

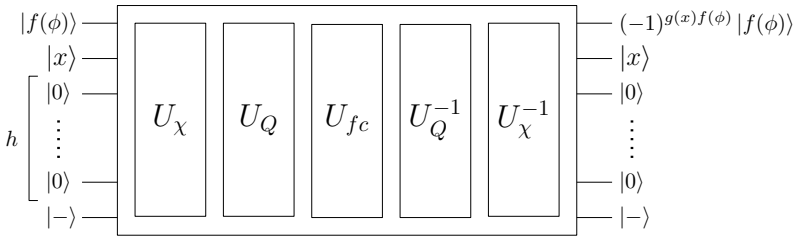


Figure 15: TLQP Oracle Pipeline.

Proof of Lemma 9. Gate TLQP executes three gates: TL-CROSS FINDER U_χ, QUASI-PLANARITY TESTER U_Q, and FINAL CHECK U_{fc}, followed by their inverse gates U_Q^{-1} and U_χ^{-1}. Refer to Fig. 15.

TL-CROSS FINDER. For the definition of gate U_χ, refer to the proof of Lemma 7. Recall that the purpose of U_χ is to compute the crossings in $D(x)$ (under the assumption that $D(x)$ is not degenerate), determined by the vertex order corresponding to x. Also Recall that when provided with the input superposition $|x\rangle|0_k\rangle$, the gate U_χ produces the output superposition $|x\rangle|\chi\rangle$.

QUASI-PLANARITY TESTER. The purpose of gate U_Q is to verify the absence of any three edges that pairwise cross in $D(x)$; refer to Fig. 17. When provided with the input the superposition $|\chi\rangle|0_h\rangle|0\rangle$, where $h \in O(m^4)$, the gate U_Q produces the output superposition $|\chi\rangle|0_h\rangle|g(x)\rangle$, where $g(x) = 1$ if $D(x)$ is not degenerate and there are not three edges that pairwise cross in $D(x)$.

The gate U_Q exploits the auxiliary gate $U_{q\chi}$, whose purpose is to check if three edges pairwise cross; refer to Fig. 16. When provided with the input superposition $|\chi_{i,j}\rangle|\chi_{i,k}\rangle|\chi_{j,k}\rangle|0\rangle$, the gate provide the output superposition $|\chi_{i,j}\rangle|\chi_{i,k}\rangle|\chi_{j,k}\rangle|q\chi_{i,j,k}\rangle$, where $q\chi_{i,j,k} = \chi_{i,j} \wedge \chi_{i,k} \wedge \chi_{j,k}$ (which is 1 if and only if e_i, e_j, and e_k pairwise cross in $D(x)$). It is implemented using a Toffoli gate with four inputs and outputs, which is activated when $\chi_{i,j} = \chi_{i,k} = \chi_{j,k} = 1$. The circuit complexity, depth, and width of $U_{q\chi}$ is $O(1)$.

The gate U_Q works as follows. Consider that if three variables $\chi_{i,j}$, $\chi_{i,k}$, and $\chi_{j,k}$ are compared to determine whether the edges e_i, e_j, and e_k pairwise cross, none of these variables can be compared with another variable at the same time. Therefore, we partition the pairs of such variables using Corollary 1.1 (with $k = 3$ and $|X| = \frac{m(m-1)}{2}$) into $p \in O(m^4)$ classes $S_1, \ldots, S_p$ each containing at most $\frac{m(m-1)}{6}$ unordered disjoint triples. For $i = 1, \ldots, p$, the gate U_Q executes in parallel a gate $U_{q\chi}$ for each triple $\{\chi_{i,j}, \chi_{i,k}, \chi_{j,k}\}$ in S_i (refer to Fig. 17). All the last output qubits of the $U_{q\chi}$ gates in S_i enter a Toffoli gate that outputs a qubit $|q_i\rangle$ such that $q_i = 1$ if and only if all of such qubits are equal to $|0\rangle$, i.e., it does not exist a triple of edges that pairwise cross. In order to allow the reuse of the ancilla qubit, except for the qubit $|q_i\rangle$, gate U_Q executes in parallel a gate $U_{q\chi}^{-1}$ for each triple in S_i. All the qubits $|q_i\rangle$, with $i = 1, \ldots, p$, enter in cascade a Toffoli gate, with three inputs and outputs, that checks that all of them are equal to $|1\rangle$, i.e., there exist not three edges that pairwise cross. The output qubit of the last Toffoli gate is the qubit $|g(x)\rangle$. The gate U_Q has circuit complexity $O(m^6)$, $O(m^4)$ depth, and $O(m^2)$ width.

FINAL CHECK. We use a gate U_{fc} to check whether the current basis state is the encoding of an admissible solution, i.e., whether the 2-level drawing $D(x)$ of G is not degenerate and the 2-level drawing $D(x)$ of G is quasi-planar. Refer to Fig. 11 and to Lemma 7. When provided with the input superposition $|f(\phi)\rangle|g(x)\rangle|-\rangle$, the gate U_{fc} produces the outputs superposition

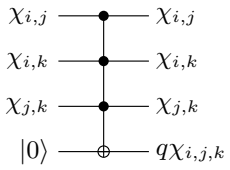


Figure 16: Gate $U_{q\chi}$

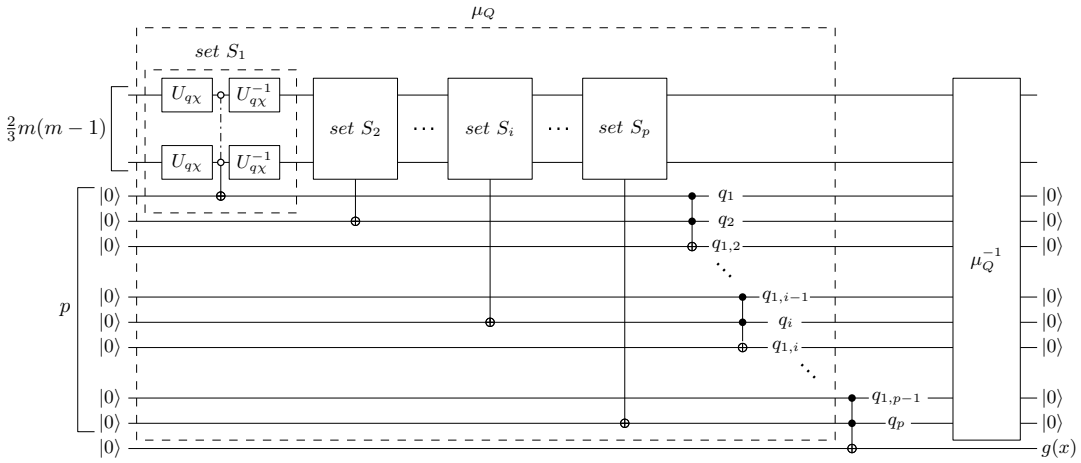


Figure 17: Gate U_Q

$|f(\phi)\rangle\,|g(x)\rangle\,(-1)^{f(\phi)g(x)}\,|-\rangle$. U_{fc} exploits a Toffoli gate with three inputs and outputs. The control qubits are $|f(\phi)\rangle$ and $|g(x)\rangle$, and the target qubit is $|-\rangle$. When at least one of $f(\phi)$ and $g(x)$ are equal to 0, the target qubit leaves unchanged. On the other hand, when $f(\phi) = g(x) = 1$, the target qubits is transformed into the qubit $-|-\rangle$. Gate U_{fc} has $O(1)$ circuit complexity, depth and width.

The inverse circuits. The purpose of circuits U_Q^{-1} and U_χ^{-1} is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 9, observe that the gates U_χ, U_Q, and U_{fc} verify all the necessary conditions for which $D(x)$ is a quasi-planar drawing of G, under the assumption that $D(x)$ is not degenerate. Therefore, the sign of the output superposition of gate TLQP, which is determined by the expression $(-1)^{g(x)f(\phi)}$, is positive when either $D(x)$ is degenerate or $D(x)$ is not degenerate and it is not a quasi-planar drawing of G, and it is negative when $D(x)$ is not degenerate and $D(x)$ is a quasi-planar drawing of G. The bounds on the circuit complexity, depth, and width descend from the circuit complexity, depth, and width of the gate U_Q. □

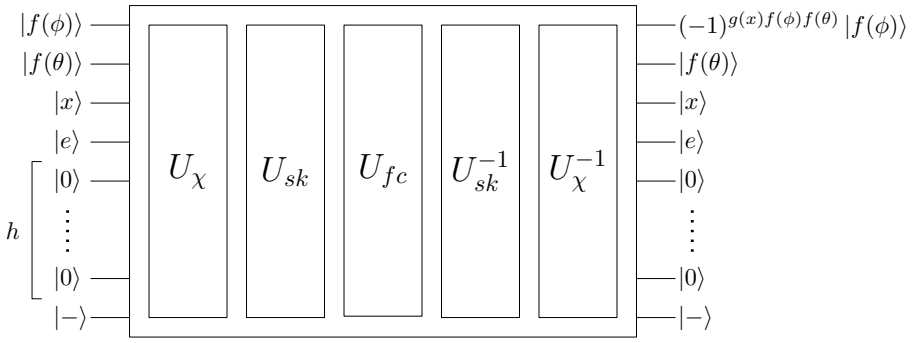


Figure 18: TLS Oracle Pipeline.

5.4 Problem TLS

Recall that the purpose of the basis state $|\theta\rangle$ is to represent a subset $K(\theta)$ of the edges of G of size at most σ, each labeled with an integer in $[m]$, and that we denote by $N(\theta)$ the set of indices of the edges in $K(\theta)$. Also, recall that we denote by $D(x)$ the 2-level drawing of G associated with the vertex order corresponding to x. In the following, for a subgraph G' of G, we use the notation $D(x, G')$ to denote the 2-level drawing of G' induced by $D(x)$.

We call TLS the Solution Detector circuit for problem TLS.

Lemma 10. *There exists a gate TLS that, provided with the input superposition* $|f(\phi)\rangle|f(\theta)\rangle|x\rangle|e\rangle|0_h\rangle|-\rangle$, *where* $h \in O(m^2)$, *produces the output superposition* $(-1)^{g(x)f(\phi)f(\theta)}|f(\phi)\rangle|f(\theta)\rangle|x\rangle|e\rangle|0_h\rangle|-\rangle$, *such that, if* $D(x)$ *and* $N(\theta)$ *are not degenerate, then* $g(x) = 1$ *if and only if* $D(x, G')$ *is planar, where* $G' = (V, E \setminus K(\theta))$. *Gate TLS has* $O(m^2)$ *circuit complexity,* $O(m)$ *depth, and* $O(m)$ *width.*

Proof of Lemma 10. Gate TLS uses three gates: TL-cross finder U_χ, skewness cross tester U_{sk}, and final check U_{fc}, followed by the inverse gates U_{sk}^{-1} and U_χ^{-1}. Fig. 18

TL-cross Finder. For the definition of gate U_χ, refer to the proof of Lemma 7. Recall that the purpose of U_χ is to compute the crossings in $D(x)$ (under the assumption that $D(x)$ is not degenerate), determined by the vertex order corresponding to x. Also Recall that when provided with the input superposition $|x\rangle|0_k\rangle$, the gate U_χ produces the output superposition $|x\rangle|\chi\rangle$.

skewness cross tester. Consider the subgraph G' of G obtained by removing from G all the edges in $K(\theta)$. The purpose of gate U_{sk} is to determine which of the crossings stored in $|\chi\rangle$ involve pairs of edges that are both absent from $|e\rangle$. In fact, all the edges not in $|e\rangle$ form the edge set of G'. Therefore, U_{sk} verifies whether the 2-layer drawing $D(x, G')$ of G' is planar; refer to Fig. 19. When provided with the input superposition $|\chi\rangle|e\rangle\left|0_{\frac{m}{2}+p}\right\rangle|0\rangle$, where $p \in O(m)$, the gate U_{sk} produces the output superposition $|\chi\rangle|e\rangle\left|0_{\frac{m}{2}+p}\right\rangle|g(x)\rangle$, such that, if $D(x)$ and $N(\theta)$ are not degenerate, then $g(x) = 1$ if and only if $D(x, G')$ is planar.

The gate U_{sk} exploits the auxiliary gate U_k, whose purpose is to check if any two edges in G' cross; refer to Fig. 20. When provided with the input superposition $|e_a\rangle|e_b\rangle|\chi_{a,b}\rangle|0\rangle$, the gate U_k provides the output superposition $|e_a\rangle|e_b\rangle|\chi_{a,b}\rangle|s_{a,b}\rangle$, where $s_{a,b} = \neg e_a \wedge \neg e_b \wedge \chi_{a,b}$ (which is 1 if and only if e_a and e_b cross in $D(x, G')$). The gate U_k is implemented using a Toffoli gate with four

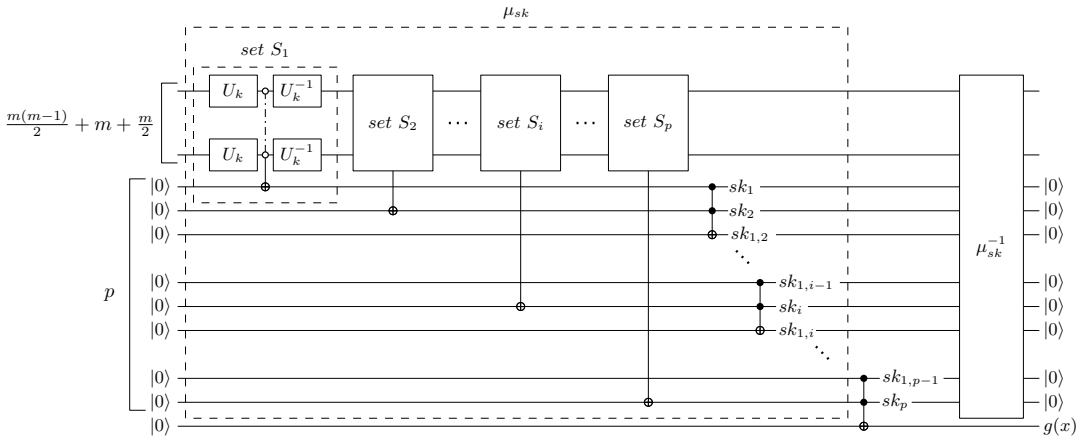


Figure 19: Gate U_{sk}.

inputs and outputs. The control qubits are $|e_a\rangle$, $|e_b\rangle$, and $|\chi_{a,b}\rangle$. The target qubit is set to $|0\rangle$. The gate is activated when $e_a = e_b = 0$ and $\chi_{a,b} = 1$. The circuit complexity, depth, and width of U_k is $O(1)$.

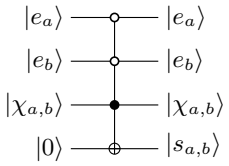


Figure 20: Gate U_k

The gate U_{sk} works as follows. Consider that if two variables e_a and e_b are compared to determine whether the corresponding edges cross in $D(x, G')$, none of these variables can be compared with another variable at the same time. Therefore, we partition the pairs of such variables using Corollary 1.1 (with $k = 2$ and $|X| = m$) into $p \in O(m)$ classes $S_1, \ldots, S_p$ each containing at most $\frac{m}{2}$ disjoint pairs. For $i = 1, \ldots, p$, the gate U_{sk} executes in parallel a gate U_k, for each pair of variables e_a, e_b in S_i (together with the corresponding qubit $\chi_{a,b}$), in order to output the qubit $|\neg e_a \wedge \neg e_b \wedge \chi_{a,b}\rangle$. All the last output qubits of the U_k gates in S_i enter a Toffoli gate that outputs a qubit $|sk_i\rangle$ such that $sk_i = 1$ if and only if all of them are equal to $|0\rangle$, i.e., there exist no two crossing edges among the pairs in S_i. In order to allow the reuse of the ancilla qubit, except for the qubit $|sk_i\rangle$, gate U_{sk} executes in parallel a gate U_k^{-1} for each pair in S_i. To check if all qubits $|sk_i\rangle$ are equal to $|1\rangle$, for $i = 1, \ldots, p$, we use a series of Toffoli gates T_i, each with three inputs and outputs. The first Toffoli gate T_1 receives in input the qubits $|sk_1\rangle$, $|sk_2\rangle$, and a qubit set to $|0\rangle$, and outputs the qubit $|sk_{1,2}\rangle = |sk_1 \wedge sk_2\rangle$. For $i = 2, \ldots, p$, the Toffoli gate T_i receives in input the qubits $|sk_{1,i-1}\rangle$, $|sk_i\rangle$, and a qubit set to $|0\rangle$, and outputs the qubit $|sk_{1,i}\rangle = |sk_{1,i-1} \wedge sk_i\rangle$. The output qubit $|sk_{1,p}\rangle$ of the last Toffoli gate T_p is the qubit $|g(x)\rangle$. The gate U_{sk} has circuit complexity $O(m^2)$, depth $O(m)$, and width $O(m)$.

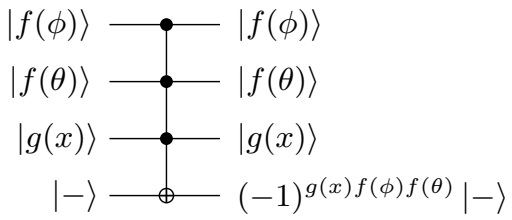


Figure 21: Gate U_{fc}.

FINAL CHECK. The purpose of gate U_{fc} is to check whether the current basis state is the encoding of an admissible solution, i.e., whether the 2-level drawing $D(x)$ and the set of indices $N(\theta)$ are both not degenerate and the 2-level drawing $D(x, G')$ of G' is planar. See Fig. 21. When provided with the input superposition $|f(\phi)\rangle |f(\theta)\rangle |g(x)\rangle |-\rangle$, the gate U_{fc} produces the outputs superposition $|f(\phi)\rangle |f(\theta)\rangle |g(x)\rangle (-1)^{g(x)f(\phi)f(\theta)} |-\rangle$. Gate U_{fc} exploits a Toffoli gate with four inputs and outputs. The control qubits are $|f(\phi)\rangle$, $|f(\theta)\rangle$, and $|g(x)\rangle$, and the target qubit is $|-\rangle$. When at least one of $f(\phi)$, $f(\theta)$, and $g(x)$ are equal to 0, the target qubit leaves unchanged. On the other hand, when $f(\phi) = f(\theta) = g(x) = 1$, the target qubit is transformed into the qubit $-|-\rangle$. Gate U_{fc} has $O(1)$ circuit complexity, depth, and width.

The inverse circuits. The purpose of circuits U_{sk}^{-1} and U_{χ}^{-1} is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 10, observe that the gates U_{χ}, U_{sk}, and U_{fc} verify all the necessary conditions for which $D(x, G')$ is a 2-level planar drawing of G', under the assumption that $D(x)$ and $N(\theta)$ are not degenerate. Therefore, the sign of the output superposition of gate TLS, which is determined by the expression $(-1)^{g(x)f(\phi)f(\theta)}$, is defined as follows. It is positive when either $D(x)$ or $N(\theta)$ are degenerate or $D(x)$ and $N(\theta)$ are not degenerate and the drawing $D(x, G')$ of G' is not planar. It is negative when $D(x)$ and $N(\theta)$ are not degenerate and the 2-level drawing $D(x, G')$ of G' is planar. The bounds on the circuit complexity, depth, and width of gate TLS descend from those of gate U_{sk}. □

5.5 Problem OPCM

We call OPCM the SOLUTION DETECTOR circuit for problem OPCM. Recall that for the OPCM problem, we denote by ρ the maximum number of crossings allowed in the sought 1-page layout of G. Also, recall that we denote by $\Pi(x)$ the vertex order along the spine of a book layout of G defined by the vertex order corresponding to x.

Lemma 11. *There exists a gate OPCM that, when provided with the input superposition* $|f(\phi)\rangle |x\rangle |0_h\rangle |-\rangle$, *where* $h \in O(m^2)$, *produces the output superposition* $(-1)^{g(x,f(\phi))} |f(\phi)\rangle |x\rangle |0_h\rangle |-\rangle$, *where* $g(x) = 1$ *if* $\Pi(x)$ *is not degenerate and the* 1*-page layout of* G *defined by* $\Pi(x)$ *has at most* ρ *crossings. OPCM has* $O(n^8)$ *circuit complexity,* $O(n^6)$ *depth, and* $O(m^2)$ *width.*

Proof of Lemma 11. Gate OPCM executes four gates: OP-CROSS FINDER U_{χ_p}, CROSS COUNTER U_{cc}, CROSS COMPARATOR $U_{c<}$, and FINAL CHECK U_{fc}, followed by the inverse gate U_{cc}^{-1}, $U_{c<}^{-1}$, and $U_{\chi_p}^{-1}$. Refer to Fig. 22.

OP-CROSS FINDER. The purpose of U_{χ_p} is to compute the crossings in the 1-page layout of G defined by $\Pi(x)$, determined by the vertex order corresponding to x; refer to Fig. 23. When provided with the input superposition $|x\rangle\,|0_k\rangle$, where $k = \frac{m(m-1)}{2}$, the gate $U_{p\chi}$ produces the output superposition $|x\rangle\,|\chi\rangle$.

The gate U_{χ_p} exploits the auxiliary gate U_λ, whose purpose is to check if two edges cross in the 1-page layout of G defined by $\Pi(x)$; refer to Fig. 23a. When provided with the input superposition $|x_{i,k}\rangle\,|x_{k,j}\rangle\,|x_{j,n}\rangle\,|x_{i,n}\rangle\,|0\rangle$, the gate U_λ produces the output superposition $|x_{i,k}\rangle\,|x_{k,j}\rangle\,|x_{j,n}\rangle\,|x_{i,n}\rangle\,|\chi_{a,b}\rangle$, where $e_a = (v_i, v_j)$, $e_b = (v_k, v_n)$, and $\chi_{a,b}$ equals 1 if and only if e_a and e_b cross in the 1-page layout of G defined by $\Pi(x)$; refer to Fig. 29 and to the expression $\chi_{a,b}$ in Sect. 6.2. It is implemented using eight Toffoli gates, each with four inputs and outputs. In the following, we assume that $i < k < j < \ell$. As an example, the first gate is activated when $x_{i,\ell} = x_{j,k} = 1$ and $x_{j,\ell} = 0$ (see Fig. 29, top row, first column). The remaining Toffoli gates operate similarly, with each being activated under specific conditions involving the variables $x_{i,k}, x_{i,\ell}, x_{j,k}$ and $x_{j,\ell}$. For a detailed illustration of all cases, refer to Fig. 29.

The gate U_{χ_p} works as follows. Consider that if four variables $x_{i,k}, x_{k,j}, x_{j,n}$ and $x_{i,n}$ are compared to determine whether the edges (v_i, v_j) and (v_k, v_n) cross, none of these variables can be compared with another variable at the same time. Therefore, we partition the pairs of such variables using Corollary 1.1 (with $k = 4$ and $|X| = \frac{n(n-1)}{2}$) into $r \in O(n^6)$ classes $S_1, \ldots, S_r$ each containing at most $\frac{n(n-1)}{8}$ disjoint pairs. For $i = 1, \ldots, r$, the gate U_{χ_p} executes in parallel a U_λ gate, for each quartet $(x_{i,k}, x_{k,j}, x_{j,n}, x_{i,n})$ in S_i (refer to Fig. 23b), in order to outputs the qubit $|\chi_{a,b}\rangle$. U_{χ_p} has circuit complexity $O(n^8)$, depth complexity $O(n^6)$ and width complexity $O(n^2)$.

CROSS COUNTER. The purpose of gate U_{cc}, as mention earlier, is count the total number of crossings in the 1-page layout of G defined by $\Pi(x)$; refer to Lemma 3. Recall that when provided with the input superposition $|\chi\rangle\,|0_h\rangle\,|0_k\rangle$, where $h = \log m + \log(m-1)$ and $k = 2m(m-1) - 2(\log m + \log(m-1)) - 1$, the gate U_{cc} produces the output superposition $|\chi\rangle\,|\sigma(x)\rangle\,|0_h\rangle$.

CROSS COMPARATOR. The purpose of gate $U_{c<}$, as mention earlier, is to verify if the total number of crossings $\sigma(x)$ in the 1-page layout of G defined by $\Pi(x)$ compute by the gate U_{cc} is less than the allowed number of crossings ρ; refer to Lemma 2. Recall that when provided with the input superposition $|\sigma(x)\rangle\,|\rho\rangle\,|0_h\rangle\,|0\rangle$, where $h = \log m + \log(m-1)$, the gate $U_{c<}$ produces the output superposition $|\sigma(x)\rangle\,|\rho\rangle\,|0_h\rangle\,|g(x)\rangle$, where $g(x) = 1$ if $\Pi(x)$ is not degenerate and $\sigma(x) < \rho$.

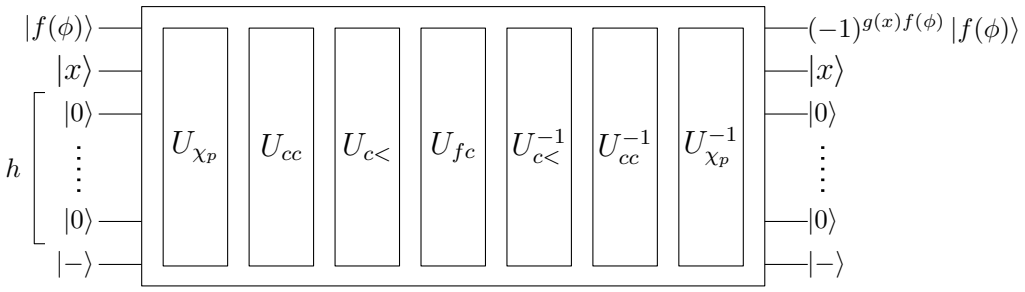
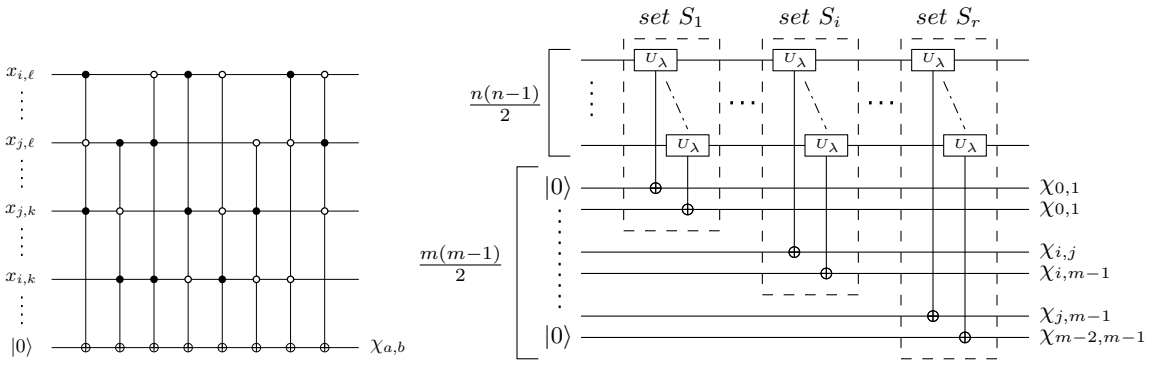


Figure 22: Oracle OPCM.



(a) Gate U_λ (b) Gate U_{χ_p}

Figure 23: The gate U_λ (left) and gate U_{χ_p} (right).

FINAL CHECK. The purpose of gate U_{fc} is to check whether the current basis state is the encoding of an admissible solution, i.e., whether $\Pi(x)$ is not degenerate and the 1-page layout of G defined by $\Pi(x)$ has at most ρ crossings. Refer to Fig. 11. When provided with the input superposition $|f(\phi)\rangle\,|g(x)\rangle\,|-\rangle$, the gate U_{fc} produces the outputs superposition $|f(\phi)\rangle\,|g(x)\rangle\,(-1)^{g(x)f(\phi)}\,|-\rangle$. U_{fc} exploits a Toffoli gate with three inputs and outputs. The control qubits are $|f(\phi)\rangle$ and $|g(x)\rangle$, and the target qubit is $|-\rangle$. When at least one of $f(\phi)$ and $g(x)$ are equal to 0, the target qubit leaves unchanged. On the other hand, when $f(\phi) = g(x) = 1$, the target qubit is transformed into the qubit $-\,|-\rangle$. Gate U_{fc} has $O(1)$ circuit complexity, depth, and width.

The inverse circuits. The purpose of circuits $U_{c<}^{-1}$, U_{cc}^{-1}, and U_{χ}^{-1} is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 11, observe that the gates U_χ, U_{cc}, $U_{c<}$, and U_{fc} verify all the necessary conditions for which the 1-page layout of G defined by $\Pi(x)$ has at most ρ crossings, under the assumption that $\Pi(x)$ is not degenerate. Therefore, the sign of the output superposition of gate OPCM, which is determined by the expression $(-1)^{g(x)f(\phi)}$, is positive when either $\Pi(x)$ is degenerate or $\Pi(x)$ is not degenerate and the number of crossings $\sigma(x)$ in the 1-page layout of G defined by $\Pi(x)$ is larger than ρ, and it is negative only if $\Pi(x)$ is not degenerate and the number of crossings $\sigma(x)$ in the 1-page layout of G defined by $\Pi(x)$ is smaller than ρ. The bound on the circuit complexity descends from the circuit complexity of the gate U_{χ_p}, the bound on the depth descends from the depth of the gate U_{χ_p}, and the bound on the width descends from the width of U_{cc}. □

5.6 Problem BT

We call BT the SOLUTION DETECTOR circuit for problem BT. Recall that for the BT problem, we denote by τ the number of pages allowed in the sought book layout drawing of G.

Recall that during the computation, we manage the superposition $|\Psi\rangle = \sum_{\psi \in \mathbb{B}^{m \log \tau}} c_\psi\,|\psi\rangle$ whose purpose is to represent a coloring of the edges of G with colors in the set $[\tau]$. Specifically, consider any basis state ψ that appears in $|\Psi\rangle$. We denote by $P(\psi)$ the page assignment of the edges of G to τ pages in which, for $i = 0, \ldots, m-1$, the edge e_i is assigned to the page $\psi[i]$.

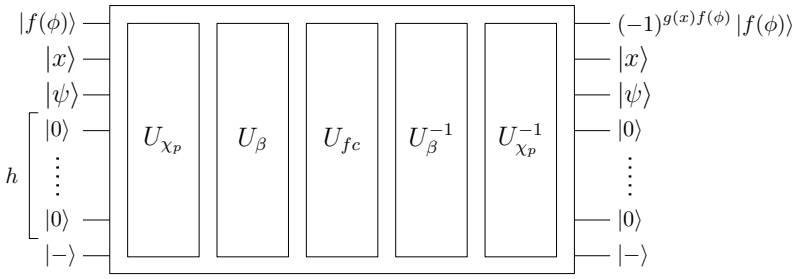


Figure 24: BT Oracle Pipeline.

Lemma 12. *There exists a gate BT that, when provided with the input superposition* $|f(\phi)\rangle|\psi\rangle|x\rangle|0_h\rangle|-\rangle$, *where* $h \in O(m^2)$, *produces the output superposition* $(-1)^{g(x,\psi)f(\phi)}|f(\phi)\rangle|\psi\rangle|x\rangle|0_h\rangle|-\rangle$, *where* $g(x,\psi) = 1$ *if* $\Pi(x)$ *is not degenerate and there exists a* τ*-page book embedding of* G *in which the vertex order is* $\Pi(x)$ *and the page assignment is* $P(\psi)$. *Gate BT has* $O(n^8)$ *circuit complexity,* $O(n^6)$ *depth, and* $O(m)$ *width.*

Proof of Lemma 12. Gate BT uses two gates: OP-CROSS FINDER U_{χ_p}, COLOR TESTER U_β, and FINAL CHECK U_{fc}, followed by the inverse gates U_β^{-1} and $U_{\chi_p}^{-1}$. Refer to Fig. 24.

OP-CROSS FINDER. For the definition of gate U_{χ_p}, refer to the proof of Lemma 11. Recall that the purpose of U_{χ_p} is to compute the crossings in the 1-page layout of G defined by $\Pi(x)$, determined by the vertex order corresponding to x. Also Recall that when provided with the input superposition $|x\rangle|0_k\rangle$, the gate $U_{p\chi}$ produces the output superposition $|x\rangle|\chi\rangle$.

COLOR TESTER. Consider the book layout $D(x,\psi)$ of G defined by the vertex order $\Pi(x)$ and the page assignment is $P(\psi)$. The purpose of gate U_β is to verify if $D(x,\psi)$ is a book embedding of G on τ pages, provided that $\Pi(x)$ is not degenerate; refer to Fig. 26. When provided with the input superposition $|\psi\rangle|\chi\rangle|0_b\rangle|0_r\rangle|0\rangle$, where $b = \frac{m}{2}(2+\log\tau)$ and $r = m-1$, the gate U_β produces the output superposition $|\psi\rangle|\chi\rangle|0_b\rangle|0_r\rangle|g(x,\psi)\rangle$, where $g(x,\psi) = 1$ if $\Pi(x)$ is not degenerate and $D(x,\psi)$ is a book embedding of G on τ pages.

The gate U_β exploits the auxiliary gate $U_{=\lambda}$, whose purpose is to check if two edges that cross have the same color; refer to Fig. 25. When provided with the input superposition

$$|\psi[a][0]\rangle\ldots|\psi[a][\log(\tau)-1]\rangle|\psi[b][0]\rangle\ldots|\psi[a][\log(\tau)-1]\rangle|0_{\log\tau}\rangle|0\rangle|\chi_{a,b}\rangle|0\rangle,$$

the gate produces the output superposition

$$|\psi[a][0]\rangle\ldots|\psi[a][\log(\tau)-1]\rangle|\psi[b][0]\rangle\ldots|\psi[a][\log(\tau)-1]\rangle|0_{\log\tau}\rangle|\psi[a]=\psi[b]\rangle|\chi_{a,b}\rangle|\chi_{a,b}\wedge(\psi[a]=\psi[b])\rangle.$$

Gate $U_{=\lambda}$ exploits the auxiliary gates $U_=$ to compare $\psi[a]$ and $\psi[b]$, and a Toffoli gate with two inputs and outputs to verify if edges e_a and e_b cross and have the same color. By Lemma 1, gate $U_{=\lambda}$ has $O(\log\tau)$ circuit complexity, depth, and width.

The gate U_β works as follows. Consider that if two variables $\psi[a]$ and $\psi[b]$ are compared to determine whether e_a and e_b have been assigned the same color, none of these variables can be compared with another variable at the same time. Therefore, we partition the pairs of such variables using Corollary 1.1 (with $k=2$ and $|X|=m$) into $r \in O(m)$ classes $S_1,\ldots,S_r$ each containing at

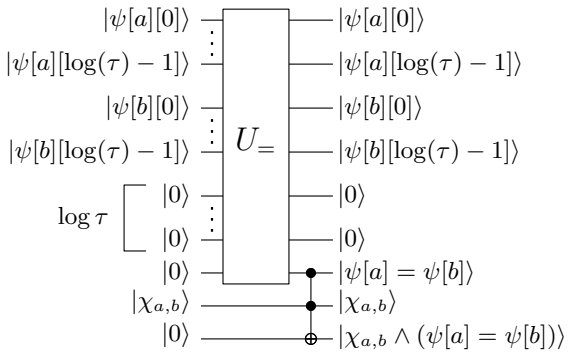


Figure 25: Gate $U_{=\lambda}$.

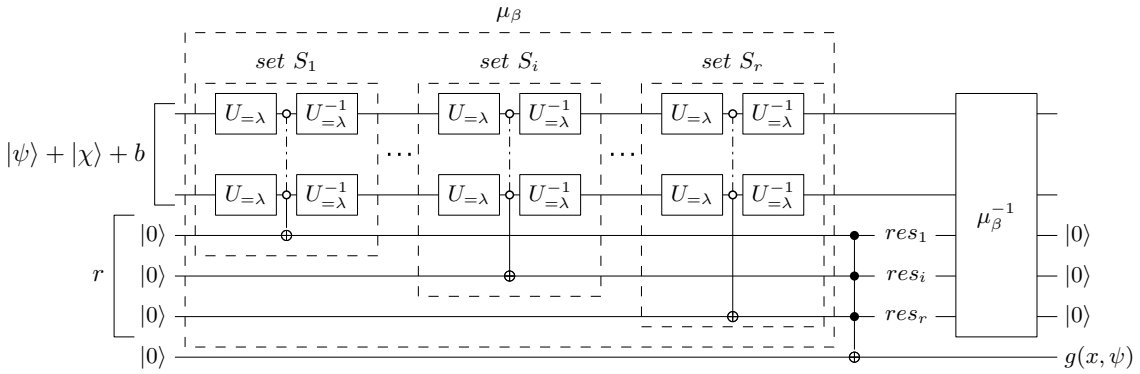


Figure 26: Gate U_β.

most $\frac{m}{2}$ disjoint pairs. For $i=1,\ldots,r$, the gate U_β executes in parallel a gate $U_{=\lambda}$, for each pair of variables $\{\psi[a],\psi[b]\}\in S_i$ (together with their corresponding qubit $\chi_{a,b}$), in order to output the qubit $|\chi_{a,b}\wedge(\psi[a]=\psi[b])\rangle$. All the last output qubits of the $U_{=\lambda}$ gates for S_i enter a Toffoli gate that outputs a qubit $|res_i\rangle$ such that $res_i=1$ if and only if all of them are equal to $|0\rangle$, i.e., it does not exist two crossing edges with the same color (among the pairs in S_i). In order to allow the reuse of the ancilla qubits, except for the qubit $|res_i\rangle$, gate U_β executes in parallel a gate $U_{=\lambda}^{-1}$ for each pair in S_i. All the qubits $|res_i\rangle$ enter a Toffoli gate that outputs a qubit $|g(x,\psi)\rangle$ such that $g(x,\psi)=1$ if and only if all of them are equal to $|1\rangle$, i.e., there exist no two edges of G with the same color that cross in $D(x,\psi)$. Gate U_β has circuit complexity $O(m^2\log\tau)$, depth $O(m\log\tau)$, and width $O(m)$.

FINAL CHECK. The purpose of gate U_{fc} is to check whether the current basis state is the encoding of an admissible solution, i.e., whether $D(x,\psi)$ is a book embedding of G on τ pages. Refer to Fig. 27. When provided with the input superposition $|f(\phi)\rangle|g(x,\psi)\rangle|-\rangle$, the gate U_{fc} produces the output superposition $|f(\phi)\rangle|g(x,\psi)\rangle(-1)^{g(x,\psi)f(\phi)}|-\rangle$. Gate U_{fc} exploits a Toffoli gate with three inputs and outputs. The control qubits are $|f(\phi)\rangle$ and $|g(x,\psi)\rangle$, and the target qubit is $|-\rangle$. When at least one of $f(\phi)$ and $g(x,\psi)$ are equal to 0, the target qubit leaves unchanged. On the other hand, when $f(\phi)=g(x,\psi)=1$, the target qubit is transformed into the qubit $-|-\rangle$. Gate

U_{fc} has $O(1)$ circuit complexity, depth, and width.

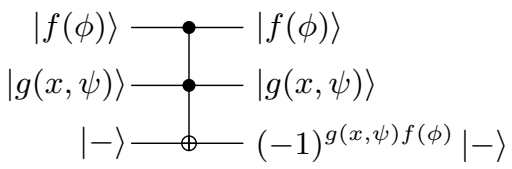


Figure 27: Gate U_{fc}.

The inverse circuits. The purpose of circuits U_{β}^{-1}, and $U_{\chi_p}^{-1}$ is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 12, observe that the gates U_{χ_p} and U_{β} verify all the necessary conditions for which $D(x,\psi)$ is a book embedding of G with τ pages, under the assumption that $\Pi(x)$ is not degenerate. Therefore, the sign of the output superposition of gate BT, which is determined by the expression $(-1)^{g(x,\psi)f(\phi)}$, is positive when either $\Pi(x)$ is degenerate or $\Pi(x)$ is not degenerate and the book layout $D(x,\psi)$ is not a book embedding of G with τ pages, and it is negative only if $\Pi(x)$ is not degenerate and $D(x,\psi)$ is a book embedding of G with τ pages. The bound on the circuit complexity descends from the circuit complexity of gate U_{χ_p}, the bound on the depth descends from the depth of gate U_{χ_p}, and the bound on the width descends from the width of gate U_{β}. □

5.7 Problem BS

We call BS the Solution Detector circuit for problem BS.

Lemma 13. *There exists a gate BS that, provided with the input superposition* $|f(\phi)\rangle\,|f(\theta)\rangle\,|x\rangle\,|e\rangle\,|0_h\rangle\,|-\rangle$, *where* $h \in O(m^2)$, *produces the output superposition* $(-1)^{g(x)f(\phi)f(\theta)}\,|f(\phi)\rangle\,|f(\theta)\rangle\,|x\rangle\,|e\rangle\,|0_h\rangle\,|-\rangle$, *such that, if* $\Pi(x)$ *and* $N(\theta)$ *are not degenerate, then* $g(x) = 1$ *if and only if the* 1*-page layout of* G' *determined by* $\Pi(x)$ *is a* 1*-page book embedding, where* $G' = (V, E \setminus K(\theta))$. *Gate BS has* $O(n^8)$ *circuit complexity,* $O(n^6)$ *depth, and* $O(m)$ *width.*

Proof of Lemma 13. Gate BS uses three gates: OP-cross finder U_{χ_p}, Skewness cross tester U_{sk}, and Final check U_{fc}, followed by the inverse gates U_{sk}^{-1} and $U_{\chi_p}^{-1}$. Refer to Fig. 28.

OP-cross finder. For the definition of gate U_{χ_p}, refer to the proof of Lemma 11. Recall that the purpose of U_{χ_p} is to compute the crossings in the 1-page layout of G defined by $\Pi(x)$, determined by the vertex order corresponding to x. Also Recall that when provided with the input superposition $|x\rangle\,|0_k\rangle$, the gate U_{χ_p} produces the output superposition $|x\rangle\,|\chi\rangle$.

Skewness cross tester. For the definition of U_{sk}, refer to the proof of Lemma 10. Consider the subgraph G' of G obtained by removing from G all the edges in $K(\theta)$. The purpose of gate U_{sk} is to determine which of the crossings stored in $|\chi\rangle$ involve pairs of edges that are both absent from $|e\rangle$. In fact, all the edges not in $|e\rangle$ form the edge set of G'. Therefore, U_{sk} verifies whether the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding. When provided with the input

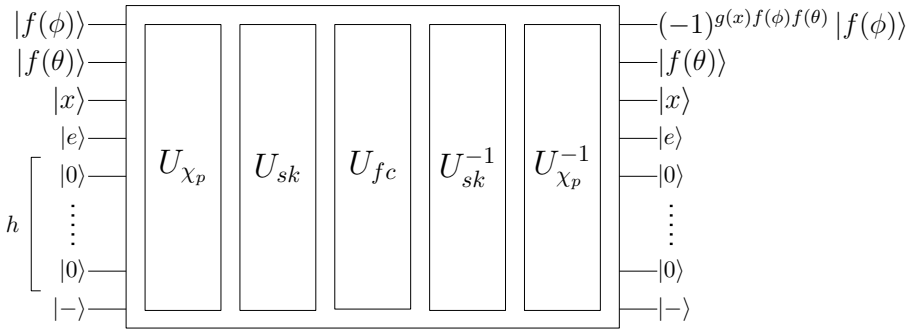


Figure 28: Oracle BS.

superposition $|\chi\rangle|e\rangle\left|0^{\frac{m}{2}+p}\right\rangle|0\rangle$, where $p \in O(m)$, the gate U_{sk} produces the output superposition $|\chi\rangle|e\rangle\left|0^{\frac{m}{2}+p}\right\rangle|g(x)\rangle$, such that, if $\Pi(x)$ and $N(\theta)$ are not degenerate, then $g(x) = 1$ if and only the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding.

Final check. For the definition of U_{fc}, refer to the proof of Lemma 10. The purpose of gate U_{fc} is to check whether the current basis state is the encoding of an admissible solution, i.e., whether the 1-page layout of G determined by $\Pi(x)$ and the set of indices $N(\theta)$ are both not degenerate and the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding. See Fig. 21. When provided with the input superposition $|f(\phi)\rangle|f(\theta)\rangle|g(x)\rangle|-\rangle$, the gate U_{fc} produces the outputs superposition $|f(\phi)\rangle|f(\theta)\rangle|g(x)\rangle(-1)^{g(x)f(\phi)f(\theta)}|-\rangle$.

The inverse circuits. The purpose of circuits U_{sk}^{-1}, and $U_{\chi_p}^{-1}$ is to restore the h ancilla qubit to $|0\rangle$ so that they can be used in the subsequent steps of Grover's approach.

Correctness and complexity. For the correctness of Lemma 13, observe that the gates U_{χ_p}, U_{sk}, and U_{fc} verify all the necessary conditions for which the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding, under the assumption that $\Pi(x)$ and $N(\theta)$ are not degenerate. Therefore, the sign of the output superposition of gate BS, which is determined by the expression $(-1)^{g(x)f(\phi)f(\theta)}$, is defined as follows. It is positive when either $\Pi(x)$ or $N(\theta)$ are degenerate or $\Pi(x)$ and $N(\theta)$ are not degenerate and the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding. It is negative when $\Pi(x)$ and $N(\theta)$ are not degenerate and the 1-page layout of G' determined by $\Pi(x)$ is a 1-page book embedding. The bounds on the circuit complexity and depth of gate BS descend from those of U_{χ_p}, whereas the bound on the width of BS descends from U_{sk}.□

6 Exploiting Quantum Annealing for Graph Drawing

In this section, we explore the 2-level problems and the book layout problems, that we have addressed so far from the quantum circuit model perspective, in the context of the quantum annealing model of computation. We pragmatically concentrate on the D-Wave platform, which offers quantum annealing services based on large-scale quantum annealing solver. To utilize the hybrid facility of D-Wave for solving an optimization problem, there are essentially two ways: Either the problem is provided with its QUBO formulation or it is provided with a CBO formulation

with constraints that are at most quadratic. Also, given a CBO formulation, it is quite simple to construct a QUBO formulation. Hence, in Sects. 6.1 and 6.2, we first provide CBO formulations for the problems introduced in the previous section. Second, we overview (Sect. 6.3) a standard method for transforming a CBO formulation into a QUBO formulation. Third, in Sect. 6.4, we discuss a detailed experiment conducted on the quantum annealing services provided by D-Wave, specifically focusing on TLCM, which has extensive experimental literature compared to other problems considered in this paper. These experiments evaluate the efficiency of D-Wave with respect to well-known classical approaches to the TLCM problem.

6.1 CBO Formulations for Two-Level Problems

Let $G = (U, V, E)$ be a bipartite graph. We denote by u_i, for $i = 1, \ldots, |U|$, and v_j, for $j = |U| + 1, \ldots, |U| + |V|$, the vertices in U and V, respectively. We start by describing the variables and the constraints needed to model the vertex ordering in a 2-level drawing, which are common to the formulations of TLCM, TLKP, TLQP, and TLS.

Ordering variables. To model the order of the vertices in U and V in a 2-level drawing Γ of G, we use $|U| \cdot (|U| - 1)$ binary variables $u_{i,j}$ for each ordered pair of vertices $u_i, u_j \in U$ and $|V| \cdot (|V| - 1)$ binary variables $v_{i,j}$ for each ordered pair of vertices $v_i, v_j \in V$. The variable $x_{i,j}$ is equal to 1 if and only if x_i precedes x_j in Γ, with $x \in \{u, v\}$.

Ordering constraints. We define the following constraints. As in [28], to model the fact that an assignment of values to the variables $x_{i,j}$, with $x \in \{u, v\}$, correctly models a linear ordering of the vertices in U and in V, we exploit two types of constraints:

Consistency: For each ordered pair of vertices $u_i, u_j \in U$, we have the constraint **(CU)** $u_{i,j} + u_{j,i} = 1$. Similarly, for each ordered pair of vertices $v_i, v_j \in V$, we have the constraint **(CV)** $v_{i,j} + v_{j,i} = 1$. Clearly, there exist $O(|U|^2)$ and $O(|V|^2)$ constraints of type **(CU)** and **(CV)**, respectively.

Transitivity: For each ordered triple of vertices $u_i, u_j, u_k \in U$, we have the constraints **(TU)** $u_{i,j} + u_{j,k} - u_{i,k} \geq 0$ and $u_{i,j} + u_{j,k} - u_{i,k} \leq 1$. The constraint **(TV)** for each ordered triple of vertices of V is defined analogously. Clearly, there exist $O(|U|^3)$ and $O(|V|^3)$ constraints of type **(TU)** and **(TV)**, respectively. Constraints **(TU)** and **(TV)** are linear. We also consider alternative quadratic constraints for transitivity: for each ordered triple of vertices $u_i, u_j, u_k \in U$, we have the constraints **(TQU)** $1 - (u_{i,j} \cdot u_{j,k}) + u_{i,k} \geq 1$. The constraints **(TQV)** for each ordered triple of vertices of V are defined analogously. Clearly, the number of **(TQU)** and **(TQV)** constraints is half the number of **(TU)** and **(TV)** constraints. It is possible to further reduce the number of transitivity constraints by substituting constraints **(TQU)** with the constraints **(U-TQU)** described below. To define constraint **(U-TQU)** observe that the transitivity constraints between variables u_{ij}, u_{ik}, and u_{jk} can be modeled by means of the Boolean formula $(u_{ij} \wedge \neg u_{jk}) \vee (\neg u_{ij} \wedge \neg u_{ik}) \vee (u_{ik} \wedge u_{jk})$; refer to Table 2. For each such Boolean formula, we introduce a constraint **(U-TQU)** given by $(u_{ij} \cdot (1 - u_{jk})) + ((1 - u_{ij}) \cdot (1 - u_{ik})) + (u_{ik} \cdot u_{jk}) = 1$. Observe that this constraint is satisfied if and only if the corresponding Boolean formula is true. The constraints **(U-TQV)** for each ordered triple of vertices of V are defined analogously. Clearly, the number of **(U-TQU)** and **(U-TQV)** constraints is half the number of **(TQU)** and **(TQV)** constraints.

Next, we provide specific variables and constraints that allow us to correctly model the problems TLCM, TLKP, TLQP, and TLS. To this aim, for each pair of independent edges $e_a = (u_i, v_k)$ and

u_{ij}	u_{ik}	u_{jk}	$(u_{ij} \wedge \neg u_{jk}) \vee (\neg u_{ij} \wedge \neg u_{ik}) \vee (u_{ik} \wedge u_{jk})$	$Transitivity$
0	0	0	1	$Valid$
0	0	1	1	$Valid$
0	1	0	0	$Not-Valid$
0	1	1	1	$Valid$
1	0	0	1	$Valid$
1	0	1	0	$Not-Valid$
1	1	0	1	$Valid$
1	1	1	1	$Valid$

Table 2: Transitivity truth table.

$e_b = (u_j, v_\ell)$, we define the expression $\chi_{a,b} = u_{i,j} \cdot v_{\ell,k} + u_{j,i} \cdot v_{k,\ell}$, which is equal to 1 if and only if e_a and e_b cross. For each edge $e \in E$, we denote by $I(e)$ the set of edges in E that do not share an endpoint with e.

Two-level Crossing Minimization (TLCM). We consider the minimization version of the problem. In order to minimize the total number of crossings in the sought 2-level drawing of G, we define the objective function (**OF**)

$$\min \sum_{e_a \in E} \sum_{e_b \in I(e_a)} \chi_{a,b}.$$

Two-level k-planarity (TLKP). We show how to model the fact that on each edge at most k crossings are allowed. Hence, for each edge e_a, we have the constraint **(KP)**
$\sum_{e_b \in I(e_a)} \chi_{a,b} \leq k$.
Clearly, over all the edges of G, there are $|E|$ constraints of type **(KP)**.

Two-level Quasi Planarity (TLQP). We show how to model the fact that no three edges are allowed to pairwise cross. For each ordered triple (e_a, e_b, e_c) of edges of E such that $e_b \in I(e_a)$ and $e_c \in I(e_a) \cap I(e_b)$, we have the constraint **(QP)**

$$\chi_{a,b} + \chi_{b,c} + \chi_{a,c} < 3.$$

Clearly, over all the edges of G, there are $O(|E|^3)$ constraints of type **(QP)**.

Two-level Skewness (TLS). To model the membership of the edges to a subset S such that $|S| \leq \sigma$, whose removal from G yields a forest of caterpillars, we use $|E|$ binary variables $s_{i,j}$ for each edge (u_i, v_j). The variable $s_{i,j}$ is equal to 1 if and only if (u_i, v_j) belongs to S. First, to enforce that $|S| \leq \sigma$, we use the constraint **(CS)**

$$\sum_{(u_i, v_j) \in E} s_{i,j} \leq \sigma.$$

Second, we show how to model the fact that no two edges in $E \setminus S$ are allowed to cross. For each edge $e_a = (u_i, v_j) \in E$ and for each edge $e_b = (u_\ell, v_k) \in I(e_a)$, we have the constraint **(S)**

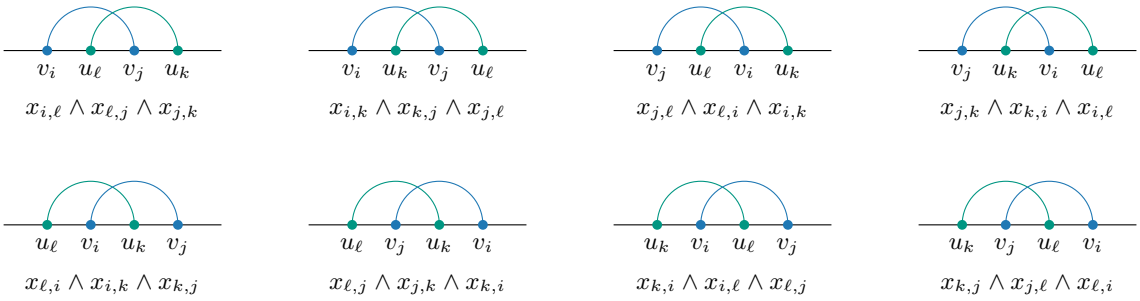


Figure 29: (top) The four possible crossing configurations of the edges $e_a = (v_i, v_j)$ and $e_b = (v_\ell, v_k)$, in which an end-vertex of e_a precedes both endpoints of e_b. (bottom) The four possible crossing configurations of the edges e_a and e_b, in which an end-vertex of e_b precedes both endpoints of e_a.

$$\chi_{a,b} - s_{i,j} - s_{\ell,k} < 1.$$

Over all the edges of G there are $O(|E|^2)$ constraints of type **(S)**.

6.2 CBO Formulations for Book-layout Problems

Let $G = (V, E)$ be a graph. We denote by v_i, for $i = 1, \ldots, |V|$, the vertices in V. As in Sect. 6.1, we use the variable $x_{i,j}$ to encode the precedence between the ordered pair of vertices $v_i, v_j \in V$. Moreover, in order for an assignment of values in $\mathbb{B}$ to such variables to correctly model a linear ordering of V, we use the *consistency* and *transitivity* constraints described in Sect. 6.1.

Next, we provide the specific variables and constraints that allow us to correctly model the problems OPCM, BT, and BS. Two edges are *independent* if they do not share an end-vertex. To this aim, for each ordered pair of independent edges $e_a = (v_i, v_j)$ and $e_b = (v_\ell, v_k)$, we define the expression $\chi_{a,b} = x_{i,\ell} \cdot x_{\ell,j} \cdot x_{j,k} + x_{i,k} \cdot x_{k,j} \cdot x_{j,\ell} + x_{j,\ell} \cdot x_{\ell,i} \cdot x_{i,k} + x_{j,k} \cdot x_{k,i} \cdot x_{i,\ell}$, which is equal to 1 if and only if e_a and e_b cross and (exactly) one of the endpoints of e_a precedes both the endpoints of e_b; refer to Fig. 29(top). More specifically, let $x_{\alpha,\beta} \cdot x_{\beta,\gamma} \cdot x_{\gamma,\delta}$ be any of the four terms that define $\chi_{a,b}$. We have that such a term evaluates to 1 if and only if the vertices v_α, v_β, v_γ, and v_δ appear in this left-to-right order along the spine.

One-Page Crossing Minimization (OPCM). We consider the minimization version of the problem. In order to minimize the total number of crossings in the sought 1-page layout of G, we define the objective function **(OF)**

$$\min \sum_{e_a \in E} \sum_{e_b \in I(e_a)} \chi_{a,b}.$$

Book Thickness (BT). To model the membership of the edges to one of the τ pages, we use $\tau|E|$ binary variables $e_{i,j,c}$, for each edge (v_i, v_j) and for each page $c \in [\tau]$. The variable $e_{i,j,c}$ is equal to 1 if and only if (v_i, v_j) is assigned to page c. First, in order to enforce that each edge belongs exactly to one page, for each edge $(v_i, v_j) \in E$, we use the constraint **(BC)**

$$\sum_{c\in[\tau]} e_{i,j,c} = 1.$$

Second, we show how to model the fact that no two edges assigned to the same page are allowed to cross. For each edge $e_a = (v_i, v_j) \in E$, for each edge $e_b = (v_l, v_k) \in I(e_a)$, and for each page $c \in [\tau]$, we have the constraint **(CC)**

$$\chi_{a,b} + e_{i,j,c} + e_{l,k,c} < 3.$$

Book Skewness (BS). For this problem, we adopt the same constraints **(CS)** as for the TLS problem. Moreover, we adopt the constraints **(S)** as for the TLS problem.

6.3 From CBO to QUBO

The formulations presented in Sects. 6.1 and 6.2 contain quadratic and even cubic constraints. Such constraints, however, only involve binary variables. Hence, they can be easily linearized, by means of standard operations research techniques, to be exploited to define a QUBO formulation suitable for quantum annealing.

Specifically, let $\mu = \Pi_{i=1}^{k} x_i$ be a (not necessarily quadratic) monomial of total degree k, such that each variable x_i is a binary variable. We obtain an equivalent constraint by replacing each occurrence of μ in all the constraints of our formulation with a new binary variable z_μ and by adding the following $k+1$ constraints:

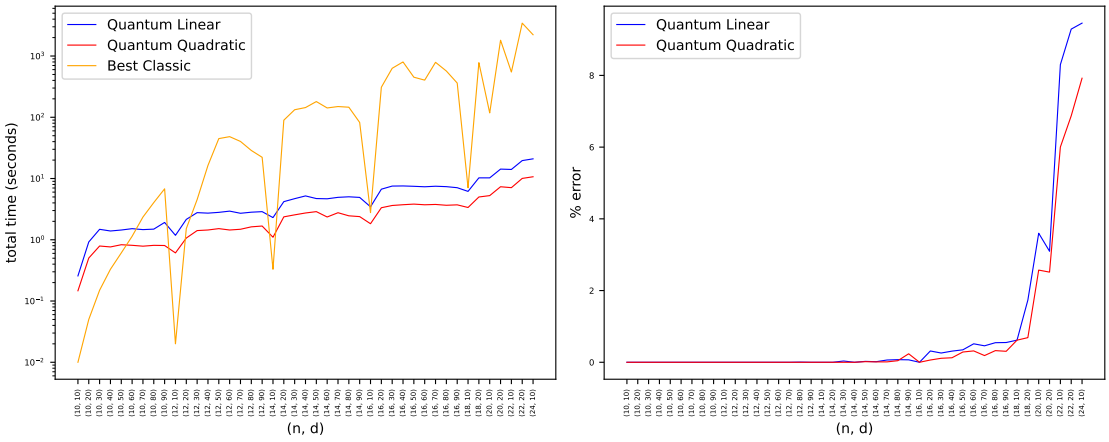
$$\begin{cases} z_\mu \leq x_i & i = 1, \ldots, k \\ z_\mu \geq 1 - k + \sum_{i=1}^{k} x_i \end{cases}$$

In our formulation for the 2-level problems, the maximum degree of all monomials is 2 and these monomials arise from the expressions $\chi_{a,b}$, for each pair of independent edges $e_a = (u_i, v_i)$ and $e_b = (u_\ell, v_k)$. Thus, there exist at most $|E(G)|^2$ distinct degree-2 monomials in these formulations. Therefore, by applying the replacement described above, we introduce at most $|E(G)|^2$ new variables and $3|E(G)|^2$ new constraints. Similarly, in our formulation for the book layout problems, the maximum degree of all monomials is 3 and these monomials arise from the expressions $\chi_{a,b}$, for each pair of independent edges $e_a = (u_i, v_i)$ and $e_b = (u_\ell, v_k)$. Thus, there exist at most $|E(G)|^2$ distinct degree-3 monomials in these formulations. Therefore, by applying the replacement described above, we introduce at most $|E(G)|^2$ new variables and $4|E(G)|^2$ new constraints.

Once the constraints are linearized, a QUBO formulation is obtained by inserting all constraints in the objective function. Note that if the problem is not an optimization problem an objective function is anyway created such that its optimum value is equal to zero. Constraints are first transformed so that their right member is equal to zero. For inequalities an extra variable is inserted so to transform the inequality into an equality. Then left member is squared and the result is inserted into the objective function.

6.4 D-Wave Experimentation

We performed our experiments on TLCM using the hybrid solver of D-Wave, which suitably mixes quantum computations with classic tabu-search and simulated annealing heuristics. The obtained results are not guaranteed to be optimal.



(a) Time comparison.

(b) Percentage difference with respect to the optimum number of crossings.

Figure 30: D-Wave experimentation results: (a) comparison of times and (b) quality of the solutions.

The D-Wave hybrid solver receives as input either a CBO or a QUBO formulation of a problem. We used it with a linear CBO formulation (with **(TU)** and **(TV)** constraints) and with a quadratic CBO formulation (with **(TQU)** and **(TQV)** constraints). Roughly, the D-Wave hybrid solver works as follows. First, it decomposes the problem into parts that fit the quantum processor. The decomposition aims at selecting subsets of variables, and hence sub-problems, maximally contributing to the problem energy. Second, it solves such sub-problems with the quantum processor. Third, it injects the obtained results into the original overall problem that is solved with traditional heuristics. These steps can be repeated several times, since an intermediate solution can re-determine the set of variables that contribute the most to the energy of the problem. An interesting description of the behaviour and of the limitations of D-Wave is presented in [27], although it refers to the quantum processor called Chimera[5] that has been replaced by the new processors called Pegasus[5] and Zephyr[5].

We compare our results with the figures proposed in [11]. Namely, the authors compare three exact algorithms for TLCM: LIN, which is the standard linearization approach; JM, which is the algorithm in [28]; and SDP, which is the branch-and-bound in [36]. Their experiments were carried out on an Intel Xeon processor with 2.33 GHZ.

Fig. 30 and Table 3 illustrate the results of the experimentation we conducted on the D-Wave platform. We used the code and seed provided by the authors of [11] to generate the same set of graphs as in [11]. Following their approach, for each pair (n, d), where n is the number of vertices on each layer and d is the density of the graph, we performed 10 experiments on the first 10 generated distinct graph instances. As in [11], the number of the edges of the graphs generated for a pair (n, d) is $\lfloor dn^2/100 \rfloor$. Fig. 30a reports for each pair n, d, consisting of number of vertices n and density d: (1) the time spent to find the optimum by the fastest classical algorithm between LIN, JM, and SDP, (2) the time spent by our implementation using linear constraints, and (3) the time spent by our implementation using quadratic constraints. All the values are the average computed on the cited 10 instances. The columns of Table 3 report what follows. *Best [11] Time*: the best performance

[5]D-Wave QPU architecture: Chimera, Pegasus, and Zephyr topology.

among the exact algorithms for TLCM known so far (time measured by [11]). *Best [11] Approach*: the fastest among LIN, JM, and SDP. The double slash indicates that an optimal solution has not been found. *D-Wave Time (Linear)*: time of our experiments, using the linear formulation. *Constrains (Linear)*: number of constraints of our linear formulation. *Crossings (Linear)*: number of crossings obtained with our linear formulation. *D-Wave Time (Quadratic)*: time of our experiments, using the quadratic formulation. *Constrains (Quadratic)*: number of constraints of our quadratic formulation. *Crossings (Quadratic)*: number of crossings obtained with our quadratic formulation.

Fig. 30b shows that the number of crossings obtained by the quadratic implementation was the optimal one for all graphs with up to 14 vertices per layer and up to 40% of density. Also, the number of crossings obtained by the quadratic (resp., linear) implementation, in all sets of instances deviates of at most 8% (resp., 9%) from the optimal one. Futher, for 60% of the sets both the linear and the quadratic implementations achieve the optimum.

The comparison of the time employed by D-Wave (linear and quadratic) with the one of the best exact methods is quite promising, even if the times in the *Best Current Time* column are the results of a computation performed on a non-up-to-date classical hardware, and indicate that D-Wave can be used to efficiently tackle instances of TLCM. The valleys in 30a show that classical algorithms perform well for sparse instances, whereas quantum algorithms appear to perform better on dense instances. The comparison between linear and quadratic CBO formulations indicates that the quadratic formulation is more efficient, since it generates fewer constraints. Their behaviour in terms of number of crossings are quite similar. The time we report is the overall time elapsed between the remote call from our client and the reception of the result. The actual time spent on the quantum processor is always between 0.016 and 0.032 sec.

7 Conclusions and Open Problems

We initiate the study of quantum algorithms in the Graph Drawing research area, providing a framework that allows us to tackle several classic problems within the 2-level and book layout drawing standards. Our framework, equipped with several quantum circuits of potential interest to the community, builds upon Grover's quantum search approach. It empowers us to achieve, ignoring polynomial terms, a quadratic speedup compared to the best known classical exact algorithms for all the problems under consideration. In addition, we conducted experiments using the D-Wave quantum annealing platform for the Two-Level Crossing Minimization problem. Our experiments demonstrated that the platform is highly suitable for addressing graph drawing problems and showcased significant efficiency when compared to the top approaches available for solving such problems [11, 28]. The encounter between Graph Drawing and Quantum Computing is still in its nascent stage, offering a vast array of new and promising problems (see, for instance, the recent developments in [12]). Virtually, all graph drawing problems can be explored through the lenses of quantum computation, utilizing both the quantum circuit model and, more pragmatically, quantum annealing platforms.

Acknowledgments. We thank the anonymous reviewers of this paper for their valuable comments that helped us improve the quality of our manuscript.

Table 3: TLCM: D-Wave vs other approaches. Bipartite graphs with increasing number of vertices per layer and density. All times are in seconds.

n	d (%)	Best [11] Approach	Best [11] Time	D-Wave Time (Linear)	D-Wave Time (Quadratic)	Crossings (Exact)	Crossings (Linear)	Crossings (Quadratic)	Constraints (Linear)	Constraints (Quadratic)
10	10	LIN	0,01	0,26	0,15	0,22	0,22	0,22	509	269
10	20	JM	0,05	0,92	0,50	12,22	12,22	12,22	1926	996
10	30	JM	0,15	1,48	0,79	61,33	61,33	61,33	2969	1529
10	40	JM	0,33	1,39	0,76	151,33	151,33	151,33	2970	1530
10	50	JM	0,61	1,44	0,83	281,88	281,88	281,88	2970	1530
10	60	JM	1,14	1,52	0,81	458,55	458,55	458,55	2970	1530
10	70	JM	2,35	1,47	0,79	706,55	706,55	706,55	2970	1530
10	80	JM	4,05	1,49	0,81	1007,11	1007,11	1007,11	2970	1530
10	90	SDP	6,79	1,92	0,81	1408	1408	1408	2970	1530
12	10	LIN	0,02	1,18	0,61	0,67	0,67	0,67	2293	1183
12	20	JM	1,52	2,15	1,07	34,22	34,22	34,22	4110	2110
12	30	JM	4,53	2,77	1,41	139,11	139,11	139,11	5263	2696
12	40	JM	16,36	2,72	1,44	339,89	339,89	339,89	5337	2734
12	50	SDP	44,84	2,80	1,52	664,56	664,56	664,56	5412	2772
12	60	SDP	48,26	2,94	1,44	1040	1040	1040	5412	2772
12	70	SDP	40,31	2,71	1,48	1535	1535	1535	5412	2772
12	80	SDP	28,71	2,82	1,63	2228,55	2228,67	2228,56	5412	2772
12	90	SDP	22,21	2,88	1,68	3023,67	3023,67	3023,67	5412	2772
14	10	LIN	0,33	2,30	1,10	2,67	2,67	2,67	3912	2008
14	20	SDP	89,61	4,19	2,37	89,33	89,33	89,33	7701	3933
14	30	SDP	132,72	4,69	2,55	316,78	316,89	316,78	8512	4344
14	40	SDP	144,03	5,20	2,74	716	716	716	8918	4550
14	50	SDP	180,49	4,71	2,88	1315,78	1316,11	1316	8918	4550
14	60	SDP	141,93	4,67	2,35	2052,67	2053	2052,89	8918	4550
14	70	SDP	149,68	4,93	2,76	3015,89	3017,78	3016,22	8918	4550
14	80	SDP	145,97	5,03	2,46	4255,78	4258,89	4257,67	8918	4550
14	90	SDP	81,27	4,92	2,39	5861,33	5865,33	5875,22	8918	4550
16	10	LIN	2,77	3,45	1,83	11,56	11,56	11,56	6672	3410
16	20	SDP	309,31	6,70	3,34	176,22	176,78	176,33	12423	6324
16	30	SDP	630,31	7,53	3,63	603,22	604,78	603,89	13397	6817
16	40	SDP	800,87	7,56	3,73	1322,78	1326,89	1324,44	13680	6960
16	50	SDP	451,09	7,46	3,82	2368,67	2376,89	2375,44	13680	6960
16	60	SDP	403,82	7,33	3,73	3750,78	3770,11	3762,67	13680	6960
16	70	SDP	789,62	7,48	3,77	5476,56	5501,67	5486,78	13680	6960
16	80	SDP	568,55	7,37	3,66	7550,56	7591,78	7575,11	13680	6960
16	90	SDP	362,29	7,08	3,72	10279,45	10336,11	10310,89	13680	6960
18	10	LIN	7,06	6,16	3,36	18	18,11	18,11	12154	6187
18	20	SDP	778,86	10,25	4,99	307,45	312,78	309,56	18424	9358
20	10	LIN	117,72	10,25	5,26	43,22	44,78	44,33	19357	9828
20	20	SDP	1813,87	14,25	7,34	534,78	551,33	548,22	26589	13479
22	10	LIN	546,71	14,06	7,09	87	94,22	92,22	25942	13152
22	20	SDP	3443,81	19,62	10,04	900,56	984,22	962,44	35217	17830
24	10	LIN	2225,82	20,96	10,69	137,44	150,44	148,33	37757	19110
24	20	//	//	27,70	14,17	//	1794,22	1835,11	48788	24669

References

[1] S. Aaronson. Introduction to quantum information science lecture notes, April 2019. URL: https://www.scottaaronson.com/qclec.pdf.

[2] P. Angelini, G. Da Lozzo, G. Di Battista, F. Frati, and M. Patrignani. 2-Level quasi-planarity or how caterpillars climb SPQR-trees. In D. Marx, editor, *SODA 2021*, pages 2779–2798. SIAM, 2021. doi:10.1137/1.9781611976465.165.

[3] P. Angelini, G. Da Lozzo, H. Förster, and T. Schneck. 2-Layer k-planar graphs - density, crossing lemma, relationships, and pathwidth. In D. Auber and P. Valtr, editors, *Graph Drawing and Network Visualization - 28th International Symposium, GD 2020, Vancouver, BC, Canada, September 16-18, 2020, Revised Selected Papers*, volume 12590 of *LNCS*, pages 403–419. Springer, 2020. doi:10.1007/978-3-030-68766-3_32.

[4] P. Angelini, G. Da Lozzo, H. Förster, and T. Schneck. 2-Layer k-Planar Graphs Density, Crossing Lemma, Relationships And Pathwidth. *The Computer Journal*, 04 2023. arXiv:https://academic.oup.com/comjnl/advance-article-pdf/doi/10.1093/comjnl/bxad038/50062043/bxad038.pdf, doi:10.1093/comjnl/bxad038.

[5] H. Babu. *Quantum Computing: A Pathway to Quantum Logic Design*. G - Reference,Information and Interdisciplinary Subjects Series. Institute of Physics Publishing, 2020. URL: https://books.google.it/books?id=KVRAygEACAAJ.

[6] M. J. Bannister and D. Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In C. A. Duncan and A. Symvonis, editors, *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*, volume 8871 of *Lecture Notes in Computer Science*, pages 210–221. Springer, 2014. doi:10.1007/978-3-662-45803-7_18.

[7] M. J. Bannister and D. Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. *J. Graph Algorithms Appl.*, 22(4):577–606, 2018. doi:10.7155/jgaa.00479.

[8] Z. Baranyai. The edge-coloring of complete hypergraphs I. *J. Comb. Theory B*, 26(3):276–294, 1979. doi:10.1016/0095-8956(79)90002-9.

[9] F. Bernhart and P. C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

[10] D. Bhattacharyya, D. BHATTACHARYYA, J. Guha, and I. of Physics (Great Britain). *Quantum Optics and Quantum Computation: An Introduction*. IOP series in advances in optics, photonics and optoelectronics. IOP Publishing, 2022. URL: https://books.google.it/books?id=ubnzzgEACAAJ.

[11] C. Buchheim, A. Wiegele, and L. Zheng. Exact algorithms for the quadratic linear ordering problem. *INFORMS J. Comput.*, 22(1):168–177, 2010. doi:10.1287/ijoc.1090.0318.

[12] S. Caroppo, G. Da Lozzo, and G. Di Battista. Quantum algorithms for one-sided crossing minimization. In S. Felsner and K. Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024, September 18-20, 2024, Vienna, Austria*, volume

320 of *LIPIcs*, pages 20:1–20:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.GD.2024.20, doi:10.4230/LIPICS.GD.2024.20.

[13] S. Caroppo, G. Da Lozzo, and G. Di Battista. Quantum Graph Drawing. In R. Uehara, K. Yamanaka, and H. 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.

[14] S. Caroppo, G. Da Lozzo, and G. Di Battista. Source code for the experimental study presented in the paper "Quantum Graph Drawing", May 2024. URL: https://github.com/s-caro/Quantum-TLCM.

[15] B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

[16] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[17] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

[18] V. Dujmovic, M. R. Fellows, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008. doi:10.1007/s00453-007-9151-1.

[19] P. Eades, B. D. McKay, and N. C. Wormald. On an edge crossing problem. In *9th Australian Comp. Science Conference, ACSC 1986, Proceedings*, pages 327–334, 1986.

[20] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.

[21] S. Fukuzawa, M. T. Goodrich, and S. Irani. Quantum Tutte embeddings. *CoRR*, abs/2307.08851, 2023. arXiv:2307.08851, doi:10.48550/arXiv.2307.08851.

[22] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983. arXiv:https://doi.org/10.1137/0604033, doi:10.1137/0604033.

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

[24] C. Gutwenger, K. Klein, and P. Mutzel. Planarity testing and optimal edge insertion with embedding constraints. In M. Kaufmann and D. Wagner, editors, *Graph Drawing, 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers*, volume 4372 of *Lecture Notes in Computer Science*, pages 126–137. Springer, 2006. doi:10.1007/978-3-540-70904-6_14.

[25] C. Gutwenger, K. Klein, and P. Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *J. Graph Algorithms Appl.*, 12(1):73–95, 2008. URL: https://doi.org/10.7155/jgaa.00160, doi:10.7155/JGAA.00160.

[26] A. W. Harrow. Quantum algorithms for systems of linear equations. In *Encyclopedia of Algorithms*, pages 1680–1683. 2016. doi:10.1007/978-1-4939-2864-4_771.

[27] M. Jünger, E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G. Rinaldi, and T. Stollenwerk. Performance of a quantum annealer for ising ground state computations on chimera graphs. *CoRR*, abs/1904.11965, 2019. arXiv:1904.11965.

[28] M. Jünger and P. Mutzel. 2-Layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997. doi:10.7155/jgaa.00001.

[29] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush. Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization. *Quantum*, 4:296, July 2020. doi:10.22331/q-2020-07-16-296.

[30] Y. Kobayashi, H. Maruta, Y. Nakae, and H. Tamaki. A linear edge kernel for two-layer crossing minimization. *Theor. Comput. Sci.*, 554:74–81, 2014. doi:10.1016/j.tcs.2014.06.009.

[31] Y. Kobayashi and H. Tamaki. A faster fixed parameter algorithm for two-layer crossing minimization. *Inf. Process. Lett.*, 116(9):547–549, 2016. doi:10.1016/j.ipl.2016.04.012.

[32] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the NP-completeness of a computer network layout problem. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS 1987)*, page 292–295, 1987.

[33] C. C. McGeoch. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Synthesis Lectures on Quantum Computing. Morgan & Claypool Publishers, 2014. doi:10.2200/S00585ED1V01Y201407QMC008.

[34] H. Nagamochi and N. Yamada. Counting edge crossings in a 2-layered drawing. *Inf. Process. Lett.*, 91(5):221–225, 2004. URL: https://doi.org/10.1016/j.ipl.2004.05.001, doi:10.1016/J.IPL.2004.05.001.

[35] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[36] F. Rendl, G. Rinaldi, and A. Wiegele. A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In M. Fischetti and D. P. Williamson, editors, *IPCO 2007*, volume 4513 of *LNCS*, pages 295–309. Springer, 2007. doi:10.1007/978-3-540-72792-7_23.

[37] E. Rieffel and W. Polak. *Quantum Computing: A Gentle Introduction*. The MIT Press, 1st edition, 2011.

[38] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrto. Book embeddings and crossing numbers. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *WG 1994*, volume 903 of *LNCS*, pages 256–268. Springer, 1994. doi:10.1007/3-540-59071-4_53.

[39] R. Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

[40] J. Tan and L. Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48(3):287–299, 2007. doi:10.1007/s00453-007-0118-z.

[41] A. Wigderson. The complexity of the Hamiltonian circuit problem for maximal planar graphs. Technical Report TR-298, Princeton University, 1982. arXiv:https://www.math.ias.edu/avi/node/820.

[42] M. Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, 1981. doi:10.1137/0210021.