

Article

A Hybrid Quantum–Classical Algorithm for k -SAT via Grover-Enhanced Local Search

Yingyu Zhang 

School of Artificial Intelligence, Hainan Normal University, Haikou 571158, China; 240120@hainnu.edu.cn

Abstract

The k -SAT problem is a canonical NP-complete problem for which no efficient algorithm is known. This paper proposes a hybrid quantum–classical search algorithm that employs Grover’s algorithm as the inner local search of a randomized outer loop. Each outer iteration measures a random assignment, prepares a problem-dependent superposition over the Hamming ball of radius r around it, and applies amplitude amplification to locate a satisfying assignment; it returns an assignment rather than a yes/no answer and cannot certify unsatisfiability. Because the ball center is measured, the outer loop remains a classical repetition over independently sampled balls and contributes the inverse success probability rather than its square root; balancing this against the inner quadratic speedup shifts the optimal radius from $n/(k+1)$ to $n/(1+\sqrt{k})$ and yields the running time $(2 - 2/(1+\sqrt{k}))^n$, up to polynomial factors— 1.268^n for $k=3$, below the best known classical 1.307^n . The contribution is not an asymptotic record—the best known quantum exponents for k -SAT, obtained by amplifying stronger classical solvers as black boxes, remain smaller—but a fully explicit, gate-level construction: the problem-dependent state over the Hamming ball and the k -SAT oracle are decomposed into elementary quantum gates, and the polynomial gate and qubit overheads are accounted for in full.

Keywords: k -SAT problem; Grover’s algorithm; amplitude amplification; quantum local search; Hamming ball; hybrid quantum–classical algorithm; gate-level construction

1. Introduction

The k -SAT problem ($k \geq 3$) is a canonical NP-complete problem [1], and no polynomial-time algorithm for it is known. Classical algorithms for k -SAT fall broadly into two categories: deterministic and probabilistic. Deterministic approaches typically rely on divide-and-conquer, exhaustive search, or backtracking, while probabilistic methods often employ local search heuristics. The worst-case time complexity of state-of-the-art algorithms in both categories is generally $O(\alpha^n)$, where n is the number of Boolean variables, and $\alpha > 1$ is a constant base. Since a smaller α implies lower computational complexity, much research [2–4] has focused on minimizing this exponential base.

On the probabilistic side, Paturi, Pudlák, and Zane [2] introduced the PPZ algorithm in 1997, achieving a randomized upper bound significantly below 2^n . Schöning [3] subsequently proposed a simple local search algorithm with expected running time $(2(1 - 1/k))^n$, yielding $O((4/3)^n)$, which improved upon the earlier PPZ bound of Ref. [2]. Hofmeister et al. [5] later refined Schöning’s approach, reducing the three-SAT complexity to $O(1.3302^n)$. Meanwhile, Paturi et al. [6] enhanced the original PPZ framework into the PPSZ algorithm, achieving $O(1.308^n)$ for unique three-SAT (the variant with exactly



Academic Editor: Ferdinando Di Martino

Received: 11 September 2026

Revised: 16 October 2026

Accepted: 20 October 2026

Published: 21 October 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

one satisfying assignment). Hertli [7] subsequently extended that analysis, proving that the PPSZ bound under the uniqueness assumption applied equally to general three-SAT. More recently, Hansen et al. [8] introduced a biased variant of PPSZ, further improving the unique three-SAT bound to $O(1.307^n)$. These algorithms use polynomial working space. Their running time is exponential, and their analyses give the smallest exponential bases known. The analyses do rest on structural assumptions, usually uniqueness of the solution or a bound on the number of satisfying assignments. These assumptions then have to be removed by a separate argument, as in the passage from unique three-SAT to general three-SAT of Ref. [7]. Over two decades of refinements, the base has improved by about two percent over that of far simpler methods: from $4/3 = 1.3333$ for the random walk of Ref. [3] to 1.307 for the biased PPSZ analysis of Ref. [8].

Parallel to these randomized advances, significant effort has been devoted to derandomization and deterministic algorithm design. Dantsin et al. [9] showed how to simulate Schöning's random walk deterministically, at the price of a mildly larger base: their Hamming-ball local search ran in time $O((2 - 2/(k + 1))^n)$, i.e., 1.5^n for $k = 3$. For three-SAT specifically, they introduced a tailored local search strategy that further reduced the bound to $O((1.481)^n)$. Moser and Scheder [10] later completed the full derandomization of Schöning's algorithm using limited independence, and they did so at the price of only an arbitrarily small ε in the base: their deterministic algorithm ran in time $O((2(k - 1)/k + \varepsilon)^n)$, i.e., $(4/3 + \varepsilon)^n$ for $k = 3$. Brueggemann and Kern [4] refined the local search strategy of Dantsin et al., reducing the three-SAT complexity to $O(1.473^n)$. Scheder [11] further improved that bound to $O(1.465^n)$. Makino, Tamaki and Yamamoto [12] then derandomized the faster randomized algorithm of Hofmeister, Schöning, Schuler and Watanabe [5] and obtained a deterministic $O(1.3303^n)$ algorithm for three-SAT. The smallest deterministic base for three-SAT is therefore the $O(1.3303^n)$ of Ref. [12]. For general k , it is the $(2(k - 1)/k + \varepsilon)^n$ of Ref. [10]. Moreover, a deterministic algorithm needs no randomness and no assumption on the instance, and its bounds are reproducible worst-case bounds. What it pays for this is the replacement of the random choices of the probabilistic algorithms by an exhaustive or pseudo-random enumeration. The base is therefore left larger than the randomized one (1.3303^n against 1.307^n for three-SAT). The deterministic base has come close to the randomized one without going below it, which is why we turn to a different mechanism: a quadratic speedup inside the ball, which changes the base rather than refining the classical search.

Quantum algorithms have demonstrated advantages over classical counterparts in several computational tasks, including integer factorization, unstructured search, and search on graphs [13–15], and they are studied as a tool for application areas ranging from machine learning to combinatorial optimization [16–18].

Hybrid quantum–classical schemes have become the standard way of exploiting quantum resources while devices remain small. In such a scheme, a classical outer loop calls a quantum subroutine. The optimization landscape of this family is reviewed in Ref. [19]. Such algorithms can be implemented on near-term hardware with few qubits and shallow circuits and are assessed by the quality of the solution that the classical optimizer reaches. In addition, a related line of work embeds quantum or quantum-inspired search into a classical metaheuristic. Nardin and D'Andreagiovanni [20], for instance, combine a quantum-inspired ant colony optimization with exact local searches that guarantee convergence to an optimum within a suitable solution neighborhood. The method works on structured combinatorial problems and is validated empirically. Grover-type search has also been adapted to combinatorial problems through dedicated state preparations and oracles. Examples are a simplified quantum counter for the exact cover problem [21] and Dicke-state search for vertex cover [22]. Hybrid schemes have further been proposed

for continuous optimization [23], text search [24] and reconfigurable quantum–classical computing frameworks [25]. These works address other aspects of hybrid schemes (implementability on near-term devices, empirical performance on structured problems, dedicated state preparations and oracles) rather than the quantity analyzed in this paper, namely, the exponent of the total cost of the hybrid with the classical repetitions included; each is cited as an example of one line of work in the family to which the present scheme belongs, and that is the use made of them here. These works show how a quantum subroutine can be inserted into a classical search while keeping the quantum resource small. Ref. [26] optimizes such a hybrid method as a whole, and it is the one taken up next.

Among the hybrid schemes for combinatorial optimization, the one closest in approach to the present work is Ref. [26], which also solves three-SAT with a hybrid quantum–classical algorithm. It asks whether a quantum computer of Q qubits can speed up a computation on $n \gg Q$ variables, a question motivated by the size of near-term devices, and shows that it can, by exhibiting a hybrid algorithm faster than its fully classical counterpart. Four points separate the two contributions. (i) *Model of the device, and the question asked.* Ref. [26] imposes a size restriction (the device is much smaller than the instance) and asks whether a speedup exists under it. Here, no size restriction is imposed (the device must hold the n assignment qubits together with polynomially many ancillas), and the question is quantitative: What exponent does an explicit construction achieve for every $k \geq 3$ (Ref. [26] treats three-SAT), and how must the classical outer loop and the quantum inner loop be balanced to obtain it? (ii) *What is designed.* Both works search a Hamming ball of radius $r = \rho n$ around a center, and both balance the number of balls, $2^{n(1-h(\rho))}$, against the cost of the search inside one ball. Ref. [26] writes out that balance as well and optimizes it for a classical ball search, obtaining for $k = 3$ the radius $\rho = 1/4$ of Ref. [9] and $\rho = 1/3$ of Ref. [3]. There, however, the radius that the quantum routine can handle is capped by the device ($r = f(c)n$ for $Q = cn$ qubits), so the optimum of the balance with a quantum inner search is not attainable in that model. Here, no such cap is imposed: because the center of each ball is measured, the outer loop stays classical and contributes the inverse success probability $2^{n(1-h(\rho))}$ rather than its square root, and re-balancing that factor against the $k^{\rho n/2}$ cost of the inner amplitude amplification moves the radius to its own optimum, $\rho^* = 1/(1 + \sqrt{k})$, which fixes the exponent $(2 - 2/(1 + \sqrt{k}))^n$ (Section 3.7). The radius is thus fixed by the optimization, not by the size of the device. (iii) *Where the quantum search is placed.* In Ref. [26] the quantum routine is inserted at the bottom of a classical recursion: the ball search recurses with a decreasing radius, and the device is called once the radius is small enough to fit into Q qubits, hence the effort that reference spends on compressing the branch register into $O(r \log(n/r) + r + \log n)$ qubits. Here, the whole ball of radius $r = \rho^* n$ is searched coherently within one run. The tree is in both cases the clause-driven one, selected at each round by the first unsatisfied clause. What the present work adds is the bound on the marked amplitude: a leaf of depth d carries amplitude $k^{-d/2}$, so the state of Equation (6) has amplitude at least $k^{-r/2}$ on the satisfying assignments, and reflecting about that state rather than about the uniform superposition is what fixes the polynomial overhead (Sections 3.5 and 3.7). (iv) *What is established.* Ref. [26] establishes that a polynomial speedup over Schöning’s algorithm survives for a device of any constant fraction $c = Q/n$ of the instance size; the radius is in that case fixed by the device, and the exponent is expressed relative to Schöning’s, as $\gamma_0 + \varepsilon - 0.21 f(c)$. By contrast, the present paper removes the size restriction and evaluates the same balance at its own optimum, which yields the exponent in closed form for every $k \geq 3$: it gives a gate-level decomposition of the state preparation and of the k -SAT oracle (Sections 3.3 and 3.4), counts the polynomial overhead in full (Section 3.7) and checks numerically the two bounds on which the exponent rests (Section 4). Stated positively, the two works share the idea

of placing amplitude amplification inside a Hamming-ball search; what the present work adds is the three points (ii) to (iv), namely, the radius fixed by the optimization rather than by the size of the device, together with the closed-form exponent $(2 - 2/(1 + \sqrt{k}))^n$ obtained at that radius for every $k \geq 3$ and the explicit construction with the two bounds verified numerically. The two works are complementary rather than competing: Ref. [26] asks what a device of a given size can still achieve, the present paper what the construction achieves once the size restriction is removed.

Motivated by these developments, we investigate the application of quantum computation to the k -SAT problem. Prior quantum approaches to k -SAT largely accelerate an existing classical solver while keeping its structure inside a black-box oracle. Grover's algorithm alone yields $O(2^{n/2}) = O(1.414^n)$ [14]. Combining Grover search with the then best classical three-SAT algorithm ($O(1.3302^n)$ [5]) lowers the exponent to $O(1.153^n)$ [27], and applying amplitude amplification [28] to the PPSZ algorithm [6–8] gives $O(1.143^n)$ for three-SAT. These are, to our knowledge, the strongest known exponents for worst-case three-SAT in the gate model. Inspired by the classical algorithm in [3], we propose a hybrid quantum–classical algorithm that employs Grover's algorithm as a local search subroutine for solving k -SAT. The proposed algorithm is a hybrid quantum–classical procedure: its outer loop classically samples local search spaces, while its inner loop performs quantum search. It proceeds in t outer iterations. In each iteration, a problem-dependent superposition state is constructed over a local search space—specifically, a Hamming ball of radius r centered at a randomly generated assignment—by exploiting the problem structure. Grover's algorithm is then applied to that state in its amplitude-amplification form—the number of solutions inside a ball is not known in advance, so the exponential-search variant is used—to search for a satisfying assignment. Because the outer loop is a classical repetition over independently sampled balls, the number of outer iterations is governed by the classical success probability, and the resulting running time is $(2 - 2/(1 + \sqrt{k}))^n$ (with the optimal radius $r = n/(1 + \sqrt{k})$), which improves upon the classical local-search bounds for k -SAT [3,9] and, at $k = 3$, upon the best known classical algorithm [8] (1.268^n versus 1.307^n ; see Section 3.9 for the general- k comparison). These gains are over classical solvers: black-box amplification of a stronger classical solver yields smaller quantum exponents [6,8,27,28].

The contributions of this work can be summarized as follows.

1. **An explicit, gate-level construction.** Both the problem-dependent superposition over the Hamming ball and the k -SAT oracle are decomposed into elementary gates, so that no black-box oracle is left in the complexity and the polynomial overhead is fixed at $O(nmk)$ gates.
2. **A hybrid framework with a re-optimized ball radius.** Because the center of each ball is measured, the outer loop is a classical repetition over independently sampled balls and contributes the inverse success probability rather than its square root; balancing this factor against the quadratic speedup of the inner search shifts the optimal radius from the classical $r = n/(k + 1)$ to $r = n/(1 + \sqrt{k})$, so that the classical repetition does not simply reproduce the classical Hamming-ball exponent.
3. **A complete accounting and a numerical verification.** The qubit and gate costs of the whole construction are counted, and the two bounds on which the exponent rests—the success probability of a ball and the marked amplitude of the constructed state—are checked numerically for $k = 3$ on small instances (Section 4).
4. **A precise statement of what the algorithm computes, and of how robust it is.** The procedure is a search algorithm: it returns a classically verifiable satisfying assignment and never certifies unsatisfiability (Section 3.6). The effect of state-preparation and gate errors on the success probability is quantified, and the assumptions needed for a reliable implementation are listed. The quantum and classical resources are counted explicitly

(Section 3.8). Finally, the numerical verification is extended to instance classes with few satisfying assignments, where the worst-case amplitude bound is nearly tight (Section 4).

The remainder of this paper is organized as follows. Section 2 introduces the k -SAT problem, defines quantum analogs of the three fundamental classical logic gates involved, and reviews Grover's algorithm. Section 3 presents the proposed algorithm, beginning with an overview of the framework, followed by the construction of the problem-dependent state and the decomposition of the Grover oracle for k -SAT into elementary quantum gates. The section concludes with an analysis of the algorithm's feasibility and time complexity and with a comparison against prior quantum algorithms and additionally provides the pseudocode of the complete procedure (Section 3.2), a precise statement of the problem the algorithm solves (Section 3.6) and an analysis of its robustness to state-preparation and gate errors together with the full resource accounting (Section 3.8). Section 4 reports the numerical verification of the two bounds entering the exponent, on planted instances as well as on instance classes with few satisfying assignments; Section 5 offers concluding remarks, including the limitations of the study and directions for future work, and the Appendix A details the gate-level implementation of the conditional neighborhood expansion.

2. Preliminaries

2.1. Definition of the k -SAT Problem

The k -SAT problem in conjunctive normal form (CNF) is defined as follows. Let $X = \{x_1, x_2, \dots, x_n\}$ be a finite set of n Boolean variables, and let $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ be a set of m clauses. Each clause is given by $c_i = l_{i,1} \vee l_{i,2} \vee \dots \vee l_{i,k}$, where each $l_{i,j} \in \{x_p, \neg x_p \mid 1 \leq p \leq n\}$ is called a *literal*; for simplicity, the k literals of a clause are assumed to involve k distinct variables, so that no clause contains both x_p and $\neg x_p$. This is the case considered throughout, and it is what makes the k branches generated by F_{c_i} [Equation (5)] distinct. The k -SAT problem asks whether there exists an *assignment* $a = \{a_1, a_2, \dots, a_n\}$, such that the formula $\mathcal{F} = c_1 \wedge c_2 \wedge \dots \wedge c_m$ is satisfied. If $\mathcal{F}$ is satisfied by a , then a is called a *satisfying assignment* of $\mathcal{F}$.

Note that for fixed n and m , each distinct set of clauses defines a unique instance of the k -SAT problem. Throughout, m is taken to be polynomially bounded in n , so that the clause count contributes only a polynomial factor to all running times reported below. In classical computation, a Boolean variable takes a binary value of either zero (false) or one (true), and logical operations act on one or more such variables. By analogy, a *quantum truth value* can be defined as a superposition of the basis states $|0\rangle$ and $|1\rangle$. A *quantum logical variable* assumes such a quantum truth value and is naturally represented by a qubit, whose state encodes that value. Consequently, a *quantum logical operation* is defined as a quantum gate acting on one or more qubits, transforming their joint quantum state.

The k -SAT problem involves n Boolean variables, each taking a value in $\{0, 1\}$. Consequently, there are 2^n possible assignments, which constitute the search space $\mathcal{H}_n = \{0, 1\}^n$. Each assignment $a = \{a_1, a_2, \dots, a_n\}$ can be encoded as a computational basis state $|a\rangle = |a_1 a_2 \dots a_n\rangle$, where $a_i \in \{0, 1\}$. In the remainder of this paper, we identify each classical assignment with its corresponding computational basis state.

2.2. Quantum Realizations of Logic Gates: NOT, AND, and OR

As shown above, the k -SAT problem is built upon three basic logical operations: AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). Their quantum analogs, denoted as QAND, QOR, and QNOT, are defined as follows.

The classical NOT operation maps $0 \mapsto 1$ and $1 \mapsto 0$. Its quantum analog, QNOT, is defined by its action on the computational basis:

$$\text{QNOT}|x\rangle = |\neg x\rangle, \quad x \in \{0, 1\}.$$

This operator is equivalent to the single-qubit Pauli-X gate, which flips $|0\rangle \leftrightarrow |1\rangle$.

The classical AND gate ($x_1 \wedge x_2$) embeds into a reversible three-qubit operator QAND:

$$\text{QAND}|x_1, x_2, x_3\rangle = |x_1, x_2, x_3 \oplus (x_1 \wedge x_2)\rangle.$$

This is precisely the Toffoli gate, which applies a bit-flip to the third qubit if and only if the first two qubits are both $|1\rangle$.

The classical OR operation acts on two Boolean variables, yielding zero if and only if both inputs are zero. Its reversible quantum analog, QOR, is defined on three qubits as:

$$\text{QOR}|x_1, x_2, x_3\rangle = |x_1, x_2, x_3 \oplus (x_1 \vee x_2)\rangle,$$

where $\oplus$ denotes addition modulo two. This operation flips the target qubit x_3 conditioned on at least one of the control qubits x_1 or x_2 being one. The QOR gate can be implemented via QNOT and QAND. Specifically, one applies QNOT to each of the first two qubits, then QAND to all three qubits, and finally QNOT to each of the three qubits.

2.3. Grover's Algorithm

In Grover's algorithm, the Grover iteration is applied $O(\sqrt{N/M})$ times to the system state, where $N = 2^n$ is the search space size (n being the number of qubits) and M is the number of solutions. The qubits are then measured in the computational basis, which returns a solution with probability $\Omega(1)$; repeating the whole procedure $O(\log(1/\delta))$ times raises the success probability to at least $1 - \delta$.

The Grover iteration consists of the following four steps:

- Apply the oracle $O: |x\rangle \mapsto (-1)^{f(x)}|x\rangle$, where $f(x) = 1$ if x is a solution and $f(x) = 0$ otherwise. This marks the solution states by flipping their phases.
- Apply the Hadamard transform $H^{\otimes n}$.
- Apply the diffusion operator $D_0 = 2|0\rangle\langle 0| - I$, which reflects the state about $|0\rangle$ (equivalently, flips the phase of all non-zero computational basis states).
- Apply the Hadamard transform $H^{\otimes n}$.

Using the identity $H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I$, where $|\psi\rangle = H^{\otimes n}|0\rangle$ is the equal superposition state, the Grover iteration simplifies to

$$G = (2|\psi\rangle\langle\psi| - I)O, \quad (1)$$

where O is the oracle defined in Step 1.

If an oracle is available that marks the solution states of a given problem, Grover's algorithm can solve it without exploiting additional problem-specific structure beyond the oracle itself. In particular, for the k -SAT problem, assuming such an oracle exists and the instance has a unique solution ($M = 1$), Grover's algorithm achieves a time complexity of $O(\sqrt{2^n}) \approx O(1.414^n)$. This provides a quadratic speedup over exhaustive classical search; it does not, however, beat the classical bounds of Section 1: for $k = 3$, its base 1.414 lies above the deterministic 1.3303^n of Ref. [12] and above the randomized 1.307^n of Ref. [8], as Table 1 also records. Nevertheless, incorporating problem structure into the quantum algorithm can further reduce the complexity, as demonstrated in the subsequent analysis.

Table 1. Exponential base α in the gate-model running time $O(\alpha^n)$ for worst-case 3-SAT. The first two rows are classical reference points included for comparison; the remaining rows are quantum gate-model algorithms.

Algorithm	Base α
Classical: Schöning’s local search [3]	1.333
Classical: best known (PPSZ-type) [6–8]	1.307
Grover search (oracle only) [14]	1.414
Grover search + Hofmeister et al. [5,27]	1.153
Amplitude amplification + PPSZ [6,8,28]	1.143
This work (Grover within a Hamming ball, hybrid)	1.268

3. The Proposed Algorithm

3.1. Algorithm Framework

The proposed algorithm is a hybrid quantum–classical procedure whose outer loop performs t classically repeated iterations, each consisting of the following three steps:

- **Random assignment generation.** Prepare the uniform superposition state by applying $H^{\otimes n}$ to n qubits initialized in $|0\rangle^{\otimes n}$, then measure all qubits in the computational basis to obtain a random assignment $|a\rangle \in \mathcal{H}_n$.
- **Subspace state preparation.** Construct a problem-dependent quantum state $|\psi_a\rangle$ starting from the randomly generated assignment $|a\rangle$.
- **Grover search within subspace.** Apply Grover’s algorithm to $|\psi_a\rangle$ to search for a solution to the k -SAT instance. The number of solutions inside a ball is not known in advance, so the exponential-search variant of amplitude amplification is used, as in Step 3 of Algorithm 1 (Section 3.2).

Algorithm 1 Hybrid quantum–classical k -SAT search.

Input: a k -CNF formula $\mathcal{F}$ with n variables and m clauses; radius $r = \lceil n/(1 + \sqrt{k}) \rceil$; $t = \lceil 2^{n(1-h(r/n))} \rceil$, the number of trials that brings the success probability to $\Omega(1)$; failure tolerance $\delta > 0$; total number of trials $T = \lceil c_0 t \ln(1/\delta) \rceil$, where the constant $c_0 = O(1)$ absorbs the $\Omega(1)$ (rather than unit) success probability of the amplification of Step 3.

Output: a satisfying assignment of $\mathcal{F}$, or FAIL.

Repeat Steps 1–4 for $j = 1, \dots, T$; the T trials are independent classical repetitions of the same procedure.

1. *Sample a center.* Prepare $H^{\otimes n}|0\rangle^{\otimes n}$, measure in the computational basis and store the outcome as the classical string $a \in \{0, 1\}^n$.
 2. *Prepare the ball state (quantum).* Load $|a\rangle$ by a layer of Pauli-X gates and apply the construction $|\psi_a\rangle = C_a|0\rangle$, $C = F_r E_r \dots F_1 E_1$ [Equation (6)]: in each of the r rounds evaluate all clauses (E_i) and expand the first false clause coherently (F_i), keeping the label register $|s_i\rangle$ and the clause ancillas of the round.
 3. *Amplify (quantum).* Run the exponential-search variant of amplitude amplification with the oracle O [Equation (9)] and the reflection $2|\psi_a\rangle\langle\psi_a| - \mathbb{1} = C_a(2|0\rangle\langle 0| - \mathbb{1})C_a^\dagger$, for $J = O(k^{r/2})$ iterations.
 4. *Read out and verify (classical).* Measure the assignment register, obtaining x ; evaluate the m clauses of $\mathcal{F}$ on x in $O(mk)$ time.
 If every clause is satisfied,
return x and halt; otherwise discard x and continue with the next trial.
 If all T trials have been run without a verified assignment,
return FAIL. (no certificate of unsatisfiability is produced)
-

3.2. Pseudocode of the Complete Procedure

Algorithm 1 collects the three steps of Section 3.1 into a single procedure and marks which operations are quantum, which are classical, and where the two meet. Step 1 is a

measurement: the center a is a classical bit string. The T repetitions of the loop are therefore independent classical trials, and the loop contributes the inverse success probability rather than its square root. This is what fixes the optimal radius at $\rho^* = 1/(1 + \sqrt{k})$ in Section 3.7. Steps 2 and 3 are performed coherently: the r rounds of state preparation, the J amplitude-amplification iterations and the oracle calls of one round are executed without intermediate measurement, the only measurements being the intermediate checks of the exponential-search schedule of Step 3 and the final readout of Step 4. That last step costs $O(mk)$ classical time per trial, a factor $O(k^{-r/2}/n)$ of the cost of one run, and $O(mk 2^{n(1-h(\rho^*))})$ over the t trials, which is sub-dominant to the total of Equation (12) and leaves the exponent unchanged. It is what makes the output verifiable, since the algorithm never returns an assignment that does not satisfy $\mathcal{F}$. A single trial succeeds with probability $p_B \cdot \Omega(1)$, so the budget $t = \lceil 2^{n(1-h(r/n))} \rceil = \Theta(1/p_B)$ is what lifts the success probability of the procedure to $\Omega(1)$. The trials are independent and identically distributed, so the single loop of $T = \lceil c_0 t \ln(1/\delta) \rceil$ trials already brings the failure probability below any prescribed $\delta > 0$. Note that the t samples that bring the success probability of a single trial to $\Omega(1)$ and the $O(\log(1/\delta))$ repetitions that raise it to $1 - \delta$ are not two separate loops: they are one and the same classical repetition, written here as the single loop of T trials that is also used in Section 3.6.

3.3. Construction of the Problem-Dependent State

The problem-dependent state $|\psi_a\rangle$ is constructed by applying the following two procedures iteratively r times, starting from the randomly generated assignment $|a\rangle$; throughout this section, $|\psi_a\rangle$ denotes the state of the round being considered.

- **Clause evaluation (E).** This procedure computes the quantum truth value of each clause in the k -SAT instance. For a clause $c_i = l_{i,1} \vee l_{i,2} \vee \dots \vee l_{i,k}$, we introduce $(k-1)$ auxiliary qubits $|\alpha_{i,1}\rangle, \dots, |\alpha_{i,k-1}\rangle$, all initialized to $|0\rangle$. Let $|l_{i,j}\rangle$ denote the qubit corresponding to literal $l_{i,j}$. The quantum OR operation (QOR) is applied sequentially: first to $|l_{i,1}\rangle, |l_{i,2}\rangle$, and $|\alpha_{i,1}\rangle$; then to $|l_{i,3}\rangle, |\alpha_{i,1}\rangle$, and $|\alpha_{i,2}\rangle$; and so on, until finally to $|l_{i,k}\rangle, |\alpha_{i,k-2}\rangle$, and $|\alpha_{i,k-1}\rangle$. The final result is stored in $|\alpha_{i,k-1}\rangle$. Across all m clauses, this requires $m(k-1)$ auxiliary qubits, with the truth values of clauses $c_1, \dots, c_m$ stored in $|\alpha_{1,k-1}\rangle, \dots, |\alpha_{m,k-1}\rangle$, respectively.

The full clause-evaluation operator is defined as

$$E = \text{QOR}_{c_m} \text{QOR}_{c_{m-1}} \dots \text{QOR}_{c_1}, \quad (2)$$

where QOR_{c_i} denotes the sequence of $(k-1)$ QOR gates applied to clause c_i , as described above. After applying E , the state of the current round becomes entangled with each clause register $|\alpha_{i,k-1}\rangle$ as

$$|\psi_a, \alpha_{i,k-1}\rangle = |\psi_a^{c_i,0}, 0\rangle + |\psi_a^{c_i,1}, 1\rangle,$$

where $|\psi_a^{c_i,0}\rangle$ and $|\psi_a^{c_i,1}\rangle$ are the (unnormalized) components of that state supported on the subspaces in which clause c_i evaluates to false and true, respectively.

- **Conditional neighborhood expansion (F).** This procedure applies a conditional transformation to each computational basis state in the superposition $|\psi_a\rangle$, expanding the neighborhood of *one* unsatisfied clause. Let $i(a) = \min\{i \mid c_i \text{ is false under } a\}$ be the index of the first clause that is false under a , and let $C_i(\cdot)$ denote conditioning on the clause-register pattern in which clauses $c_1, \dots, c_{i-1}$ are true and clause c_i is false, i.e., the multiply-controlled unitary that acts as the identity unless the clause registers $|\alpha_{1,k-1}\rangle, \dots, |\alpha_{m,k-1}\rangle$ hold that pattern and that applies the operation in its argument

whenever they do. The conditional expansion operator is defined as the product of these controlled unitaries,

$$F = \prod_{i=1}^m C_i(F_{c_i}), \quad (3)$$

where F_{c_i} is the expansion operator associated with clause c_i . The m control patterns are mutually exclusive, and the expansion F_{c_i} changes only the assignment and label registers. The clause registers are therefore untouched while a factor acts. Consider a computational basis state whose clause registers hold the values $(\eta_1, \dots, \eta_m)$. Exactly one sector of Equation (3) is then selected, namely, that of the first clause that is false, i.e., clause $c_{i(a)}$. If a satisfies every clause, no expansion takes place, and F acts as the identity on the assignment and label registers. On such a basis state, F therefore reduces to a direct sum over the mutually exclusive sectors. The summands are one unitary F_{c_i} for each i , together with the identity on the sector in which all clauses are true. An individual summand is only a partial isometry. It is the completeness of the sectors that makes the operator F unitary. Thus, F performs a single k -way branching per round instead of a simultaneous expansion of all m clauses. This mirrors the branching step of a classical local search, in which a single unsatisfied clause is selected and each of its k literals is tried in turn; applying instead the expansion of every false clause, with each F_{c_i} controlled by the single clause c_i , would generate k^{m_f} branches in a single round for an assignment with m_f false clauses and would not correspond to the classical search tree. The circuit implementation of F is illustrated in Figure 1.

Each block F_{c_i} acts on the assignment register and on an auxiliary label register $|s\rangle$ of $q = \lceil \log_2 k \rceil$ qubits, initialized to $|0\rangle^{\otimes q}$: a unitary W first prepares a uniform superposition over k basis states of $|s\rangle$, which then controls a Pauli-X flip of the corresponding literal,

$$F_{c_i} = \left(\sum_{j=1}^k X_{p(i,j)} \otimes |j\rangle\langle j|_s + \mathbb{1}_a \otimes \sum_{j=k+1}^{2^q} |j\rangle\langle j|_s \right) W, \quad (4)$$

where $W|0\rangle^{\otimes q} = k^{-1/2} \sum_{j=1}^k |j\rangle$ prepares the uniform superposition in $|s\rangle$, the label values being understood cyclically, $|2^q\rangle \equiv |0\rangle^{\otimes q}$, so that the 2^q basis states of $|s\rangle$ are indexed by $j = 1, \dots, 2^q$; the second sum then acts as the identity on the unused label values $j > k$, which for $k = 2^q - 1$ is the single all-zero label $|0\rangle^{\otimes q}$ (cf. Appendix A) and is empty when $k = 2^q$, so that it completes the first factor to a unitary on the whole register space. $X_{p(i,j)}$ flips the variable $x_{p(i,j)}$ to which the literal $l_{i,j}$ refers. Acting on an assignment a that falsifies c_i , this block gives

$$|a\rangle|0\rangle_s \xrightarrow{F_{c_i}} \frac{1}{\sqrt{k}} \sum_{j=1}^k |g_{l_{i,j}}(a)\rangle |j\rangle_s, \quad (5)$$

where $g_{l_{i,j}}(a)$ denotes the assignment obtained by flipping the bit corresponding to literal $l_{i,j}$ in a . Each $|g_{l_{i,j}}(a)\rangle$ differs from $|a\rangle$ in exactly one bit position and satisfies c_i , so F_{c_i} maps $|a\rangle$ to a uniform, coherent superposition over its k Hamming-distance-1 neighbors that satisfy c_i , with the chosen branch recorded in the label register $|s\rangle$. This label is retained rather than uncomputed, and no unitary can do otherwise: the corresponding map on the assignment register alone, $T_{\text{bf}}|a\rangle = k^{-1/2} \sum_{j=1}^k |g_{l_{i,j}}(a)\rangle$, is not an isometry. For $k = 2$ and $\{p(i,1), p(i,2)\} = \{1, 2\}$, it gives $T_{\text{bf}}|00\rangle = T_{\text{bf}}|11\rangle = (|10\rangle + |01\rangle)/\sqrt{2}$. The obstruction persists for $k \geq 3$ inside the sector where F_{c_i} is active. Were the label reset, the falsifying input $|000\rangle$ would keep a component of amplitude

$1/\sqrt{k}$ on $|100\rangle|0\rangle^{\otimes q}$. That is the very state that the control leaves untouched for the orthogonal input $|100\rangle$. Therefore, no unitary that generates Equation (5) can also return $|s\rangle$ to $|0\rangle^{\otimes q}$. Consequently, each of the r rounds of the construction employs its own label register $|s_1\rangle, \dots, |s_r\rangle$ (and, in the case $k = 2^q - 1$, its own copy of the auxiliary register $|w\rangle$ of Appendix A), and each round keeps its own copy of the clause registers as well, since these cannot be uncomputed either (see the discussion of Equation (6)). We provide an implementation scheme of F_{C_i} using elementary quantum gates in Appendix A.

The full construction procedure can then be written as

$$C = F_r E_r \cdots F_2 E_2 F_1 E_1, \quad (6)$$

where E_i denotes the clause-evaluation operator [Equation (2)] acting on the i th copy of the clause registers, which starts in $|0\rangle^{\otimes m(k-1)}$, and F_i the conditional neighborhood expansion [Equation (3)] controlled by those registers and acting on the label register $|s_i\rangle$ of the i th round. In each round, the quantum truth values of all clauses are first computed by E_i for the assignment currently held in the assignment register, and the conditional neighborhood expansion F_i is then applied on the basis of these truth values. Unlike the oracle of Equation (9), the expansion changes the assignment register, so the clause registers of a round cannot be uncomputed by $E_i^\dagger$: applying $E_i^\dagger$ after the expansion computes the clause values of the new assignment on top of those of the previous one, and the registers return to $|0\rangle$ only when the assignment register is left unchanged. That holds for the oracle but not here. Like the branch label of F [Equation (5)], they are therefore retained rather than uncomputed and reused. This iterative process leverages the clause structure of the k -SAT instance to transform the initial random assignment into a problem-dependent superposition. The resulting state is supported on basis states of the form $|x\rangle|s_1\rangle \cdots |s_r\rangle$, tensored with the retained clause registers, in which $|x\rangle$ lies in the Hamming ball $B(a, r)$, and the labels record the k -way choices taken along the corresponding branch of the search tree, so that the state forms a local search subspace tailored to the problem instance.

Remark on unitarity. All the operators employed above are unitary: QNOT, QAND, QOR and CNOT act as permutations of the computational basis (and are self-inverse); the Hadamard transform H and $D_0 = 2|0\rangle\langle 0| - I$ are Hermitian and satisfy $H^2 = D_0^2 = \mathbb{1}$; W is a state-preparation unitary [Equation (4) and Appendix A]; E [Equation (2)], $\mathcal{T}$ [Equation (7)], F_{C_i} [Equation (4)] and F [Equation (3)] are products of multiply-controlled unitaries, each control condition being a computational-basis pattern on the clause registers (so that the operator reduces to a direct sum of unitaries over the mutually exclusive sectors), and the same holds for the construction C [Equation (6)], the oracle O [Equation (9)] and the Grover iteration G [Equation (1)]. Within the construction, the only non-unitary object in the analysis is the branch-and-forget map T_{bf} introduced below Equation (5). It is not an executed operation but the shortcut of discarding the branch labels. It is also the reason why those labels and, as explained above, the clause registers of each round are retained rather than uncomputed. In this way, every step of the construction is a unitary gate. Non-unitary operations occur only where the algorithm measures: in generating the random assignment in Step 1, in the intermediate checks performed by the exponential-search variant of amplitude amplification in Step 3, and in the final readout. It is the measurement in Step 1 that keeps the outer loop over balls a classical repetition.

3.4. Oracle for the k -SAT Problem

In this subsection, we decompose the oracle O into elementary quantum operations. As previously described, the oracle O identifies solution states and flips their phases. To detect these states, we first apply the clause-evaluation operator E [Equation (2)] to compute the quantum truth values of all m clauses. Subsequently, the quantum AND operation

(QAND) is applied sequentially $(m - 1)$ times to evaluate the entire formula $\mathcal{F}$. This formula-evaluation procedure is defined as

$$\mathcal{T} = \text{QAND}_{\alpha_{m,k-1}, \beta_{m-2}, \beta_{m-1}} \cdots \text{QAND}_{\alpha_{3,k-1}, \beta_1, \beta_2} \cdot \text{QAND}_{\alpha_{1,k-1}, \alpha_{2,k-1}, \beta_1}, \quad (7)$$

where $\beta_1, \dots, \beta_{m-1}$ are auxiliary qubits initialized to $|0\rangle$, and $\text{QAND}_{x,y,z}$ denotes the QAND gate acting on qubits x, y , and z . This step requires $(m - 1)$ auxiliary qubits in total, with the final truth value of $\mathcal{F}$ stored in $|\beta_{m-1}\rangle$.

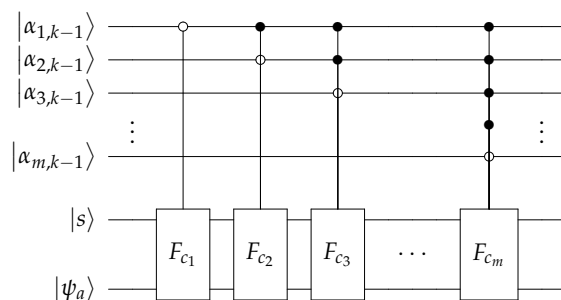


Figure 1. Quantum circuit implementing procedure F . The qubit $|\alpha_{i,k-1}\rangle$ encodes the quantum truth value of the i th clause; the block F_{c_i} is activated only when $|\alpha_{i,k-1}\rangle = |0\rangle$ (open circle, i.e., clause c_i is false) while all preceding clauses are true (filled circles), so that at most one block acts on each computational basis state, and no block acts when every clause is satisfied (the filled control drawn on the row of dots belongs to the control line coming from the preceding clause qubits and passes through the omitted rows); $|s\rangle$ denotes the label register of the round, shared by its blocks F_{c_i} , prepared in a uniform superposition of k basis states and retained as the branch label [Equations (4) and (5)] (each round uses its own copies of the clause registers and its own register $|s_i\rangle$); and $|\psi_a\rangle$ denotes the state of the round being expanded, i.e., the problem-dependent superposition constructed so far.

Writing the collection of labels produced by the construction as $|l\rangle$, the problem-dependent state $|\psi_a\rangle$ can be decomposed into two orthogonal components based on the value of $f(x)$:

$$|\psi_a\rangle = |\psi_a^0\rangle + |\psi_a^1\rangle = \sum_{x: f(x)=0} \sum_l A_{x,l} |x\rangle |l\rangle + \sum_{x: f(x)=1} \sum_l B_{x,l} |x\rangle |l\rangle, \quad (8)$$

where $|\psi_a^0\rangle$ and $|\psi_a^1\rangle$ are unnormalized and span the subspaces corresponding to unsatisfying and satisfying assignments, respectively (the label and clause registers of the construction are spectators: the oracle below acts as the identity on them and uses its own clause and formula registers, initialized to $|0\rangle$, so the marking condition depends only on the assignment). After applying the clause-evaluation operator E [Equation (2)] and the formula-evaluation operator $\mathcal{T}$ [Equation (7)], the system becomes entangled with the ancilla qubit $|\beta_{m-1}\rangle$:

$$|\psi_a, \beta_{m-1}\rangle = |\psi_a^0, 0\rangle + |\psi_a^1, 1\rangle.$$

Finally, a controlled-NOT (CNOT) gate is applied to the ancilla qubits $|\beta_{m-1}\rangle$ (control) and $|\beta_m\rangle$ (target), where $|\beta_m\rangle$ is initialized to $|- \rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. This operation transforms the entangled state into

$$|\psi_a, \beta_{m-1}, \beta_m\rangle = |\psi_a^0, 0, - \rangle - |\psi_a^1, 1, - \rangle.$$

The complete oracle for the k -SAT problem is then given by

$$O = E^\dagger \mathcal{T}^\dagger \text{CNOT}_{\beta_{m-1}, \beta_m} \mathcal{T} E, \quad (9)$$

where E^+ and $\mathcal{T}^+$ are the inverses of the clause-evaluation operator E [Equation (2)] and the formula-evaluation operator $\mathcal{T}$ [Equation (7)], respectively. These uncomputation steps restore the clause and formula registers to their initial states, ensuring that the oracle acts as a pure phase oracle on the computational basis; it is the identity on the label and clause registers of the construction.

3.5. Reachability Within Hamming Balls

Suppose $a^* \in B(a, r)$ is a satisfying assignment. We now show that after r iterations of the construction procedure [Equation (6)], the state $|\psi_a\rangle$ has amplitude at least $k^{-r/2}$ on the marked subspace, i.e., on the components of Equation (8) whose assignment register holds a satisfying assignment. The *Hamming distance* between two assignments x and y , denoted $d(x, y)$, is the number of bit positions in which they differ. The set of all assignments within Hamming distance r from a center assignment a is called a *Hamming ball*, denoted $B(a, r) = \{x \mid d(x, a) \leq r\}$.

By construction [Equation (6)], each literal-flip operator $g_{l_{i,j}}$ satisfies $d(a, g_{l_{i,j}}(a)) = 1$. Consequently, all assignments reachable via the state-generation procedure lie within $B(a, r)$. For any clause $c_i = l_{i,1} \vee l_{i,2} \vee \dots \vee l_{i,k}$ that is unsatisfied by a , at least one literal $l_{i,j}$ in c_i must evaluate to true under a^* . Flipping the corresponding variable in a yields an assignment $g_{l_{i,j}}(a)$ such that

$$d(g_{l_{i,j}}(a), a^*) = d(a, a^*) - 1.$$

Thus, after r rounds of the construction procedure [Equation (6)], the state $|\psi_a\rangle$ necessarily contains a component whose assignment register holds a satisfying assignment: at every round the selected clause contains at least one literal that is true under a^* (which satisfies every clause), so a distance-decreasing flip is always available, and the path that applies such a flip in each of the r rounds ends at a^* unless it meets another satisfying assignment after $d < r$ rounds, in which case no further expansion takes place and the path ends there. More quantitatively, every branch of the expansion carries the same amplitude $k^{-1/2}$ [Equation (5)], so the distance-decreasing path carries the amplitude $k^{-r/2}$, or the larger amplitude $k^{-d/2}$ if it ends after $d < r$ rounds. Distinct paths are recorded by distinct label configurations and are therefore mutually orthogonal, so their contributions add in quadrature, with no interference and hence no cancellation; the squared overlap, i.e., the probability, of $|\psi_a\rangle$ with the marked subspace is thus at least k^{-r} , since the distance-decreasing path alone has an amplitude of magnitude no less than $k^{-r/2}$ on that subspace. Moreover, once the assignment register reaches a satisfying assignment, it is no longer modified: every clause is satisfied there, hence none of the control patterns of Equation (3) holds, and F acts as the identity on the assignment and label registers, so the amplitude of the component that ends there is preserved. The amplitude of $|\psi_a\rangle$ on the marked subspace is therefore at least $k^{-r/2}$, so amplitude amplification locates a satisfying assignment within $O(k^{r/2})$ iterations. Since the number of satisfying assignments inside the ball is not known in advance, we use the exponential-search variant of amplitude amplification [28], which preserves this scaling up to a constant factor.

3.6. Nature of the Algorithm: Search, Not Decision

We now state what Algorithm 1 computes: the distinction determines how its running time and its claimed advantage are to be read.

It is a search algorithm. Given a satisfiable instance, the algorithm returns a satisfying assignment. On every outer iteration, the assignment produced by the final measurement is checked against all m clauses in $O(mk)$ classical time, and it is returned only if the check succeeds. The output is therefore always a genuine solution, and the error is one-sided: the algorithm can fail to find a solution, but it cannot return an incorrect one. Formally, for every satisfi-

able instance and every constant $\delta > 0$, running the procedure for $O(2^{n(1-h(\rho^*))} \log(1/\delta))$ outer iterations returns a satisfying assignment with probability at least $1 - \delta$. The cost is $O(\text{poly}(n) 2^{n(1-h(\rho^*))} k^{\rho^* n/2} \log(1/\delta))$ elementary gates. The dependence on δ is the multiplicative factor $\log(1/\delta)$, because the passes are independent classical repetitions.

It is not a bounded-error decision procedure. There is no mechanism for certifying that no satisfying assignment exists. On an unsatisfiable instance, every outer iteration ends with Step 4 finding a violated clause. Once the budget is exhausted, the algorithm reports FAIL without having produced a certificate. The failure probability δ is therefore the probability of not having hit a solution within the allocated repetitions, not the probability of a wrong yes/no answer. The algorithm is a zero-error (Las Vegas) search procedure with a randomized running time, not a decision procedure with two-sided error.

This has three consequences for the comparisons of Section 3.9. (i) The bound of Equation (11) is a worst-case bound over *satisfiable* instances, with respect to an arbitrary satisfying assignment a^* of the given instance. Nothing is asserted about unsatisfiable instances, beyond the fact that the algorithm terminates and reports failure. (ii) The classical reference values that apply are therefore the exponents of the classical *search* algorithms: Schönning's local search [3] and the PPSZ-type algorithms [6–8]. These are randomized search procedures of the same kind. Comparing with the deterministic algorithms of Section 1 is not a fair comparison, because those do decide satisfiability. Their base is larger (1.3303^n for 3-SAT). Unlike the present algorithm, however, they also settle the unsatisfiable case. If a decision answer is required, the search has to be combined with a deterministic decision procedure, and the resulting base is the larger of the two. (iii) Every exponent quoted in this paper is accordingly the base of a search-to-solution procedure. The improvement claimed is an improvement of that base over the classical search bounds. It is not a claim about deciding satisfiability.

3.7. Time Complexity

Throughout, running times are counted as the number of elementary gates; the depth of a single run is of the same order as its gate count, since the $O(k^{\rho n/2})$ amplitude-amplification iterations are sequential, and the t classically repeated runs multiply both quantities by the same factor.

The size of the Hamming ball $B(a, r)$ satisfies the following bounds [9]:

$$\frac{2^{nh(\rho)}}{\sqrt{8n\rho(1-\rho)}} \leq |B(a, r)| \leq 2^{nh(\rho)},$$

where $\rho = r/n$, and $h(\rho) = -\rho \log_2 \rho - (1 - \rho) \log_2 (1 - \rho)$ is the binary entropy function.

Assume the given k -SAT instance has at least one satisfying assignment, and fix one such assignment a^* . The analysis below therefore does not require the instance to have a unique solution, and the unique-solution assumption made for plain Grover search in Section 2.3 is not needed here. Under the uniform distribution over all 2^n assignments, the probability that a fixed solution lies in $B(a, r)$ is at least

$$p_B = \frac{|B(a, r)|}{2^n} \geq \frac{2^{n(h(\rho)-1)}}{\sqrt{8n\rho(1-\rho)}}. \quad (10)$$

Since the initial random assignment is measured before the quantum search, different balls are sampled classically and independently, so the outer loop is a purely classical repetition. The probability that a single sampled ball contains a fixed satisfying assignment is p_B [Equation (10)], so $t = O(1/p_B) = O(2^{n(1-h(\rho))})$ repetitions are required to locate a ball containing a solution. Within such a ball, the constructed state is spread over the reachable branches of the search tree, at most k^r of them, each recorded by its own label configuration; by the amplitude bound established in Section 3.5, its amplitude on the satisfying assign-

ments is at least $k^{-r/2}$; applying amplitude amplification [28] therefore locates a satisfying assignment in $O(k^{r/2}) = O(k^{\rho n/2})$ iterations. Note also that the reflection is taken about the prepared state $|\psi_a\rangle$ rather than about the equal superposition $H^{\otimes n}|0\rangle$ [cf. Equation (1)]: writing $|\psi_a\rangle = C_a|0\rangle$, where C_a denotes the construction C [Equation (6)] preceded by the Pauli-X layer that prepares the measured assignment $|a\rangle$, the diffusion operator is realized as $C_a(2|0\rangle\langle 0| - I)C_a^\dagger = 2|\psi_a\rangle\langle\psi_a| - I$. The total runtime (up to polynomial factors) is therefore

$$T_{\text{tot}} = 2^{n(1-h(\rho))} \cdot k^{\rho n/2}.$$

Minimizing this expression with respect to ρ (equivalently, minimizing $(1-h(\rho)) + \frac{\rho}{2} \log_2 k$) and using $h'(\rho) = \log_2((1-\rho)/\rho)$ yields the optimal choice $\rho^* = 1/(1+\sqrt{k})$, i.e., $r^* = n/(1+\sqrt{k})$, which gives

$$T_{\text{tot}} = \left(\frac{2\sqrt{k}}{1+\sqrt{k}} \right)^n = \left(2 - \frac{2}{1+\sqrt{k}} \right)^n. \quad (11)$$

For comparison, the corresponding classical template performs $t \cdot k^r$ local-search steps, since the search tree of depth r has k^r leaves, and the two factors are balanced at $\rho = 1/(k+1)$; the resulting exponent is $(2 - 2/(k+1))^n$, i.e., exactly the Hamming-ball local search of Dantsin et al. [9]. Replacing k^r by the quadratic $k^{r/2}$ of amplitude amplification is what shifts the optimal radius to $\rho^* = 1/(1+\sqrt{k})$ and yields Equation (11). In particular, when $k = 3$, the time complexity is $T_{\text{tot}} \approx 1.268^n$.

Finally, we make the polynomial overhead explicit. Each QOR gate is implemented by a single QAND (Toffoli) gate together with a constant number of single-qubit QNOT gates, so the clause-evaluation operator E [Equation (2)] requires $m(k-1)$ QOR gates and $m(k-1)$ auxiliary qubits, while the formula-evaluation operator $\mathcal{T}$ [Equation (7)] requires $m-1$ QAND gates and $m-1$ auxiliary qubits. Since the oracle [Equation (9)] consists of E , $\mathcal{T}$, their inverses, and a single CNOT, a single oracle call costs $O(mk)$ elementary gates and uses mk auxiliary qubits of its own. Together with the $r = \rho n$ rounds of state preparation [Equation (6)] and the $O(k^{\rho n/2})$ Grover iterations performed for each of the $t = O(2^{n(1-h(\rho))})$ sampled balls, the total number of elementary gates over all runs is

$$O(nmk) \left(2 - \frac{2}{1+\sqrt{k}} \right)^n, \quad (12)$$

with $O(mk)$ auxiliary qubits for the oracle's own clause and formula registers, which are uncomputed by $E^\dagger$ and $\mathcal{T}^\dagger$ and are therefore reused across all Grover iterations and t sampled balls. The registers of the construction, by contrast, cannot be uncomputed and are therefore retained: because both the branch label [Equation (5)] and the clause registers [Equation (6)] of a round are kept, each of the $r = \rho n$ rounds uses its own label register of $q = \lceil \log_2 k \rceil$ qubits, plus L qubits in the case $k = 2^q - 1$ (see Appendix A), and its own $m(k-1)$ clause ancillas, i.e., $O(r(\log k + L)) = O(n \log n)$ and $O(rmk)$ qubits in total, respectively. Both counts are polynomial in n for $m = \text{poly}(n)$ and leave the exponential scaling of Equation (11) unchanged. In summary, the explicit construction fixes the polynomial overhead of the algorithm rather than leaving it inside an unaccounted-for oracle. Table 2 collects the resulting resources for $k = 3, 4, 5$: the optimal radius $\rho^* = 1/(1+\sqrt{k})$, the number $t = O(2^{n(1-h(\rho^*))})$ of classically repeated runs, the number $O(k^{\rho^* n/2})$ of amplitude-amplification iterations performed in each run, and the exponential base α of Equation (11). The two factors balance at ρ^* , and their product reproduces α ; note that the classical factor alone, $2^{n(1-h(\rho^*))}$, is much smaller than the total cost, so that the exponent is dominated by the inner quantum search. The polynomial factors are those of Equations (12) and (13) and the accompanying text: $O(nmk)$ gates per state preparation and per amplitude-amplification iteration, and $O(rmk + n \log n + mk)$ qubits in total, with $r = \rho^* n$; a single run costs the $O(nmk k^{\rho^* n/2})$

gates of Equation (13). Both polynomial factors multiply the base α^n ; Section 3.8 collects them, together with the classical cost, in one place.

3.8. Robustness and Resource Requirements

The exponent of Equation (11) is derived for an ideal, noiseless device. We now quantify how state-preparation and gate errors degrade it. We also list the assumptions a reliable implementation requires, and collect the resources in one place.

State-preparation errors of the branch label. For $k = 2^q - 1$ (in particular for $k = 3$) the uniform superposition over the k branch labels is prepared only approximately. The unwanted all-zero label keeps a residual weight $4^{-(L+1)}$ per round, where L is the number of iterations of the scheme in Appendix A. The residual component is carried by its own register $|w\rangle$. It is therefore orthogonal to the branches of the search tree recorded by $|s\rangle$ and by the other copies of $|w\rangle$, and it does not interfere with them. The amplitude of every retained branch is multiplied by $\sqrt{1 - 4^{-(L+1)}}$ instead of by 1. Over the $r = \rho n$ rounds the marked amplitude is thus multiplied by $1 - O\left(r 4^{-(L+1)}\right)$, and with $L = (\frac{1}{2} + \varepsilon) \log_2 n$, this factor is $1 - O(n^{-2\varepsilon}) = 1 - o(1)$. The number of amplitude-amplification iterations $O(1/\gamma)$ and the total cost are multiplied by $1 + o(1)$, so this source of error changes the prefactor and not the exponent.

Generic state-preparation errors. The picture changes if the noise is not orthogonal to the constructed state. Suppose the state actually prepared in a ball satisfies $|\langle \tilde{\psi}_a | \psi_a \rangle|^2 \geq 1 - \eta$. Its amplitude on the marked subspace is then bounded below only by $\sqrt{1 - \eta} \gamma - \sqrt{\eta}$. Amplitude amplification still returns a solution in $O(1/\gamma)$ iterations, as long as $\eta = O(\gamma^2) = O(k^{-r})$. The admissible error is therefore exponentially small in r , rather than a constant. This is the precision that any quadratic speedup demands of its initial state. It is the price paid for amplifying an amplitude as small as $k^{-r/2}$.

Gate errors. A single run consists of the state preparation, of $O(nmk)$ gates, followed by $J = O(k^{\rho^* n/2})$ amplitude-amplification iterations of $O(nmk)$ gates each, so its depth is

$$D = O(nmk k^{\rho^* n/2}), \quad (13)$$

which for $k = 3$ is $O(nmk) \cdot 1.223^n$ (Table 2). If every elementary gate fails independently with probability p_g , the state of the run deviates from the ideal one by $O(p_g D)$, and combining this with the bound $\eta = O(k^{-r})$ above gives the requirement

$$p_g = O\left(\frac{k^{-r}}{nmk k^{r/2}}\right) = O\left(\frac{k^{-3r/2}}{nmk}\right)_{k=3, r=\rho^* n} = O\left(\frac{0.547^n}{nmk}\right). \quad (14)$$

The per-gate error rate must therefore decrease exponentially with n . Read the other way round, a fixed physical error rate p_g limits the usable instance size. The limit is $n \lesssim \ln(1/p_g) / \ln(1/0.547)$, that is, $n \lesssim 1.66 \ln(1/p_g)$. This is of order 23 variables at $p_g = 10^{-6}$, and of order 57 at $p_g = 10^{-15}$. The usable size therefore grows only logarithmically in $1/p_g$. The exponent of Equation (11) can consequently be realized only on fault-tolerant, logically encoded gates, whose error rate scales as the inverse of the gate count. This is the same requirement that any Grover-type quadratic speedup places on its hardware, and it is the strongest assumption of the present analysis. On a device with a constant error rate the algorithm remains correct, since every returned assignment is still verified. The window of n in which it beats the classical search bounds is, however, small or empty.

Effect on the success probability. Errors enter the running time in two different ways, and only one of them affects the exponent. An error that reduces the success probability of a single ball by a constant factor $c < 1$ is absorbed by the outer loop. This is the situation covered by the branch-label analysis above. The outer loop is a classical repetition over independent balls, so the number of repetitions t is multiplied by $1/c$, and so is the total

cost. Equation (11) is left unchanged. An error budget that grows with n is different. It either forces J to exceed $O(k^{r/2})$ or makes the amplified amplitude vanish. The exponent then degrades towards the classical value. This is the content of Equations (13) and (14).

Assumptions required for a reliable implementation. In summary, the analysis assumes that (i) the instance is satisfiable, the algorithm being a search procedure that cannot refute unsatisfiability (Section 3.6); (ii) the device holds the n assignment qubits together with $O(rmk + n \log n + mk)$ ancillas, and executes the D gates of Equation (13) coherently within one run; its coherence time must therefore grow with D ; (iii) the effective per-gate error rate obeys Equation (14), which in practice means fault-tolerant operation; (iv) no classical pre-processing beyond reading the formula is used, and the output is verified classically in $O(mk)$ time; (v) the number of solutions inside a ball need not be known, which is why the exponential-search variant of amplitude amplification is used. Assumptions (ii) and (iii) are the demanding ones. They are the reason why the result is to be read as a statement about the gate complexity of the construction, rather than as a near-term prescription.

Resources. Table 2 collects the exponential ingredients at the optimal radius. The polynomial factors are as follows. The state preparation costs r rounds of $O(mk)$ gates each (clause evaluation plus the conditional structure of Appendix A, which adds $O(m)$ gates per round), that is, $O(nmk)$ gates in total. One amplitude-amplification iteration costs one oracle call, $O(mk)$ gates, plus the reflection $C_a(2|0\rangle\langle 0| - \mathbb{1})C_a^\dagger$ about $|\psi_a\rangle$ [Equation (1)]. That reflection is realized by two state preparations ($C_a^\dagger$ and C_a) of $O(nmk)$ gates each. A single run therefore uses $O(nmk k^{\rho^* n/2})$ gates and $O(rmk + n \log n + mk)$ qubits. Over the $t = 2^{n(1-h(\rho^*))}$ balls, the whole algorithm uses the $O(nmk) \left(2 - 2/(1 + \sqrt{k})\right)^n$ elementary gates of Equation (12). The classical cost stays sub-dominant throughout and amounts to $O((n + mk) 2^{n(1-h(\rho^*))})$, which is 1.037^n up to polynomial factors for $k = 3$ and is absorbed into the 1.268^n of Equation (12) without changing the exponent. It covers the sampling of the t centers, the $O(mk)$ check of each readout and the bookkeeping that the exponential-search schedule requires.

3.9. Comparison with Prior Quantum Algorithms

Table 1 collects the best known exponents for worst-case 3-SAT in the gate model together with the exponent obtained here. Table 3 lists the same comparison for general k with $3 \leq k \leq 10$, where Schöning's bound $(2 - 2/k)^n$ and the deterministic Hamming-ball template $(2 - 2/(k + 1))^n$ are confronted with the exponent $(2 - 2/(1 + \sqrt{k}))^n$ obtained here. The exponent of the present algorithm lies below both classical bounds for every $k \geq 3$, since the optimal fraction $1/(1 + \sqrt{k})$ exceeds both $1/k$ and $1/(k + 1)$; the gap to the Hamming-ball template that it accelerates is substantial—at $k = 3$ the base falls from 1.500 to 1.268, a reduction of about 15%—and it persists over the whole range, the exponent obtained here being strictly smaller than both classical exponents.

Table 2. Resources of the proposed algorithm at the optimal radius $\rho^* = 1/(1 + \sqrt{k})$, for $k = 3, 4, 5$. Classically repeated runs: $t = O(2^{n(1-h(\rho^*))})$, with h the binary entropy function. Amplitude-amplification iterations per run: $O(k^{\rho^* n/2})$. Exponential base of Equations (11) and (12): $\alpha = 2^{1-h(\rho^*)} k^{\rho^* n/2} = 2 - 2/(1 + \sqrt{k})$, so that $t \cdot k^{\rho^* n/2} = \alpha^n$ up to polynomial factors.

k	ρ^*	t	Iterations	α
3	0.366	1.037^n	1.223^n	1.268
4	0.333	1.058^n	1.260^n	1.333
5	0.309	1.078^n	1.282^n	1.382

Table 3. Exponential bases α in the running time $O(\alpha^n)$ for general k : Schönning’s local search $(2 - 2/k)^n$ [3], the deterministic Hamming-ball template $(2 - 2/(k + 1))^n$ [9], and the exponent $(2 - 2/(1 + \sqrt{k}))^n$ obtained here. The present exponent lies below both classical bounds for every $k \geq 3$.

k	Schönning	Hamming Ball	This Work
3	1.333	1.500	1.268
4	1.500	1.600	1.333
5	1.600	1.667	1.382
6	1.667	1.714	1.420
7	1.714	1.750	1.451
8	1.750	1.778	1.478
9	1.778	1.800	1.500
10	1.800	1.818	1.519

A quantum algorithm that accelerates a classical solver as a black box achieves, up to polynomial factors, the square root of that solver’s exponent; the quality of the resulting bound is therefore determined by the quality of the underlying classical algorithm. Grover search alone gives $O(1.414^n)$ [14]; combining it with the best classical 3-SAT algorithm of its time ($O(1.3302^n)$ [5]) gives $O(1.153^n)$ [27]; and applying amplitude amplification [28] to the PPSZ algorithm [6–8] gives $O(1.143^n)$. The present construction, by contrast, accelerates a Hamming-ball (Dantsin-type) local search whose classical exponent is $(2 - 2/(k + 1))^n$ (1.5^n for $k = 3$; instance-specific refinements of this Hamming-ball search reach 1.481^n [9], 1.473^n [4], and 1.465^n [11], but they modify the local search itself rather than the ball-by-ball template accelerated here), and it does so only in its inner loop: because the random assignment of Step 1 is measured, the outer loop over balls remains classical, so the exponent is $(2 - 2/(1 + \sqrt{k}))^n$ rather than the square root $(2 - 2/(k + 1))^{n/2}$ that a fully coherent variant would give. For general k , the classical reference points are Schönning’s local search $(2 - 2/k)^n$ [3], the deterministic Hamming-ball search $(2 - 2/(k + 1))^n$ [9], and the PPSZ-type bounds [6,8]; the exponent $(2 - 2/(1 + \sqrt{k}))^n$ obtained here lies below Schönning’s $(2 - 2/k)^n$ for every $k \geq 3$ (for instance $1.268 < 1.333$ at $k = 3$ and $1.333 < 1.5$ at $k = 4$; Table 3 extends the comparison up to $k = 10$), so the improvement claimed in the abstract refers to the classical local-search bounds, while for $k = 3$ it also applies to the best known classical algorithm, whose exponent is 1.307^n [8]. Consequently, the exponents obtained here do not compete with those listed above, and we make no claim of an improvement in the asymptotic exponent over previous quantum algorithms. The contribution of this work is instead an explicit, gate-level realization of a problem-structured quantum local search, in which the problem-dependent superposition over the Hamming ball and the k -SAT oracle are both decomposed into elementary quantum gates (Sections 3.3 and 3.4). Extending this explicit construction so that its exponent becomes competitive—for example, by removing the measurement in Step 1 so that the outer loop can also be amplified—is an interesting direction for future work; for a recent study of quantum advantage for structured k -SAT that accounts for fault-tolerant overheads, see Ref. [29].

4. Numerical Verification

Experimental setup. For reproducibility, the ingredients of this section are stated explicitly. *Instances:* random three-CNF formulas over n variables with $m = \lfloor 4.26 n \rfloor$ clauses (the nearest integer to $4.26 n$, i.e., 51, 68, 85, 102 clauses at $n = 12, 16, 20, 24$). Every clause contained three literals on distinct variables. The formulas were drawn from the planted model. There, a uniformly random assignment a^* was fixed first, and each clause was constrained to be satisfied by it. Satisfiability was therefore guaranteed, and a reference solution was known. The ratio $m/n = 4.26$ was the value at which uniformly random

three-SAT instances were the hardest. *Algorithm:* The procedure of Algorithm 1, whose quantum part was simulated classically. The branch tree of Equation (6) was enumerated exactly, so that the two quantities entering the exponent were measured directly. No quantum hardware was used, and no classical SAT solver was run. *Parameters:* $n = 12$ to 38; radii r equal to the integer nearest ρn for $\rho \in \{0.20, 0.25, 0.30, \rho^* = 0.366, 0.42, 0.50\}$. (The ceiling $r = \lceil n/(1 + \sqrt{k}) \rceil$ of Algorithm 1 differed from it by at most one, which changed no exponent.) The cost measurements were taken at the optimal radius ρ^* . Each point used 2×10^6 sampled ball centers for p_B . It used 40 enumerated branch trees for $r \leq 9$, and 13–20 for $r \geq 10$. *Performance measures:* (i) p_B , the probability that a sampled ball contains a solution, estimated by Monte Carlo and compared with the exact binomial tail $2^{-n} \sum_{i \leq r} \binom{n}{i}$; (ii) γ , the marked amplitude of Equation (15), obtained by exact enumeration of the branch tree; (iii) the implied cost tJ , with $t = 1/p_B$ balls and $J = (\pi/4)/\gamma$ amplitude-amplification iterations, and its base $(tJ)^{1/n}$. The figure $(\pi/4)/\gamma$ is the iteration count of a fixed schedule; the exponential-search variant of Step 3 of Algorithm 1 differs from it by a constant factor, which leaves the base unchanged. *Reference values:* All comparisons were against the *analytic* worst-case exponents of the classical algorithms: Schönig’s $(2 - 2/k)^n$ [3], the Hamming-ball template $(2 - 2/(k + 1))^n$ [9] and the PPSZ-type bound 1.307^n for three-SAT [6–8]. They were not against measured running times of solvers, and no wall-clock advantage over an implementation is claimed. What follows is a verification of the two bounds on which the exponent rests, on the ensembles stated here.

The exponent of Equation (11) rests on two worst-case bounds: the probability p_B that a uniformly sampled ball contains a solution [Equation (10)], and the lower bound $k^{-r/2}$ on the amplitude of the constructed state on the marked subspace (Section 3.5). Both are checkable independently on small instances, and we did so here for $k = 3$ on the ensembles and with the parameters stated above. The probability p_B was measured by Monte Carlo sampling of the ball center, while the marked amplitude

$$\gamma = \left(\sum_{\ell} k^{-d_{\ell}} \right)^{1/2} \quad (15)$$

was obtained by exact enumeration of the branch tree of Equation (6); the sum runs over those leaves ℓ whose assignment satisfies every clause, and d_{ℓ} is the depth of ℓ . Equation (15) is precisely the amplitude of $|\psi_a\rangle$ on the marked subspace: by Equation (5) such a leaf carries the amplitude $k^{-d_{\ell}/2}$, and distinct leaves are recorded by distinct label configurations and are therefore mutually orthogonal, so that their amplitudes combine in quadrature. The enumeration also reproduced the structural statement of Section 3.5 directly: in every run reported below, a ball containing a solution produced a satisfying leaf of the enumerated tree.

Figure 2a confronts the measured p_B with Equation (10). Because $d(a, a^*)$ is a sum of n independent fair bits, the probability that a uniformly random center lies within distance r of a fixed solution is exactly the binomial tail $p_B = 2^{-n} \sum_{i \leq r} \binom{n}{i}$, which can be evaluated without sampling error; the figure shows both this exact value and the Monte Carlo estimate. The two agree at every point to within the statistical error of the sampler, which validates the sampling of the ball center on which the amplitude measurements rely, and every point lies inside the window delimited below by Equation (10) and above by the bound $|B(a, r)| \leq 2^{nh(\rho)}$ of Section 3.7. The plotted ratio tends toward the lower end of that window rather than toward unity: the entropy factor $2^{n(h(\rho)-1)}$ taken alone overestimates p_B by a root-of- n factor, and at the optimal radius the ratio to the lower bound of Equation (10) lies between 1.8 and 2.3 for every n studied, so that the bound is tight up to its root-of- n correction. At the other radii, the ratio to the lower bound runs from 1.3 at $\rho = 0.20$ to 4.6 at $\rho = 0.50$, the increase reflecting both the $\sqrt{n}$ growth of the correction $\sqrt{8n\rho(1-\rho)}$ of Equation (10) and the loosening of the entropy approximation $2^{n(h(\rho)-1)}$ as ρ approaches $1/2$.

Figure 2b shows the measured γ in units of the guaranteed bound $k^{-r/2}$. Every point satisfies $\gamma \geq k^{-r/2}$, as required, but the bound is far from tight: at the optimal radius, the ratio grows from 4.4 at $(n, r) = (12, 4)$ to 831 at $(38, 14)$, in proportion to $k^{r/2}$; the measured γ stays of order 0.1 to 0.5 rather than decaying, while the guaranteed bound $k^{-r/2}$ falls off exponentially. Two effects contribute. First, a leaf that satisfies the formula at depth $d < r$ carries $k^{-d/2}$ rather than $k^{-r/2}$, so every satisfying leaf reached before the last round raises γ above the bound. Second, and more importantly, the planted instances used here are not unique-solution instances: exact enumeration of the instances of class A of Table 4 gives a mean of 43.5, 61.8, 198.0 and 663.8 satisfying assignments at $n = 12, 16, 20$ and 24, respectively. This is far more than the $O(1)$ count that the estimate $2^n(1 - 2^{-k})^m$ predicts for a *uniform* random instance of the same density, because conditioning every clause on the planted solution keeps a whole cluster of assignments around it alive. (The estimate $2^n(1 - 2^{-k})^m$ applies to uniformly random instances and understates the count on the planted ensemble by more than an order of magnitude.) Thus, a ball of radius r typically contains several solutions instead of the single one assumed in the worst case. The measured γ therefore probes the typical instance, whereas $k^{-r/2}$ is a guarantee that must hold for the least favorable placement of a single solution, which is the situation the analysis of Section 3.5 is designed for. The enumeration visits k^r leaves per tree, which is why the number of trees per point falls with r as stated above. This is the main source of the residual scatter of the points at the largest n , and it also dominates the uncertainty of the last two entries of Table 5: repeating the measurement at $(38, 14)$ with a different number of trees moves γ from 0.22 to 0.38, that is, the ratio to the bound from 476 to 831.

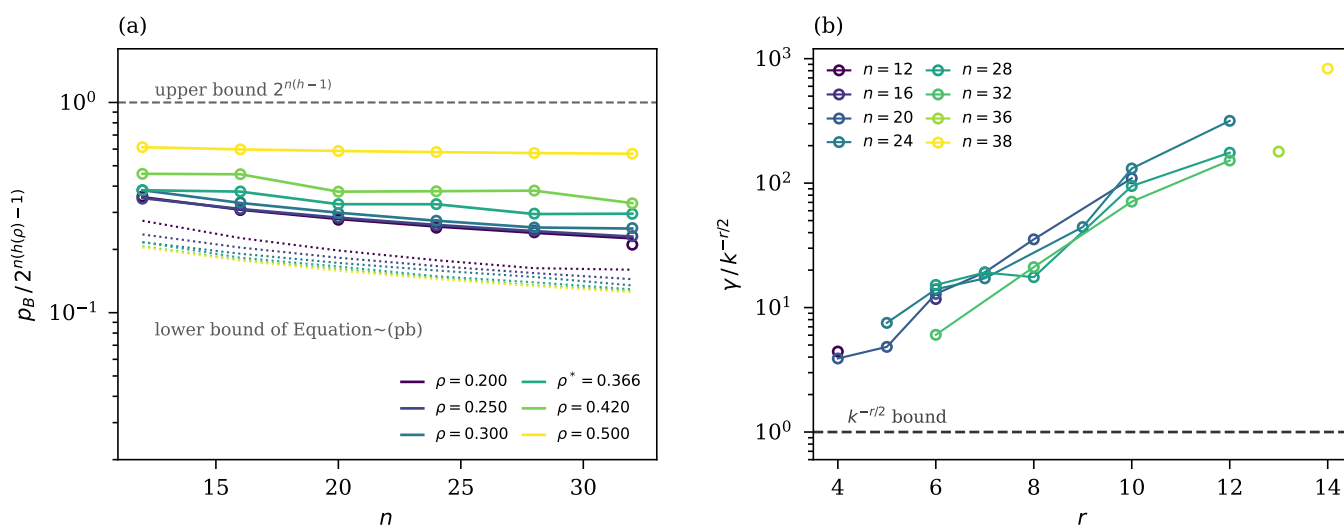


Figure 2. Verification of the two bounds entering the exponent, for $k = 3$. (a) The measured p_B in units of the entropy factor $2^{n(h(\rho)-1)}$, for $\rho = 0.20, 0.25, 0.30, \rho^* = 0.366, 0.42$ and 0.50 . Solid lines are the exact binomial tail $2^{-n} \sum_{i \leq r} \binom{n}{i}$ with r the integer nearest ρn , open symbols the Monte Carlo estimate at the same points, dotted lines the lower bound of Equation (10) and the dashed line its upper bound; every value lies inside the window. (b) The measured marked amplitude γ in units of the guaranteed bound $k^{-r/2}$, drawn as the dashed line at unity, for the radii of panel (a) and for the optimal radius ρ^* up to $n = 38$. The bound holds everywhere and is increasingly conservative as r grows.

Table 5 collects the corresponding costs at ρ^* , with $t = 1/p_B$ balls and $J = (\pi/4)/\gamma$ amplitude-amplification iterations per ball, and Figure 3 shows the total cost against n . The measured base $(tJ)^{1/n}$ decreases from 1.19 at $n = 12$ to 1.09 at $n = 38$, and a least-squares fit over $n \geq 20$ gives 1.075, below the asymptotic value $2 - 2/(1 + \sqrt{3}) = 1.268$ of Equation (11). This gap is not a claim of a better complexity. The measured quantities are

averages over planted instances, whereas Equation (11) bounds every instance, and over the range of n accessible here, the polynomial overhead alone, of order nmk elementary gates as in Equation (12), is already large enough to bridge the difference: the bound exceeds the measured cost by a factor growing from two at $n = 12$ to 290 at $n = 38$, whereas a polynomial overhead of order $n^{1.5}$ in the total cost is 230 at the largest n studied and exceeds that difference at every smaller n ; extrapolating a base from this window would therefore not be meaningful. What the numbers do establish is that both ingredients of the analysis are valid, and conservative in the typical case: p_B obeys Equation (10) including the root-of- n correction, the amplitude bound $\gamma \geq k^{-r/2}$ of Section 3.5 holds at every point, and the inner quantum search is far cheaper than the worst-case estimate $J \lesssim (\pi/4)k^{r/2}$ on which Equation (11) is built.

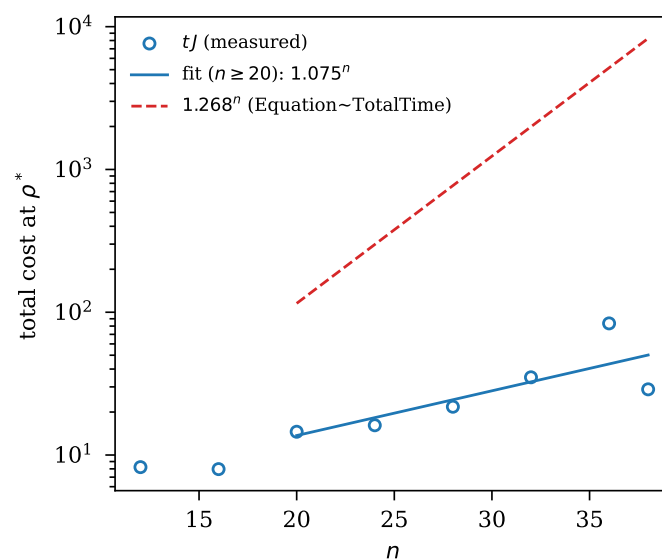


Figure 3. Total cost tJ at the optimal radius ρ^* against n , for $k = 3$. Symbols are the measured values of Table 5, the solid line a least-squares fit over $n \geq 20$ and the dashed line the asymptotic bound 1.268^n of Equation (11). The measured base lies below the bound because the bound uses the worst-case amplitude $k^{-r/2}$, and over this range of n , the polynomial overheads that Equation (11) discards are of the same order as the difference.

Instance Classes with Few Satisfying Assignments

The measurements above are averages over the planted ensemble. The solution counts just quoted show that this ensemble is far from the worst case the analysis is designed for: a ball typically contains many solutions, and each of them contributes to γ . From those numbers alone, one cannot decide whether the slack of the bound $k^{-r/2}$ comes from the analysis or from how easy the ensemble is. We therefore repeated the measurement on two further instance classes. They were chosen so that the number of satisfying assignments was small, and the situation approached the one the worst-case analysis assumed:

- **Class A** (reference): planted instances with $m = \lfloor 4.26n \rfloor$ clauses, the ensemble used in Table 5;
- **Class B**: planted instances with $m = \lceil 6n \rceil$ clauses, i.e., above the satisfiability threshold, where the cluster of solutions around the planted assignment is much smaller;
- **Class C**: uniform random instances with $m = \lceil 5n \rceil$ clauses, kept only if they are satisfiable and have at most eight solutions (rejection sampling; no solution is planted, and the number of solutions is counted exactly by enumeration).

For every class, the same quantities as in Table 5 were measured at the optimal radius ρ^* , with four instances and 24 enumerated branch trees per point. The class-A entries therefore differed slightly from the corresponding rows of Table 5, which used more trees

per point. In addition, the exact enumeration of the solution set restricted class C to $n \leq 24$. The same range was used for the other two classes, so that the three could be compared row by row. Table 4 and Figure 4 summarize the outcome.

Table 4. Instance classes with few satisfying assignments, $k = 3$ at the optimal radius $\rho^* = 1/(1 + \sqrt{k})$. N_{sol} is the number of satisfying assignments of the instance, counted by exact enumeration and averaged over the four instances of the row; p_B is the exact binomial tail of Equation (10) for one fixed solution; γ is the marked amplitude, averaged over the enumerated balls that contain a solution; $\gamma_{\min} = k^{-r/2}$ is the guaranteed bound of Section 3.5; and $(tJ)^{1/n}$ is the base of the total cost tJ with $t = 1/p_B$ balls of $J = (\pi/4)/\gamma$ amplitude-amplification iterations each, to be compared with $2 - 2/(1 + \sqrt{3}) = 1.268$. Four instances and 24 branch trees per row. The measured hit rate—the fraction of sampled centers whose ball contains *some* solution—is not tabulated separately; its ratio to p_B , which tends to 1 as the number of solutions tends to 1, is reported in the text below.

Class	n	r	N_{sol}	p_B	γ	$\gamma/\gamma_{\min}$	$(tJ)^{1/n}$
A: planted, $m = \lfloor 4.26n \rfloor$	12	4	43.5	0.194	0.362	3.3	1.223
A: planted, $m = \lfloor 4.26n \rfloor$	16	6	61.8	0.227	0.230	6.2	1.185
A: planted, $m = \lfloor 4.26n \rfloor$	20	7	198.0	0.132	0.164	7.7	1.197
A: planted, $m = \lfloor 4.26n \rfloor$	24	9	663.8	0.154	0.153	21.5	1.157
B: planted, $m = \lfloor 6n \rfloor$	12	4	10.2	0.194	0.256	2.3	1.259
B: planted, $m = \lfloor 6n \rfloor$	16	6	46.2	0.227	0.231	6.2	1.184
B: planted, $m = \lfloor 6n \rfloor$	20	7	60.0	0.132	0.141	6.6	1.206
B: planted, $m = \lfloor 6n \rfloor$	24	9	74.2	0.154	0.082	11.5	1.188
C: random, $m = \lfloor 5n \rfloor$, SAT-filtered	12	4	3.5	0.194	0.178	1.6	1.297
C: random, $m = \lfloor 5n \rfloor$, SAT-filtered	16	6	3.2	0.227	0.134	3.6	1.225
C: random, $m = \lfloor 5n \rfloor$, SAT-filtered	20	7	5.0	0.132	0.070	3.3	1.249
C: random, $m = \lfloor 5n \rfloor$, SAT-filtered	24	9	2.5	0.154	0.051	7.2	1.212

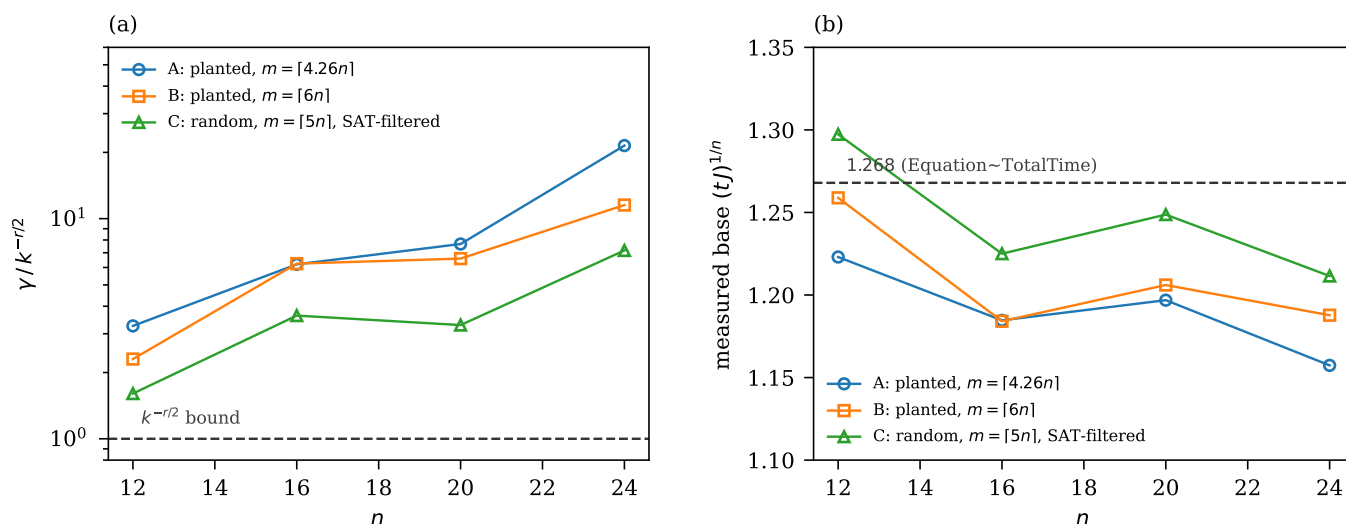


Figure 4. The two bounds of the exponent on the three instance classes of Table 4, for $k = 3$ at the optimal radius ρ^* . (a) The measured marked amplitude γ in units of the guaranteed bound $k^{-r/2}$ (dashed line at unity). On the planted classes A and B the bound is exceeded by a factor that grows with n . On class C, which has only a few satisfying assignments, the ratio stays within one order of magnitude of unity. (b) The base $(tJ)^{1/n}$ of the measured total cost, against the asymptotic bound $2 - 2/(1 + \sqrt{3}) = 1.268$ of Equation (11) (dashed line): the planted classes lie systematically below the bound, whereas class C lies on both sides of it.

Table 5. Numerical verification at the optimal radius $\rho^* = 1/(1 + \sqrt{k})$ for $k = 3$. p_B is the exact binomial tail $|B(a, r)|/2^n$; γ is the marked amplitude of Equation (15) measured by exact enumeration of the branch tree; and $(tJ)^{1/n}$ is the base of the total cost tJ , with $t = 1/p_B$ classically repeated balls of $J = (\pi/4)/\gamma$ amplitude-amplification iterations each, to be compared with $2 - 2/(1 + \sqrt{3}) = 1.268$ of Equation (11). The guaranteed lower bound is $\gamma_{\min} = k^{-r/2}$.

n	r	p_B	γ	$\gamma/\gamma_{\min}$	$(tJ)^{1/n}$
12	4	0.194	0.493	4.4	1.192
16	6	0.227	0.434	11.7	1.139
20	7	0.132	0.410	19.2	1.143
24	9	0.154	0.316	44.4	1.123
28	10	0.0925	0.390	94.7	1.116
32	12	0.108	0.208	152	1.118
36	13	0.0662	0.142	179	1.131
38	14	0.0717	0.380	831	1.093

Two conclusions follow. First, the bound $\gamma \geq k^{-r/2}$ is not uniformly loose: its slack is a property of the ensemble rather than of the analysis. On the planted classes, the ratio $\gamma/k^{-r/2}$ grows with n , and reaches 21.5 (class A) and 11.5 (class B) at $n = 24$. On class C, where an instance has between one and a handful of solutions, the ratio stays between 1.6 and 7.2, an order of magnitude smaller at the same n . The measured base $(tJ)^{1/n}$ moves correspondingly. For classes A and B it lies clearly below the asymptotic bound (1.16–1.26). For class C it lies on both sides of the bound (1.21–1.30), and the bound 1.268 of Equation (11) falls inside that interval. On instances close to the worst case, the bound is thus attained, up to the constant $\pi/4$ and the polynomial factors discarded by Equation (11). The sub-bound bases of Table 5 and Figure 3, by contrast, are a property of the planted ensemble. The base measured on planted instances must not be extrapolated into a complexity claim. That is why the measured bases are not put forward as an exponent anywhere in this paper. Second, the classical factor behaves as assumed: the ratio of the measured hit rate to p_B falls from 6.1 (class A, 663.8 solutions at $n = 24$) to 1.5 (class C, 2.5 solutions at $n = 24$). It moves towards the value of one that the analysis postulates for an instance with a single solution. Thus, $t = 1/p_B$ is the correct repetition count precisely in the regime where the bound is proved. However, the comparison is limited in range: exact enumeration of the solution set and of the branch tree restricts it to $n \leq 24$ with four instances per point. The numbers therefore indicate a trend rather than an asymptotic behavior.

5. Conclusions

The k -SAT problem is NP-complete and admits no known efficient classical solution. The best known classical algorithms for three-SAT run in time $O(1.307^n)$ [6–8]. In this paper, we proposed a hybrid quantum–classical algorithm that integrates Grover’s search as a local subroutine within a randomized framework.

Assuming the given k -SAT instance is satisfiable, the proposed hybrid algorithm requires $O\left(2^{n(1-h(\rho))} \log(1/\delta)\right)$ classically repeated iterations of the outer loop (up to polynomial factors) to find a satisfying assignment with probability at least $1 - \delta$, for any constant $\delta > 0$. Each round uses its own clause registers of $O(mk)$ qubits together with a label register of $O(\log k + L)$ qubits; these are retained rather than uncomputed and therefore accumulate over the r rounds, whereas the oracle uses its own clause and formula registers, which are uncomputed and reused. Both counts are polynomial in n and immaterial for the exponential scaling. In each iteration, Grover’s algorithm searches within a Hamming ball of radius $r = n/(1 + \sqrt{k})$ centered at a uniformly random assignment. As detailed in Section 3.3, we constructed a problem-dependent superposition state that encodes all candidate solutions within this Hamming ball based on the clause structure.

Subsequently, Section 3.4 presented an explicit, gate-level implementation of the oracle that identifies solution states within that superposition, built from the elementary gates QNOT, QAND, QOR, and CNOT.

The overall runtime of our algorithm is $\left(2 - \frac{2}{1+\sqrt{k}}\right)^n$, which improves upon the classical local-search bounds for k -SAT and, at $k = 3$, upon the best known classical algorithm (1.307^n). The gain over the classical Hamming-ball template comes from the inner quantum search alone: because the center of each ball is measured, the outer sampling of balls remains a classical repetition and contributes the factor $2^{n(1-h(\rho))}$ rather than its square root, and balancing this factor against the $k^{pn/2}$ cost of amplitude amplification shifts the optimal radius from $\rho = 1/(k+1)$ to $\rho^* = 1/(1+\sqrt{k})$ (Table 2), the resulting base $(2 - 2/(1+\sqrt{k}))^n$ lying below both classical local-search bounds for every $k \geq 3$ (Table 3). The explicit constructions of Sections 3.3 and 3.4 fix the polynomial overhead at a factor $O(nmk)$ elementary gates, so that the total cost is that of Equations (11) and (12), and no unaccounted-for oracle is left in the complexity. Both bounds entering this exponent were verified numerically on small planted instances (Section 4). This exponent did not improve upon the best known quantum algorithms, obtained by accelerating stronger classical solvers [6,8,27,28] (Table 1). The main contribution of this work is therefore an explicit, gate-level realization of a problem-structured quantum local search rather than a new asymptotic record.

Limitations. The following points delimit the scope of the result. (i) *The exponent is not a record.* As Table 1 shows, black-box amplitude amplification applied to a stronger classical solver gives smaller quantum exponents, and no improvement over those is claimed. The gain is over the classical local-search bounds. It is a gain in the base of a search procedure, the algorithm being a search and not a decision procedure (Section 3.6). (ii) *The hardware assumptions are strong.* The running time assumes a device that holds $n + O(rmk + n \log n + mk)$ qubits. It also assumes that the $O(nmk k^{\rho^*n/2})$ gates of a single run are executed coherently, with a per-gate error rate obeying Equation (14). With a fixed physical error rate, the window of n in which the bound is meaningful is small, because it grows only logarithmically in $1/p_g$ (Section 3.8). (iii) *The construction is expensive in qubits.* Neither the branch labels nor the clause registers of a round can be uncomputed. The construction therefore retains a copy of each of them per round, and the width grows as $O(rmk)$ with the radius. This is polynomial, but it makes the qubit count the binding constraint at small n . (iv) *The verification is a simulation, not a run.* Section 4 measures the two bounds that carry the exponent by exact enumeration of the branch tree. It is limited to $n \leq 38$ ($n \leq 24$ for the classes with few solutions). It uses no quantum hardware, and it makes no statement about constant factors of a real implementation. (v) *No comparison against solver implementations is made.* The classical reference values are analytic worst-case exponents. Moreover, measured running times of solvers, and the effect of instance structure, are outside the scope of this paper.

Future work. The most immediate direction is to remove the measurement of Step 1 of Algorithm 1, so that the outer loop over balls can be amplified as well. A fully coherent variant would replace the factor $2^{n(1-h(\rho))}$ by its square root. With the radius re-optimized at $\rho = 1/(k+1)$, it would give the exponent $(2 - 2/(k+1))^{n/2}$, that is, 1.225^n for $k = 3$. However, whether the superposition over ball centers can be prepared at polynomial cost while keeping the construction explicit is an open question. The same gate-level treatment could be applied to stronger classical templates, in particular, to the PPSZ-type algorithms of Section 1, where the competitive exponents are to be found. It would show whether an explicit construction can match the black-box bounds. On the resource side, an uncomputation scheme for the branch labels, or a compressed label encoding, would remove the $O(rmk)$ growth of the width noted in (iii). On the robustness side, the error

budget of Equation (14) suggests studying truncated variants, in which the radius r is capped by the available coherence time. The aim would be to obtain the best exponent achievable on a device with a given error rate. Finally, the numerical verification could be carried to larger n by sampling the branch tree instead of enumerating it. This would allow the trend of Table 4, the tightening of the amplitude bound as the number of solutions decreases, to be followed into the asymptotic regime. A comprehensive recent review of the opportunities and challenges of quantum optimization, including the benchmarking needed to assess quantum advantage against classical methods, is given in Ref. [30]. A recent alternative route to quantum k -SAT is the measurement-driven approach of Ref. [31], in which generalized clause measurements implement a Zeno-dragging dynamics; that work likewise benchmarks against the classical Schöningh exponent and distinguishes instances having a unique, multiple, or no solution, which is closely related to the solution-count classes studied here. Grover-type search has meanwhile been applied to further combinatorial problems, for instance the vehicle routing problem [32]. A complementary recent development is the space-efficient formulation of Ref. [33], in which a structural reduction of the clause set lowers the effective search dimension before the instance is handed to quantum annealing or QAOA; this pursues the same goal of exploiting problem structure so as to reduce the quantum resources required, although by a mechanism different from the one used here.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/electronics15204867/s1>, Script S1: Python scripts used for the numerical verification of Section 4, together with the data underlying Figures 2 and 3 and Table 5; the script `exp_few_sat.py` and the data underlying Figure 4 and Table 4 of Section 4.

Funding: This research was funded by the Hainan Natural Science Foundation of China, grant numbers 621RC741 and 723QN237.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The Python scripts and the numerical data reported in Section 4, which also underlie Figures 2 and 3 and Table 5, are provided as Supplementary Materials (Script S1). All remaining data are contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A. Implementation of F_{c_i}

The implementation of F_{c_i} can be divided into two steps. The first step prepares the state $W|0\rangle^{\otimes q}$, i.e., a uniform superposition over k computational basis states in the label register $|s\rangle$, which is initialized to $|0\rangle^{\otimes q}$. The second step conditionally flips each qubit of $|a\rangle$ controlled by the corresponding component of $|s\rangle$ [Equation (4)]; the register $|s\rangle$ then records which branch was taken and is retained rather than uncomputed [Equation (5)]. Three cases must be considered based on the value of k : (1) $k = 2^q$, (2) $k = 2^q - 1$, and (3) $2^{q-1} < k < 2^q - 1$, where q denotes the number of qubits in $|s\rangle$.

In the first case ($k = 2^q$), the uniform superposition over all k basis states of $|s\rangle$ is prepared by applying Hadamard gates to all q qubits, i.e., $W = H^{\otimes q}$ in Equation (4). The second step of F_{c_i} then employs Pauli-X gates conditioned on $|s\rangle$ via multi-controlled X gates; for $q = 1, 2$, these are QNOT and QAND (Toffoli) gates, while for $q \geq 3$, each of them decomposes into $O(q)$ Toffoli gates together with $O(q)$ ancillas, which leaves the polynomial overhead of Section 3.7 unchanged. Taking $k = 4$ as an example, suppose for illustration that $|a\rangle$ is in the basis state $|0000\rangle$ (in the algorithm, $|a\rangle$ holds the assignment

being expanded). Hadamard gates are first applied to the two auxiliary qubits to generate the uniform superposition state:

$$|s\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

Subsequently, each of the four qubits in $|a\rangle$ is flipped conditioned on the corresponding basis state of $|s\rangle$. The quantum circuit for these controlled-flip operations is depicted in Figure A1. After these operations, $|s\rangle$ and $|a\rangle$ become entangled as follows:

$$|s\rangle|a\rangle = \frac{1}{2}(|00\rangle|1000\rangle + |01\rangle|0100\rangle + |10\rangle|0010\rangle + |11\rangle|0001\rangle).$$

In the second case, a uniform superposition over exactly k computational basis states of $|s\rangle$ cannot be prepared by directly applying Hadamard gates. However, this state can be approximately obtained through an iterative procedure: Hadamard gates are repeatedly applied to the all-zero component of $|s\rangle$, controlled by an additional auxiliary qubit register $|w\rangle$ initialized to $|0\rangle^{\otimes L}$, where L denotes the number of iterations of the procedure.

Taking $k = 3$ as an example, $|s\rangle$ requires $q = 2$ qubits to encode k computational basis states. We first prepare the full uniform superposition over all computational basis states of $|s\rangle$ by applying Hadamard gates to every qubit. The resulting state $|s\rangle$ can be decomposed as:

$$\begin{aligned} |s\rangle &= |\phi^0\rangle + |\tilde{\phi}^0\rangle \\ &= \frac{1}{2}(|01\rangle + |10\rangle + |11\rangle) + \frac{1}{2}|00\rangle, \end{aligned}$$

where $|\phi^0\rangle$ spans the desired subspace, and $|\tilde{\phi}^0\rangle$ corresponds to the unwanted all-zero component. Applying the operation depicted in Figure A2 to $|s\rangle$ and $|w\rangle$ affects only the $|\tilde{\phi}^0\rangle$ component, transforming it into $|\phi^1\rangle + |\tilde{\phi}^1\rangle$, where

$$|\phi^1\rangle = \frac{1}{2}|\phi^0\rangle|10\dots 0\rangle, \quad |\tilde{\phi}^1\rangle = \frac{1}{2}|\tilde{\phi}^0\rangle|10\dots 0\rangle.$$

In general, the i th application of the operation on $|s\rangle$ and $|w\rangle$ affects only $|\tilde{\phi}^{i-1}\rangle$, transforming it into $|\phi^i\rangle + |\tilde{\phi}^i\rangle$, where

$$\begin{aligned} |\phi^i\rangle &= 2^{-i}|\phi^0\rangle|1^i 0^{L-i}\rangle, \\ |\tilde{\phi}^i\rangle &= 2^{-i}|\tilde{\phi}^0\rangle|1^i 0^{L-i}\rangle, \end{aligned}$$

and the first i qubits of $|w\rangle$ are in the state $|1\rangle$. After L applications of this operation, the joint state of $|s\rangle$ and $|w\rangle$ can be expressed as:

$$|s, w\rangle = \sum_{i=0}^L |\phi^i\rangle + |\tilde{\phi}^L\rangle.$$

The amplitude of $|\tilde{\phi}^L\rangle$ is $2^{-(L+1)}$, so the residual weight of the unwanted all-zero component is $4^{-(L+1)}$, i.e., exponentially small in L . Within each branch i , the k desired basis states carry equal amplitudes $2^{-i-q/2}$ (i.e., $2^{-i}/2$ for the present case $k = 3$, $q = 2$), so the marginal distribution over the desired basis states $\{|01\rangle, |10\rangle, |11\rangle\}$ is uniform up to the residual weight $4^{-(L+1)}$ that the state places on the unwanted component. Each application of the operation to the all-zero component of $|s\rangle$ requires one additional auxiliary qubit $|w_i\rangle$ initialized to $|0\rangle$, so the register $|w\rangle$ comprises L qubits. Like $|s\rangle$, the register $|w\rangle$ is retained as part of the branch label of its round [Equations (4) and (5)] and does not return to $|0\rangle^{\otimes L}$, so that each

of the r rounds uses its own copy; choosing $L = (\frac{1}{2} + \varepsilon) \log_2 n$ for a fixed $\varepsilon > 0$ keeps the residual weight $4^{-(L+1)}$ of the unwanted all-zero component small enough that the total error accumulated over the $r = \rho n$ rounds, of order $O(r4^{-(L+1)}) = O(n^{-2\varepsilon})$, is $o(1)$.

The third case ($2^{q-1} < k < 2^q - 1$) requires a uniform superposition over a specified subset S of k of the 2^q computational basis states, which can be prepared exactly by standard state-preparation techniques. Starting from the equal superposition $H^{\otimes q}|0\rangle = 2^{-q/2} \sum_z |z\rangle$, the indicator function of S is computed into an auxiliary qubit and one step of amplitude amplification [28] rotates the marked component into the target state $|S\rangle = k^{-1/2} \sum_{z \in S} |z\rangle$. Since the subset occupies a fraction $k/2^q > 1/2$ of the search space, $O(1)$ amplification iterations suffice, and the indicator of S is evaluated with $O(q + k)$ elementary gates from its classical description. Read as a strict statement, the amplification route just described prepares $|S\rangle$ only up to a deviation ε , and so does the alternative route. Either way, the deviation can be made an arbitrarily small ε , at a cost of $\text{poly}(q, \log(1/\varepsilon))$ gates. This holds whether one uses exact amplitude amplification [28] or prepares the uniform superposition over the first k labels recursively and maps it onto S with a classically controlled permutation. Taking ε small enough for the residual error to fall inside the budget quantified in Section 3.8, the choice between these two standard routes therefore leaves the exponential scaling of Equation (11) unchanged.

Conditional structure. The selection that defines each factor of Equation (3)—clauses $c_1, \dots, c_{i-1}$ true and clause c_i false—is a computational-basis condition on the clause registers $|\alpha_{j,k-1}\rangle$ and is therefore realized by a multiply-controlled gate: the m control patterns are computed simultaneously into a flag register by a reversible cascade of $O(m)$ QAND gates with $O(m)$ work ancillas, F_{c_i} is then applied controlled on the i th flag, and the flags are uncomputed; if no flag is set, i.e., if every clause is satisfied, no expansion is applied. Because F_{c_i} changes only the assignment and label registers, the clause registers are untouched while the expansion acts, so the flags remain valid and their uncomputation is exact, returning all work ancillas to $|0\rangle$ for reuse in the next round. The conditional structure thus adds $O(m)$ gates per round to the $O(mk)$ gates of the clause evaluation, which leaves the polynomial overhead of Section 3.7 unchanged.

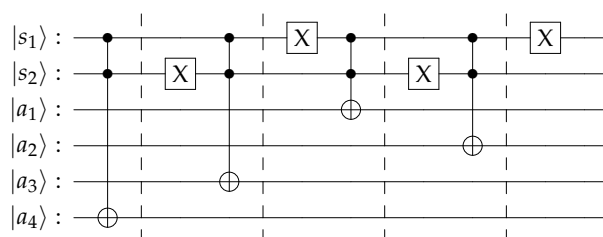


Figure A1. Entanglement between computational basis components of $|s\rangle$ and $|a\rangle$ via four Toffoli gates. The first gate entangles the $|11\rangle$ component of $|s\rangle$ with the $|0001\rangle$ component of $|a\rangle$; the second, third, and fourth gates analogously entangle $|10\rangle$, $|00\rangle$, and $|01\rangle$ of $|s\rangle$ with $|0010\rangle$, $|1000\rangle$, and $|0100\rangle$ of $|a\rangle$, respectively.

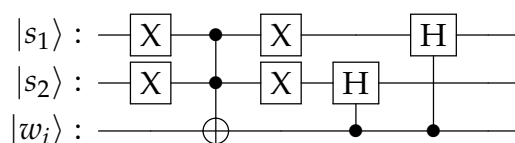


Figure A2. Circuit decomposition of the expansion operation on the all-zero component of $|s\rangle$ using an ancilla qubit $|w_i\rangle$.

References

1. Karp, R.M. Reducibility Among Combinatorial Problems. In *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*; Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 219–241. [\[CrossRef\]](#)
2. Paturi, R.; Pudlák, P.; Zane, F. Satisfiability Coding Lemma. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS '97)*; IEEE: New York, NY, USA, 1997; pp. 566–574. [\[CrossRef\]](#)
3. Schöning, U. A probabilistic algorithm for k -SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99)*; IEEE: New York, NY, USA, 1999; pp. 410–414. [\[CrossRef\]](#)
4. Brueggemann, T.; Kern, W. An improved deterministic local search algorithm for 3-SAT. *Theor. Comput. Sci.* **2004**, *329*, 303–313. [\[CrossRef\]](#)
5. Hofmeister, T.; Schöning, U.; Schuler, R.; Watanabe, O. A Probabilistic 3—SAT Algorithm Further Improved. In *STACS 2002, Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes—Juan les Pins, France, 14–16 March 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 192–202. [\[CrossRef\]](#)
6. Paturi, R.; Pudlák, P.; Saks, M.E.; Zane, F. An improved exponential-time algorithm for k -SAT. *J. ACM* **2005**, *52*, 337–364. [\[CrossRef\]](#)
7. Hertli, T. 3-SAT Faster and Simpler—Unique-SAT Bounds for PPSZ Hold in General. *SIAM J. Comput.* **2014**, *43*, 718–729. [\[CrossRef\]](#)
8. Hansen, T.D.; Kaplan, H.; Zamir, O.; Zwick, U. Faster k -SAT algorithms using biased-PPSZ. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC '19)*, Phoenix, AZ, USA, 23–26 June 2019; pp. 578–589. [\[CrossRef\]](#)
9. Dantsin, E.; Goerdt, A.; Hirsch, E.A.; Kannan, R.; Kleinberg, J.; Papadimitriou, C.; Raghavan, P.; Schöning, U. A deterministic $(2 - 2/(k+1))^n$ algorithm for k -SAT based on local search. *Theor. Comput. Sci.* **2002**, *289*, 69–83. [\[CrossRef\]](#)
10. Moser, R.A.; Scheder, D. A Full Derandomization of Schöning's k -SAT Algorithm. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC '11)*, San Jose, CA, USA, 6–8 June 2011; pp. 245–252. [\[CrossRef\]](#)
11. Scheder, D. Guided Search and a Faster Deterministic Algorithm for 3-SAT. In *LATIN 2008: Theoretical Informatics*; Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 60–71. [\[CrossRef\]](#)
12. Makino, K.; Tamaki, S.; Yamamoto, M. Derandomizing the HSSW Algorithm for 3-SAT. *Algorithmica* **2013**, *67*, 112–124. [\[CrossRef\]](#)
13. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [\[CrossRef\]](#)
14. Grover, L.K. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Phys. Rev. Lett.* **1997**, *79*, 325–328. [\[CrossRef\]](#)
15. Shenvi, N.; Kempe, J.; Whaley, K.B. Quantum random-walk search algorithm. *Phys. Rev. A* **2003**, *67*, 052307. [\[CrossRef\]](#)
16. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **2015**, *56*, 172–185. [\[CrossRef\]](#)
17. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [\[CrossRef\]](#)
18. Gemeinhart, F.; Garmendia, A.; Wimmer, M.; Weder, B.; Leymann, F. Quantum Combinatorial Optimization in the NISQ Era: A Systematic Mapping Study. *ACM Comput. Surv.* **2024**, *56*, 1–36. [\[CrossRef\]](#)
19. Ge, X.; Wu, R.B.; Rabitz, H. The optimization landscape of hybrid quantum-classical algorithms: From quantum control to NISQ applications. *Annu. Rev. Control* **2022**, *54*, 314–323. [\[CrossRef\]](#)
20. Nardin, A.; D'Andreagiovanni, F. A Quantum-Inspired Ant Colony Optimization Algorithm for Parking Lot Rental to Shared E-Scooter Services. *Algorithms* **2024**, *17*, 80. [\[CrossRef\]](#)
21. Jiang, J.R.; Wang, Y.J. Using a Simplified Quantum Counter to Implement Quantum Circuits Based on Grover's Algorithm to Tackle the Exact Cover Problem. *Mathematics* **2025**, *13*, 90. [\[CrossRef\]](#)
22. Jiang, J.R. Dicke State Quantum Search for Solving the Vertex Cover Problem. *Mathematics* **2025**, *13*, 3005. [\[CrossRef\]](#)
23. Mikes, C.; Gutman, D.H.; Howle, V.E. Quantum-Enhanced Generalized Pattern Search Optimization. *Quantum Rep.* **2024**, *6*, 509–521. [\[CrossRef\]](#)
24. Ablayev, F.; Salikhova, N.; Ablayev, M. Hybrid Classical–Quantum Text Search Based on Hashing. *Mathematics* **2024**, *12*, 1858. [\[CrossRef\]](#)
25. Pratibha; Mahmud, N. A Reconfigurable Framework for Hybrid Quantum–Classical Computing. *Algorithms* **2025**, *18*, 271. [\[CrossRef\]](#)
26. Dunjko, V.; Ge, Y.; Cirac, J.I. Computational Speedups Using Small Quantum Devices. *Phys. Rev. Lett.* **2018**, *121*, 250501. [\[CrossRef\]](#)
27. Ambainis, A. Quantum search algorithms. *ACM SIGACT News* **2004**, *35*, 22–35. [\[CrossRef\]](#)
28. Brassard, G.; Høyer, P.; Mosca, M.; Tapp, A. Quantum amplitude amplification and estimation. In *Quantum Computation and Information*; Contemporary Mathematics; American Mathematical Society: Providence, RI, USA, 2002; Volume 305, pp. 53–74. [\[CrossRef\]](#)
29. Brehm, M.; Weggemans, J. Assessing fault-tolerant quantum advantage for k -SAT with structure. *Quantum* **2026**, *10*, 1975. [\[CrossRef\]](#)
30. Abbas, A.; Ambainis, A.; Augustino, B.; Bärttschi, A.; Buhrman, H.; Coffrin, C.; Cortiana, G.; Dunjko, V.; Egger, D.J.; Elmegreen, B.G.; et al. Challenges and opportunities in quantum optimization. *Nat. Rev. Phys.* **2024**, *6*, 718–735. [\[CrossRef\]](#)
31. Zhang, Y.; Lewalle, P.; Whaley, K.B. Solving k -SAT problems with generalized quantum measurement. *npj Quantum Inf.* **2025**, *11*, 170. [\[CrossRef\]](#)

32. Liu, L.; Qian, L.; Wu, X.Y.; Fan, C.R.; Zhang, L.F.; Cai, D.B.; Lu, H.J.; Wang, T.J.; Wang, C. Solving vehicle routing problem using Grover adaptive search algorithm. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 9682–9692. [[CrossRef](#)]
33. Lu, Q.; Wei, S.; Zeng, J.; Yang, W.; Zheng, M.; Chang, P.; Ruan, D.; Wang, Z.; Long, G. Space-efficient quantum solvers for Boolean satisfiability problems via structural reduction. *Phys. Rev. A* **2026**, *accepted*. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.