LIMITATIONS OF QUANTUM COUNTING, NESTED QUANTUM SEARCH, AND AMPLITUDE AMPLIFICATION AND THEIR POTENTIAL TO SOLVE DISCRETE OPTIMIZATION PROBLEMS

Stefan Creemers, Luis Fernando Pérez Armas

REPRINT | 3338

UCLouvain

CORE | LIDAM Louvain Institute of Data Analysis and Modeling in economics and statistics

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Article submitted to journal

Subject Areas:
Quantum Optimization, Quantum Decision Making

Keywords:
Grover's algorithm, Quantum Counting, Quantum Phase Estimation, Nested Quantum Search, Amplitude Amplification

Author for correspondence:
Stefan Creemers
e-mail:
stefan.creemers@uclouvain.be

Limitations of Quantum Counting, Nested Quantum Search, and Amplitude Amplification and Their Potential to Solve Discrete Optimization Problems

Stefan Creemers[1], Luis Fernando Pérez Armas[2]

[1]Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), Voie du Roman Pays 34, B-1348 Louvain-la-Neuve, Belgium
[2]Operations Management, IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM - Lille Economie Management, 3 rue de la digue, Lille, F-59800, Nord, France

Several quantum algorithms have been proposed for solving discrete optimization problems. In this paper, we focus on the quantum counting algorithm of Brassard et al. (1998), the nested quantum search algorithm of Cerf et al. (2000), and amplitude amplification. We discuss potential limitations of these quantum algorithms and investigate how they can be used to solve discrete optimization problems. Our goal is to provide a set of practical guidelines that can help researchers to effectively implement Grover-based quantum algorithms.

THE ROYAL SOCIETY PUBLISHING

1. Introduction

In this paper, we discuss the limitations of a number of well-known quantum algorithms and assess their potential to solve discrete optimization problems. Discrete optimization problems are mentioned as one of the four use cases of quantum computing by McKinsey (2021). For a recent discussion on the challenges and opportunities of quantum optimization, please refer to Abbas et al. (2024). For recent applications of quantum computing in the field of discrete optimization, refer to Bochkarev et al. (2024) and Creemers and Pérez Armas (2025). Among quantum algorithms, Grover-based algorithms are of particular interest as the quadratic speedup that is achieved by Grover's algorithm (when compared to a classical search algorithm) has applications in almost any industry (McKinsey, 2023). Therefore, in what follows, we focus on three well-known Grover-based algorithms and investigate their potential to solve discrete optimization problems. Note, however, that there are several other (Grover-based) quantum algorithms that deserve further investigation. For instance, the quantum algorithm for finding the minimum (Dürr and Høyer, 1999) is of particular interest.

First, we discuss the quantum counting algorithm of Brassard et al. (1998; hereafter referred to as QCB). The goal of a quantum counting algorithm is to approximate the number of valid solutions (denoted m) in a set of solutions (for simplicity, we assume there are 2^n solutions). An (accurate) approximation of m is particularly useful when applying Grover's algorithm: using m, we can determine the number of iterations (required by Grover's algorithm) that maximizes the probability to measure any of the m valid solutions (i.e., we can avoid the "soufflé problem"). In addition, quantum counting algorithms can also be used to evaluate whether a solution exists (i.e., to approximate $P(m > 0)$). In what follows, however, we show that, rather than using QCB to solve discrete optimization problems, it may be an option as well to use a simple Grover-based approach such as GUM (see Appendix (a) for a discussion on GUM).

Next, we discuss the nested quantum search algorithm of Cerf et al. (2000; hereafter referred to as NQC). NQC is the quantum equivalent of a classical nested search algorithm, and allows to find a valid solution in a set of 2^n solutions using only $O(\sqrt{2^{\gamma n}})$ Grover iterations, where γ is some number less than or equal to 1 that depends on the structure of the problem. Even though we see nested quantum search as a mean to speed up a classical nested search algorithm, we show (by means of an example) that it cannot be used to solve discrete optimization problems faster in general.

Last, but not least, we also investigate amplitude amplification. Amplitude amplification is a generalization of Grover's algorithm that tries to reduce the number of iterations (and hence operations) required to measure a valid solution. In principle, amplitude amplification allows to design more complex algorithms that allow an asymptotic speedup when compared to a textbook implementation of Grover's algorithm. Whereas Grover's algorithm starts with a "uniform superposition" of qubits (i.e., a superposition where each qubit has an equal probability to collapse into either $|0\rangle$ or $|1\rangle$; a superposition where every solution has equal probability of being measured) and requires $\pi 4^{-1}\sqrt{m^{-1}2^n}$ iterations, amplitude amplification uses a superposition that allows to measure a valid solution using (far) less iterations. This is illustrated in Figure 1 that illustrates the uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition. A good superposition uses information of the optimization problem in order to reduce the number of iterations required to measure a valid solution. However, a number of experiments show that finding such a good superposition may not be that easy.

The contribution of this article can be summarized as follows: (1) we illustrate how existing quantum algorithms can be used to solve discrete optimization problems, (2) we provide practical guidelines for researchers who effectively want to implement quantum algorithms, and (3) we show that simple Grover-based algorithms such as GUM have reasonable performance when solving discrete optimization problems (when compared to the asymptotic runtime bounds of Grover's unstructured search). The goal of this article is to help build a deeper understanding of quantum algorithms and to (further) open up the field of quantum computing. To do so, we adopt a hands-on approach that uses numerical illustrations and intuitive discussions.

In what follows, Sections 2 and 3 discuss QCB and NQC. Section 4 discusses amplitude amplification and Section 5 concludes. Note that, we also illustrate the aforementioned quantum algorithms by solving the example binary knapsack problem that was used in Creemers and Pérez Armas (2025). The parameters of this example problem are provided in Table 1, where w_i and v_i denote the weight and value of item i, and W

Figure 1: Comparison of a uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition when measuring a valid solution in a system that has $m = 1$ valid solution and n qubits.

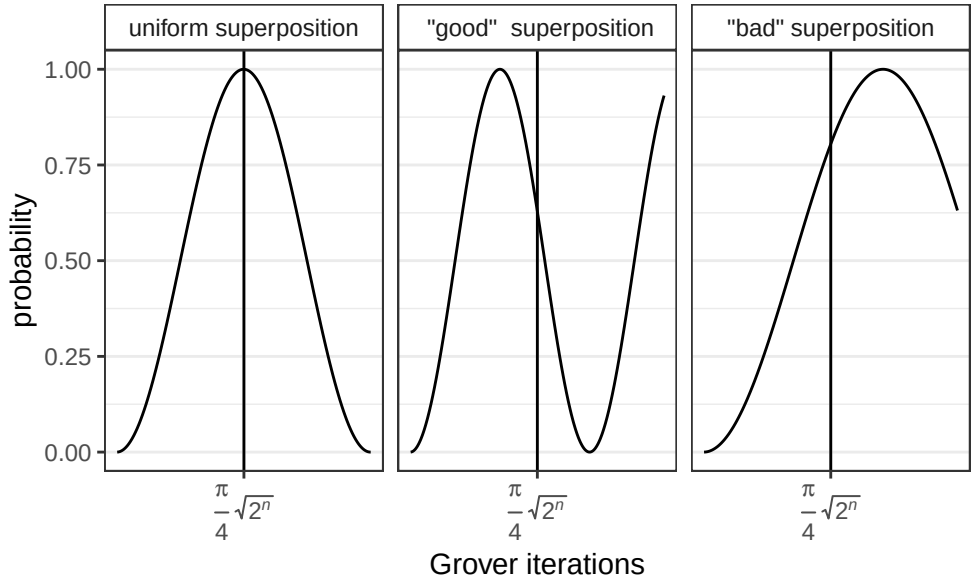


Table 1: Data for example knapsack problem that has $n = 3$ items.

i	w_i	v_i
1	2	3
2	3	1
3	2	2
$W = 4$		

is the maximum weight capacity of the knapsack. The optimal solution of the example knapsack problem is $\mathbf{x}^* = \{1, 0, 1\}$, and has value $V^* = 5$.

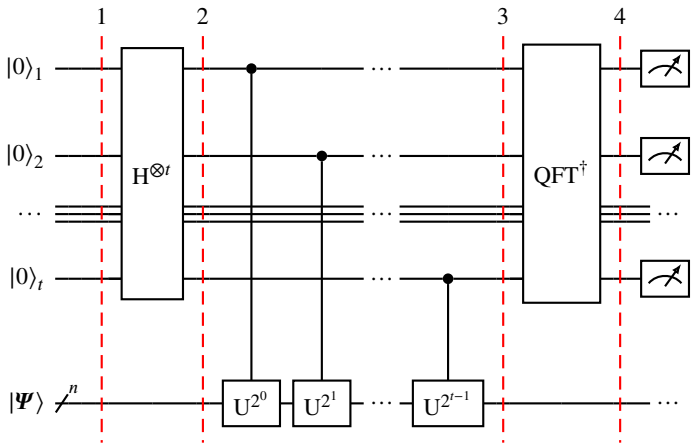
2. Quantum counting algorithms

In 1998, Brassard et al. published a quantum counting algorithm (QCB) to approximate the number of valid entries in a database that has 2^n entries. QCB can be used to approximate the number of valid solutions (and hence the required number of Grover iterations) before we run Grover's algorithm. To illustrate the limitations of QCB, we first need to introduce the quantum phase estimation algorithm (hereafter referred to as QPE).

QPE is used as a subroutine in many quantum algorithms and is considered to be one of the most important quantum algorithms (Kitaev, 1995). The circuit of QPE is presented in Figure 2. In this circuit, $|\Psi\rangle$ represents a quantum state, U is a unitary operator that operates on $|\Psi\rangle$, and $\text{QFT}^\dagger$ is an operator that performs the inverse quantum Fourier transform on a set of t counting qubits that are used to store the output of QPE. The goal of QPE is to estimate the eigenvalues of $|\Psi\rangle$. The resulting estimate is stored in a set of t counting qubits, and the precision of the estimate is determined by the size of t.

QPE can be broken down in four steps (indicated by the red dashed lines in the circuit). In what follows, we focus on the third step of QPE (this allows us to expose the main bottleneck of QCB; refer to Appendix (b) for a complete description of QPE as well as a detailed account of how QCB can be used to solve the example

Figure 2: Circuit of QPE



binary knapsack problem). In the third step, for each counting qubit $1 \leq i \leq t$, we perform a controlled $U^{2^{i-1}}$ operation on the n qubits of $|\Psi\rangle$ (i.e., operation $U^{2^{i-1}}$ is performed if counting qubit i is in basis state $|i\rangle = |1\rangle$). In order to obtain $U^{2^{i-1}}$ efficiently, we can use a method that is known as "exponentiation by squaring". This method allows to obtain operator U^{2^i} efficiently using the following recursive relationship:

$$U^{2^i} = U^{2^{(i-1)}} U^{2^{(i-1)}}, \quad \forall\, 1 \leq i < t. \tag{2.1}$$

In case of Shor's algorithm, for instance, QPE is used as a subroutine to determine the period of a function $f(r) = a^r \pmod N$. For given values of a and N, we call QPE, and perform operator U^{2^i} for each $0 \leq i < t$, where operator U^{2^i} performs operation $a^{2^i} \pmod N$. Note that, for all $i > 0$, exponentiation by squaring can be used to obtain U^{2^i} efficiently. For instance, given $a = 7$, $N = 15$, and $t = 4$, QPE performs the following operations: $7^1 \pmod{15}$ (for the first counting qubit), $7^2 \pmod{15}$ (for the second counting qubit), $7^4 \pmod{15}$ (for the third counting qubit), and $7^8 \pmod{15}$ (for the last counting qubit). Exponentiation by squaring allows to obtain $7^2 \pmod{15}$ using $7^1 \pmod{15}$, $7^4 \pmod{15}$ using $7^2 \pmod{15}$, and $7^8 \pmod{15}$ using $7^4 \pmod{15}$.

In QCB, on the other hand, operator U corresponds to a single iteration of Grover's algorithm. As a result, U^{2^i} corresponds to 2^i Grover iterations. Unfortunately, however, U^{2^i} cannot be obtained using exponentiation by squaring as this would require full knowledge of U; this would require us to know all valid solutions up front (see also Appendix (c) for a discussion and a proof). Because exponentiation by squaring cannot be used, operator U^{2^i} can only be obtained by effectively performing 2^i Grover iterations. As a result, QCB requires $\sum_{i=0}^{t-1} 2^i = (2^t - 1)$ Grover iterations if t counting qubits are used to approximate the number of valid solutions.

To illustrate how many counting qubits may be needed, we use QCB to approximate the number of valid solutions in a system that has $n = 3$ qubits and $m = 1$ valid solution. The results are presented in Table 2. From Table 2, we can conclude three things. First, the precision of the measurement depends on the number of t qubits. For instance, if $t = 1$, there are only two possible measurements: $m = 0$ and $m = 8$. For $t = 3$, we have $m = 0$, $m = 1.17$, $m = 4$, $m = 6.83$, or $m = 8$. Second, the possible measurements follow a sine pattern; it is easier to approximate a low/high number of valid solutions. As such, depending on the true value of m, more counting qubits may be needed. In this example, to be certain whether $m = 3$, we need at least 5 counting qubits (with 4 counting qubits we can only measure $m = 2.47 \approx 2$ or $m = 4.00$; it is not possible to decide that 3 is the correct measurement with only 4 counting qubits). If, for simplicity, we assume that $t = n$ qubits are used, QCB needs $(2^n - 1)$ Grover iterations that each require 2 calls to function $f_{\mathbf{x}}$ (refer to Nielsen and Chuang (2010) for a discussion on the trade-off between between the number of counting qubits and the accuracy of the approximation of m.

Table 2: Probability to measure m valid solutions using QCB for various values of t, given a system that has $n = 3$ qubits and $m = 1$ valid solution. The bottom row reports the probability to conclude that $m = 1$.

m	$P(m)$					
	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
0.00	0.8750	0.4922	0.0077	0.0072	0.0056	0.0016
0.02						0.0034
0.08					0.0138	0.0040
0.17						0.0053
0.30				0.0367	0.0282	0.0082
0.47						0.0156
0.67					0.1574	0.0456
0.91						0.6378
1.17			0.9816	0.9212	0.7085	0.2051
1.46						0.0315
1.78					0.0428	0.0124
2.11						0.0067
2.47				0.0191	0.0147	0.0042
2.84						0.0030
3.22					0.0077	0.0022
3.61						0.0018
4.00		0.4375	0.0068	0.0064	0.0049	0.0014
4.39						0.0012
4.78					0.0036	0.0010
5.16						0.0009
5.53				0.0036	0.0028	0.0008
5.89						0.0007
6.22					0.0023	0.0007
6.54						0.0006
6.83			0.0028	0.0026	0.0020	0.0006
7.09						0.0005
7.33					0.0018	0.0005
7.53						0.0005
7.70				0.0022	0.0017	0.0005
7.83						0.0005
7.92					0.0016	0.0005
7.98						0.0005
8.00	0.1250	0.0703	0.0011	0.0010	0.0008	0.0002
$P(m = 1)$	0.0000	0.0000	0.9816	0.9212	0.8658	0.9200

If only $t = n/2$ counting qubits are used, QCB needs $(\sqrt{2^n} - 1)$ Grover iterations to approximate m. When solving a discrete optimization problem, this approximation of m can be used to determine the number of Grover iterations that are required to measure a valid solution. Such an approach, however, requires $(\sqrt{2^n} - 1)$ Grover iterations (to approximate m) and $\pi/4\sqrt{2^n m^{-1}}$ Grover iterations (to measure a solution). As a result, the expected performance of an approach that uses QCB is close to its worst-case performance (i.e., we always need to perform the $(2^t - 1)$ Grover iterations required to approximate m; only if $m = 0$ we no longer need to perform the $\pi/4\sqrt{2^n m^{-1}}$ Grover iterations to measure a valid solution). Whereas GUM has also been shown to require up to $O(\sqrt{2^n})$ Grover iterations (see Appendix D of Creemers and Pérez Armas (2025)), it expects to perform far less Grover iterations (as a valid solution may be found in an early iteration of GUM). In addition, QCB only provides a single approximation of m. Given a single approximation of m, we can perform a single run of Grover's algorithm. With a single run of Grover's algorithm, however, the possibility exists that a valid solution is not found (even if our approximation of m is accurate). In contrast, GUM uses up to $(n + 1)$ approximations of m, resulting in $(n + 1)$ trials to find a valid solution. As also illustrated numerically in Appendix (a), GUM seems to be almost certain to find a valid solution (if it exists).

Table 3: Probability to measure whether a solution exists using QCB for various values of t and n, given a system that has $m = 1$ valid solution (values for which $t = \lceil n/2 \rceil$ are indicated in blue).

t	$P(m>0)$										
	$n=3$	$n=4$	…	$n=7$	$n=8$	…	$n=15$	$n=16$	…	$n=31$	$n=32$
1	0.1250	0.0625	…	0.0078	0.0039	…	0	0	…	0	0
2	0.5078	0.2822	…	0.0386	0.0194	…	0.0002	0.0001	…	0	0
3	0.9923	0.7974	…	0.1541	0.0795	…	0.0006	0.0003	…	0	0
4	0.9928	0.9616	…	0.5119	0.2913	…	0.0026	0.0013	…	0	0
5	0.9944	0.9852	…	0.9884	0.7935	…	0.0104	0.0052	…	0	0
6	0.9984	0.9992	…	0.9895	0.9640	…	0.0410	0.0207	…	0	0
7	0.9997	0.9994	…	0.9930	0.9847	…	0.1559	0.0806	…	0	0
8	0.9999	0.9998	…	0.9993	0.9997	…	0.5122	0.2919	…	0	0
9	1	1	…	0.9995	0.9997	…	0.9881	0.7933	…	0	0
10	1	1	…	1	0.9998	…	0.9893	0.9642	…	0.0002	0.0001
11	1	1	…	1	1	…	0.9930	0.9847	…	0.0007	0.0003
12	1	1	…	1	1	…	0.9993	0.9997	…	0.0026	0.0013
13	1	1	…	1	1	…	0.9996	0.9997	…	0.0104	0.0052
14	1	1	…	1	1	…	1	0.9998	…	0.0410	0.0207
15	1	1	…	1	1	…	1	1	…	0.1559	0.0806
16	1	1	…	1	1	…	1	1	…	0.5122	0.2919
17	1	1	…	1	1	…	1	1	…	0.9881	0.7933
18	1	1	…	1	1	…	1	1	…	0.9893	0.9642

To conclude, when solving discrete optimization problems, rather than using QCB to accurately approximate the number of valid solutions, we may just as well use a simple Grover-based procedure such as GUM as it achieves a quadratic speedup asymptotically and is relatively fast in practice (for a Grover-based algorithm). Often, however, we don't need a precise estimate of the number of valid solutions; it suffices to verify whether a valid solution exists. This is illustrated in Table 3 that, for various values of t, presents the probability to measure a valid solution in a system that has n qubits and only 1 valid solution (i.e., the worst-case scenario; if there are more valid solutions, less counting qubits are required to verify whether $m > 0$). Table 3 shows that we may need far fewer than $t = n$ counting qubits to get a good estimate of $P(m > 0)$. In fact, it seems that $t = \lceil n/2 \rceil$ already provides a probability of 0.29 or 0.51 to find out whether a valid solution exists (for even and uneven n, respectively). This may be of particular interest in our setting, where, for a given value of V, we don't necessarily need to know the solution itself; it suffices to know whether a solution exists. This idea is used in Algorithm 1 (hereafter referred to as CNT) that uses a binary search procedure to solve a discrete optimization problem. Procedure CNT uses QCB (or any other counting algorithm) to verify whether a solution exists. CNT performs up to $L \approx \lceil \log_2(V_{\max} - V_{\min}) \rceil$ iterations. In each iteration, we update the current value V, and use a counting algorithm to evaluate whether a solution exists that has a value of at least V. If a solution exists, we update $V_{\min}$. If no solution exists, we update $V_{\max}$. This procedure continues until we have determined that the optimal solution value is either $V_{\min}$ or $V_{\max}$. At this stage, we use GUM to obtain a solution that has value $V_{\max}$ (or $V_{\min}$ if no solution can be found that has solution value $V_{\max}$). In what follows, we use CNT to solve the example binary knapsack problem.

In the case of our example problem, the dynamics of CNT are summarized in Figure 3 for $t = 3$. To solve the example problem using CNT, we first initialize the best-found solution, the lower bound $V_{\min} = 0$, and the upper bound $V_{\max} = \sum_{i=1}^{n} v_i = 6$. Next, in a first iteration, we let $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 3$ and use QCB to verify whether a solution exists that has a value of at least 3. We have a 95.31 percent probability to conclude that a valid solution exists (i.e., $0.9531 = 0.0000 + 0.0073 + 0.9457 + 0.0000$; the sum of all probabilities to measure a value $V \geq 3$ if $t = 3$ in Table 4), after which we update the lower bound $V_{\min} = V = 3$ and use the quantum counting algorithm in a second iteration to verify if a solution exists that has value $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 5$. If we conclude that a valid solution can be found (with 99.23 percent probability), we use GUM to try and find a solution that has value $V = V_{\max} = 6$. No solution can be found that has $V = 6$, and therefore, we have to perform a second run of GUM in which we try to find a solution that

Algorithm 1: Procedure used to solve a discrete optimization problem using a quantum counting algorithm

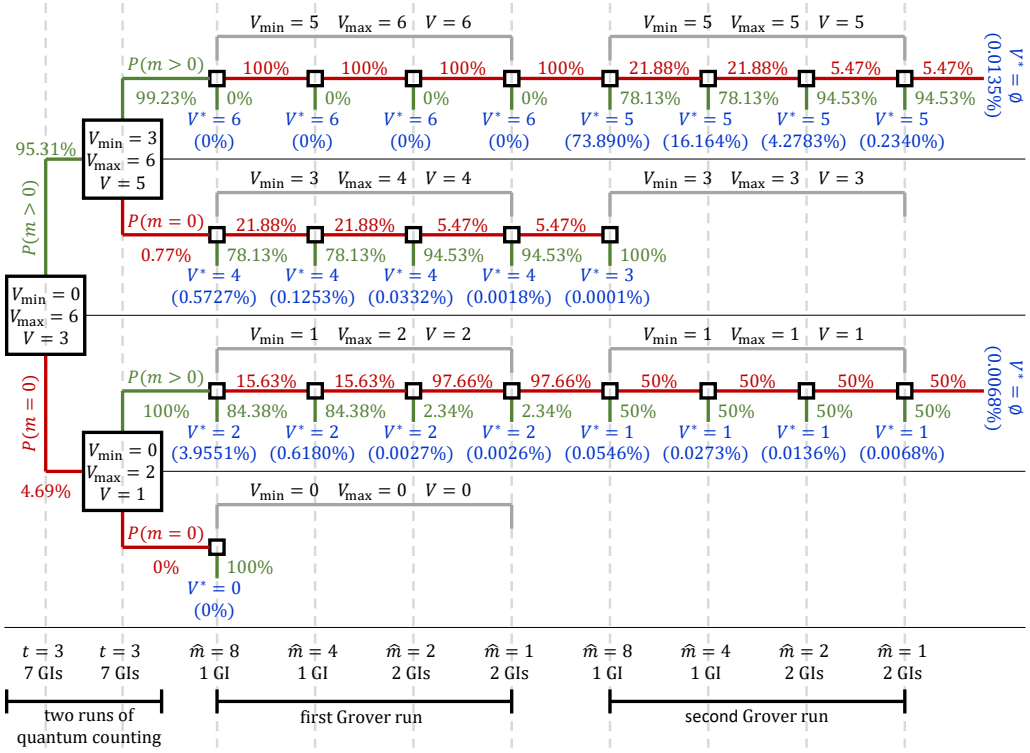
initialize $\mathbf{x}^* = \emptyset$, $V^* = 0$, $V_{\min}$, $V_{\max}$, and start procedure CNT $(n, V_{\min}, V_{\max})$;
procedure CNT $(n, V_{\min}, V_{\max})$
  do
    $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil$;
    Use quantum counting algorithm to determine $P(m > 0)$ for value V;
    if $P(m > 0) > 0$ **then**
      $V_{\min} = V$;
    else
      $V_{\max} = V - 1$;
  while $(V_{\max} - V_{\min}) > 1$;
  Use GUM to find a solution $\mathbf{x}^*$ that has value $V^* = V_{\max}$. If no solution can be found, use GUM again to find solution $\mathbf{x}^*$ that has value $V^* = V_{\min}$;

Table 4: Probability to measure a solution with value V (or to not measure a valid solution at all; row "NA"), when using CNT to solve the example knapsack problem using various number of counting qubits (t). The optimal solution and the probability to measure it are printed in blue. The worst-case and expected number of calls to function $f_{\mathbf{x}}$ are also given for each value of t.

V	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
0	0.3749	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0082	0.0041	0.0010	0.0003	0.0001	0.0000
2	0.3663	0.1831	0.0458	0.0114	0.0029	0.0007
3	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
4	0.2187	0.3998	0.0073	0.0071	0.0055	0.0016
5	0.0312	0.4125	0.9457	0.9810	0.9914	0.9975
6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
NA	0.0007	0.0003	0.0002	0.0002	0.0001	0.0001
WC[calls]	36	44	60	92	156	284
E[calls]	7.58	20.80	46.76	79.51	143.73	271.86

has value $V = V_{\max} = 5$. In this second run, we are almost certain to measure the optimal solution. If, in the second iteration, we (wrongfully) conclude that no valid solution exists that has $V = 5$ (with 0.77 percent probability), we update the upper bound $V_{\max} = V - 1 = 4$ and perform a first run of GUM in which we try to find a solution that has value $V = V_{\max} = 4$. If no solution can be found (with very low probability), we try to find a solution that has value $V = V_{\min} = 3$. In this case, we are 100 percent certain that a valid solution is measured. Returning to the first iteration, if we (wrongfully) conclude that no valid solution exists that has value $V = 3$ (with 4.69 percent probability; see also Figure 3), we update the upper bound $V_{\max} = V - 1 = 2$, and use the quantum counting algorithm in a second iteration to determine whether a solution exists that has a value of at least a value $V = \lceil 0.5(V_{\min} + V_{\max}) \rceil = 1$. In this case, we are 100 percent certain to conclude that a valid solution exists, and we perform a first run of GUM in which we try to find a solution that has value $V = V_{\max} = 2$. If no solution can be found (with very low probability), we try to find a solution that has value $V = V_{\min} = 1$. If again no solution exists, we conclude that no solution can be found (with 0.0068 percent probability). In the best case, CNT requires 15 Grover iterations (and 31 calls to function $f_{\mathbf{x}}$). In the worst case, 26 Grover iterations are required (and 60 calls). Note that a brute-force classical approach would need to evaluate at most $2^3 = 8$ solutions (i.e., would require at most 8 calls to function $f_{\mathbf{x}}$). For different values of t, Table 4 summarizes the probability to find the optimal solution, as well as the worst-case and expected number of required function calls.

Figure 3: Illustration of the dynamics of CNT for the example binary knapsack problem. We first perform two runs of QCB with 3 counting qubits (i.e., $t = 3$). Afterwards, depending on the outcome, we perform up to 2 runs of GUM and measure the solution. The solution values (and their probabilities) are given in blue. Successful outcomes of the counting algorithm and Grover's algorithm are indicated in green (vice versa, unsuccessful outcomes are indicated in red).



Each run of QCB requires $(2^t - 1) \approx 2^t$ Grover iterations, and each Grover iteration requires 2 calls to function $f_{\mathbf{x}}$. In total, CNT requires $L \approx \lceil \log_2((V_{max} - V_{min})) \rceil$ runs of the quantum counting algorithm, after which at most two runs of GUM are required. As a result, CNT requires at most $2L2^t + 2\sum_{i=0}^{n} \left(1 + 2I(n, 2^{n-i})\right)$ calls to function $f_{\mathbf{x}}$ (two calls for each Grover iteration used by the quantum counting algorithm, two calls for each Grover iteration used to measure a valid solution, and one call to validate each of the measured solutions). If we assume $t = n/2$, CNT requires $O(L\sqrt{2^n})$ calls to function $f_{\mathbf{x}}$. If $t > n/2$, however, the performance of CNT degrades. Note that, to ensure a reasonable probability to find an optimal solution, we require t to be larger than $n/2$. In addition, the expected performance of CNT is close to its worst-case performance as CNT always requires at least $2L2^t$ calls (this is also illustrated in Table 4, where the expected number of calls approaches the worst-case number of calls as t increases; the $2L2^t$ mandatory calls take the upper hand if n and/or t increase).

Note that, recently, several new quantum counting algorithms have been published (see e.g., Suzuki et al. (2020), Aaronson and Rall (2020), and Grinko et al. (2021)). Unfortunately, none of these algorithms allow to accurately approximate $P(m > 0)$ using less than $O(\sqrt{2^n})$ function calls; they are unable to significantly improve upon QCB that requires $O(\sqrt{2^n})$ calls to approximate $P(m > 0)$ (assuming $t = n/2$). In fact, in Appendix (d), we show that it is impossible to accurately approximate $P(m > 0)$ on a quantum computer using less than $O(\sqrt{2^n})$ function calls. We can, however, also use GUM to approximate $P(m > 0)$. As shown numerically in Appendix (a), GUM is almost certain to find a solution (if it exists) using at most $O(\sqrt{2^n})$ function calls. In addition, next to verifying whether $m > 0$, GUM also returns a valid solution. Therefore, also

to verify whether $m > 0$, a simple Grover-based procedure such as GUM can be used instead of a quantum counting algorithm.

3. Nested quantum search

In this section, we discuss the nested quantum search algorithm (NQC) that was introduced by Cerf et al. (2000) in order to improve the performance of Grover's algorithm when solving NP-complete problems. First, however, let's return to the initial purpose of Grover's algorithm: to find a target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries using a minimum of calls to a function $f_{\mathbf{x}}$ that returns 1 if $\mathbf{x} = \mathbf{x}^*$ (and 0 otherwise). Whereas a classical algorithm requires at most 2^n calls to function $f_{\mathbf{x}}$, Grover's algorithm only needs $\sqrt{2^n}$ calls, resulting in a quadratic speedup. However, if we consider a structured database, Grover's algorithm still needs $\sqrt{2^n}$ calls. A classical algorithm may need far less calls. For instance, if the database is sorted, binary search may be used, and $(n+1)$ calls suffice to find $\mathbf{x}^*$. In other words, if a classical algorithm can exploit the structure of a problem, it can easily outperform a straightforward implementation of Grover's algorithm.

To address this problem, Cerf et al. (2000) have suggested to use a nested quantum search. Rather than using Grover's algorithm to search the entire solution space to find an optimal solution, a nested quantum search "nests" one quantum search within another. This way, partial solutions (that are obtained efficiently using Grover's algorithm) are used to build other partial solutions that, in turn, can be used to build an optimal solution. The nested quantum search algorithm is the quantum counterpart of a classical nested search algorithm, and often allows to exploit the structure of a problem. A classical nested search algorithm performs $2^{\gamma n}$ operations, where n is the number of binary decision and γ is some number less-or-equal-than 1 that depends on the problem instance. Cerf et al. (2000) have shown that the nested quantum search only needs $\sqrt{2^{\gamma n}}$ operations, once more resulting in a quadratic speedup.

To illustrate the nested quantum search algorithm, we use it to find a single target entry in a database that has 2^n entries. To keep things simple, we assume a single level of nesting. As a result, the problem of identifying the target entry is divided in two partial problems. In a first partial problem, we try to find the basis state created by the first i qubits that correspond to the first i bits of the entry. For this purpose, let $\mathbf{x}_{1i}$ represent the first i qubits, and let $f_{\mathbf{x}1i}$ denote the function that returns 1 if $\mathbf{x}_{1i} = \mathbf{x}^*_{1i}$, where $\mathbf{x}^*_{1i}$ are the first i bits of the target entry. Conversely, let $\mathbf{x}_{(i+1)n}$ represent the last $(n-i)$ qubits, and let $f_{\mathbf{x}(i+1)n}$ denote the function that returns 1 if $\mathbf{x}_{(i+1)n} = \mathbf{x}^*_{(i+1)n}$. After solving the first partial problem, we only solve the second partial problem for those basis states where $\mathbf{x}_{1i} = \mathbf{x}^*_{1i}$. The circuit of the nested quantum search algorithm is given in Figure 4 and can be broken down in 5 steps (indicated by the red dashed lines in the circuit). To illustrate the different steps of the algorithm, we use an example that has $n = 3$ qubits (resulting in a database that has 8 entries) and one target basis state (that corresponds to the target entry). The target entry of the database is $\mathbf{x}^* = \{1, 0, 1\}$ (represented by basis state $|101\rangle$). Here, we let $i = 1$ (i.e., in a first subproblem we consider the first qubit, and after having determined the correct basis state for this first qubit, we try to find the correct basis state for the last two qubits). In a first step of the algorithm, we initialize $n+2$ qubits: a z qubit that ensures the nested search on the last two qubits is only performed for basis states where $\mathbf{x}_{11} = \mathbf{x}^*_{11} = \{1\}$, the set containing the first x qubit, the set containing the last two x qubits, and a y qubit that is used to induce phase kickback. After step 1, we obtain $|\Psi_1\rangle = |00001\rangle$. Next, in step 2, all qubits except the z qubit are passed through a Hadamard gate. We obtain $|\Psi_2\rangle = |0+++-\rangle$. In step 3, oracle $O_{f\mathbf{x}11}$ maps $|z\rangle$ to $|z \veebar f_{\mathbf{x}11}\rangle$:

$$|\Psi_3\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

In step 4, we apply $I(1,1) = 1$ Grover iteration in which: (a) oracle $O_{f\mathbf{x}11}$ flips the phase of basis state $|z1\mathbf{x}_{23}y\rangle$, and (b) the diffusion operator reflects the probability amplitudes about the average probability amplitude (note that gate D represents the diffusion operator). We have:

$$|\Psi_{4.a}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\Psi_{4.b}\rangle = \frac{1}{\sqrt{2}}\left(-|00\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

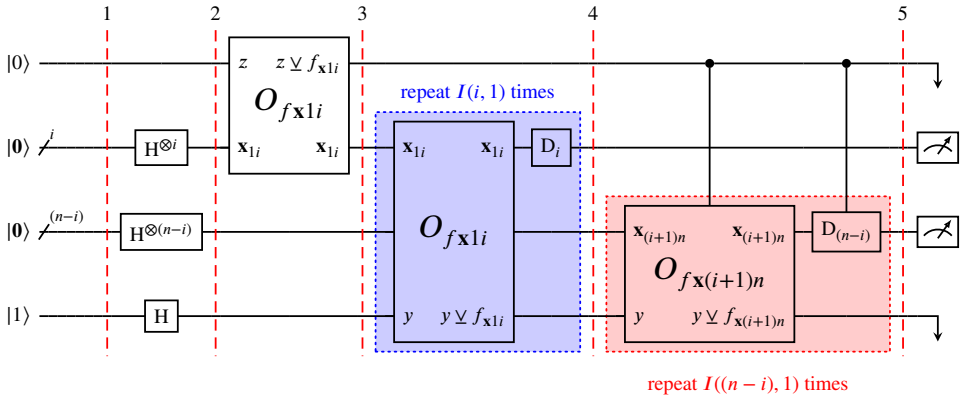
Note that, if we apply Grover's algorithm to a system that only has one x qubit, we always have a 50 percent probability to measure the target basis state, regardless of the number of Grover iterations. In step 5, we perform a controlled run of Grover's algorithm if the z qubit is in state $|1\rangle$. In other words, we only perform Grover's algorithm for basis state $|1\mathbf{x}_{11}\mathbf{x}_{23}y\rangle$ (whereas basis state $|0\mathbf{x}_{11}\mathbf{x}_{23}y\rangle$ remains unchanged). We apply $I(2,1) = 1$ Grover iteration, and obtain (unchanged basis states are given in gray):

$$|\Psi_{5.a}\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|11\rangle \otimes \frac{1}{2}\left(|00\rangle + |10\rangle - |01\rangle + |11\rangle\right) \otimes |-\rangle\,,$$

$$|\Psi_{5.b}\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|11\rangle \otimes \left(\frac{2}{2}|01\rangle + \frac{0}{2}\left(|00\rangle + |10\rangle + |11\rangle\right)\right) \otimes |-\rangle = -\frac{1}{\sqrt{2}}|00{+}{+}{-}\rangle + \frac{1}{\sqrt{2}}|1101{-}\rangle.$$

Note that, if we apply Grover's algorithm to a system that has two x qubits, we are 100 percent certain to measure the only target basis state after one Grover iteration. After step 5, we measure the qubits, and are 50 percent certain to measure correct target basis state. If we would have used Grover's algorithm on the entire search space, we need $I(3,1) = 2$ Grover iterations that perform operations on 3 qubits. Using a nested quantum search algorithm, we also require 2 Grover iterations, however, they only perform operations on 1 and 2 qubits, respectively. In addition, note that, for larger n, the nested quantum search algorithm requires far less Grover iterations than a straightforward implementation of Grover's algorithm.

Figure 4: Circuit of the nested quantum search algorithm with one level of nesting.



The question remains, however, how we can use nested quantum search to solve a discrete optimization problem. To answer this question, we first use the nested quantum search algorithm presented in Figure 4 to find a solution to the example binary knapsack problem that has a value of at least $V = 5$ (i.e., $\mathbf{x}^* = \{1, 0, 1\}$ is the only valid solution). In addition, we assume $i = 1$. Therefore, in a first partial problem, we have to decide whether or not to add item 1. We can, however, not use function $f_{\mathbf{x}}$ to evaluate whether or not to add item 1 as $f_{\mathbf{x}}$ will check if the solution where we add item 1: (1) exceeds the weight capacity, and (2) has a value of at least $V = 5$. Because item 1 only has a value of 3, $f_{\mathbf{x}}$ would incorrectly conclude that item 1 is not part of the optimal knapsack solution. As a result, we need to define a "relaxed" version of $f_{\mathbf{x}}$ that does not reject partial solutions that can still be used to build optimal solutions. In this example, other items can still be added in future partial searches such that a value of at least $V = 5$ is obtained; other items, however, cannot reduce the weight of the current knapsack solution (i.e., if the weight of a partial solution exceeds the maximum weight constraint, that partial solution may be rejected). As a result, in a first partial problem we use function $g_{\mathbf{x}}$ that returns 1 if knapsack $\mathbf{x}$ respects the maximum weight constraint. It should be clear that $g_{\mathbf{x}}$ is less effective than $f_{\mathbf{x}}$ in reducing the size of the search space. This is also illustrated in our example,

where $g_{\mathbf{x}}$ is unable to decide whether item 1 has to be added (a knapsack with or without item 1 both respect the maximum weight constraint). We have:

$$|\Psi_1\rangle = |00001\rangle\,,$$

$$|\Psi_2\rangle = |0+++-\rangle\,,$$

$$|\Psi_3\rangle = \frac{1}{\sqrt{2}}\left(|10\rangle + |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\Psi_{4.a}\rangle = \frac{1}{\sqrt{2}}\left(-|10\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,,$$

$$|\Psi_{4.b}\rangle = \frac{1}{\sqrt{2}}\left(-|10\rangle - |11\rangle\right) \otimes |{+}{+}\rangle \otimes |-\rangle\,.$$

In a second (and last) partial search, again, we have to use $g_{\mathbf{x}}$ to decide whether or not to add items 2 and 3. We identify three valid partial solutions: not adding any item, adding only item 2, and adding only item 3. After two Grover iterations, we obtain (note that after $I(2,3) = 1$ Grover iteration we are 0 percent certain to measure a target basis state):

$$|\Psi_{5.1a}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \frac{1}{2}(-|00\rangle - |10\rangle - |01\rangle + |11\rangle) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \frac{1}{2}(-|00\rangle - |10\rangle - |01\rangle + |11\rangle) \otimes |-\rangle,$$

$$|\Psi_{5.1b}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(-\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(-\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle,$$

$$|\Psi_{5.2a}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(\frac{0}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{2}{2}|11\rangle\right) \otimes |-\rangle,$$

$$|\Psi_{5.2b}\rangle = -\frac{1}{\sqrt{2}}|10\rangle \otimes \left(-\frac{1}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{1}{2}|11\rangle\right) \otimes |-\rangle - \frac{1}{\sqrt{2}}|11\rangle \otimes \left(-\frac{1}{2}(|00\rangle + |10\rangle + |01\rangle) + \frac{1}{2}|11\rangle\right) \otimes |-\rangle.$$

After step 5, we have an equal probability to measure any of the solutions. Note that: (1) several solutions exceed the maximum weight capacity, (2) there is only a 1/8 probability to identify the optimal solution, and (3) we actually never impose the constraint that a solution needs to have a value of at least $V = 5$. In other words, this example shows that it may not always be possible to use a nested (quantum) approach to solve discrete optimization problems.

4. Amplitude amplification

As Grover's algorithm requires $\pi 4^{-1}\sqrt{2^n}$ iterations to measure a (single) valid solution, it may be difficult to solve discrete problems using less than $O(\sqrt{2^n})$ operations when using a Grover-based algorithm. To resolve this problem, amplitude amplification tries to reduce the number of iterations (and hence operations). It does so by generalizing Grover's algorithm (see e.g., Brassard et al., (2002)). Whereas Grover's algorithm starts with a "uniform superposition" of qubits (i.e., a superposition where each qubit has an equal probability to collapse into either $|0\rangle$ or $|1\rangle$; a superposition that can be achieved by passing n qubits initialized as $|\mathbf{0}\rangle$ through a Hadamard gate), amplitude amplification uses a superposition that allows to measure the target state using (far) less iterations. This is illustrated in Figure 1 that illustrates the uniform superposition (used by Grover's algorithm), a "good" superposition, and a "bad" superposition. A good superposition uses information of the optimization problem to reduce the number of iterations required to measure a valid solution. For instance, consider the example problem that has $n = 3$ qubits and target basis state $|101\rangle$. If we use Grover's algorithm, we have initial state $|000\rangle$, superposition state $\mathrm{H}^{\otimes 3}\,|000\rangle = |{+}{+}{+}\rangle$, and a 78.125

percent probability to measure target basis state $|101\rangle$ after flipping the phase of $|101\rangle$ and applying diffusion operator $\left(2\left|\mathrm{H}^{\otimes 3}000\right\rangle\left\langle\mathrm{H}^{\otimes 3}000\right|-I_8\right)$:

$$\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{\sqrt{8}}\left(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle\right).$$

If, on the other hand, we use superposition $\left|\mathrm{H}^{\otimes 3}0+0\right\rangle$, we have initial state $|0+0\rangle$, superposition state $\left|\mathrm{H}^{\otimes 3}0+0\right\rangle=|+0+\rangle$, and a 100 percent probability to measure target basis state $|101\rangle$ after flipping the phase of $|101\rangle$ and applying diffusion operator $\left(2\left|\mathrm{H}^{\otimes 3}0+0\right\rangle\left\langle\mathrm{H}^{\otimes 3}0+0\right|-I_8\right)$ (i.e., the second qubit is always in state $|0\rangle$):

$$\frac{2}{2}|101\rangle+\frac{0}{2}\left(|000\rangle+|100\rangle+|001\rangle\right).$$

This example shows that amplitude amplification can indeed reduce the number of iterations (and hence operations) required to measure a target basis state. It is, however, also possible to find an example where amplitude amplification increases the number of iterations. For instance, if we use superposition $\left|\mathrm{H}^{\otimes 3}0-0\right\rangle$, we have 0 percent probability to measure target basis state $|101\rangle$ (i.e., this superposition always puts the second qubit in state $|1\rangle$, making it impossible to measure target basis state $|101\rangle$). The difficulty, therefore, is to identify a "good" superposition.

Imagine, for instance, the following superposition where the first and third qubit have a higher probability to collapse into basis state $|1\rangle$, whereas the second qubit has a higher probability to collapse into basis state $|0\rangle$:

$$\left|\Psi'\right\rangle=\left(\frac{5\,|0\rangle}{13}+\frac{12\,|1\rangle}{13}\right)\otimes\left(\frac{12\,|0\rangle}{13}+\frac{5\,|1\rangle}{13}\right)\otimes\left(\frac{5\,|0\rangle}{13}+\frac{12\,|1\rangle}{13}\right).$$

Even though this superposition makes practical sense (the qubits have a higher probability to collapse into $|101\rangle$), it does not result in a decreased number of iterations as can be seen in Figure 5 that presents the number of iterations required by a uniform superposition, a "good" superposition ($\left|\Psi''\right\rangle$), and a "bad" superposition ($\left|\Psi'\right\rangle$). From Figure 5, it is clear that superposition $\left|\Psi''\right\rangle$ performs best; it requires the least iterations to measure $|101\rangle$. Superposition $\left|\Psi''\right\rangle$, however, is given by:

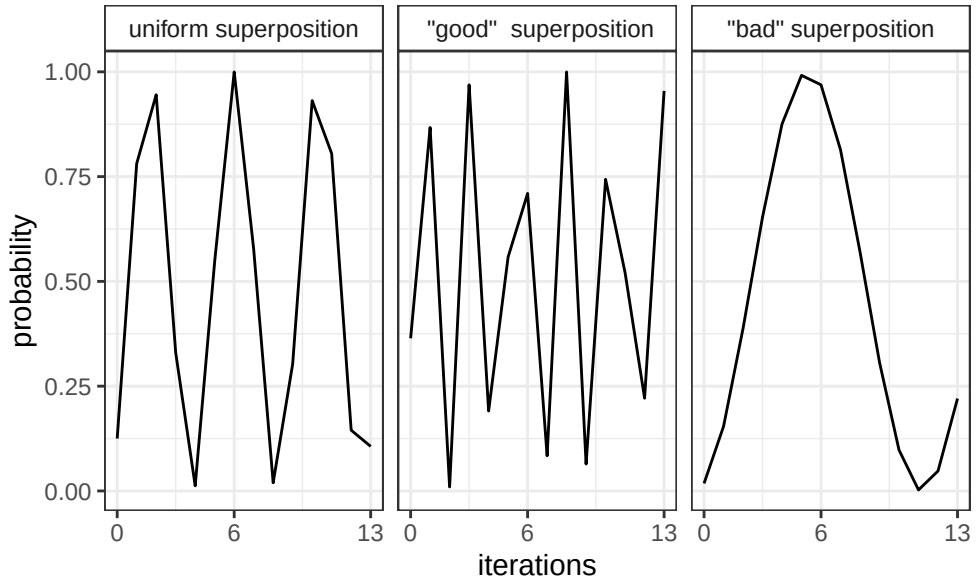
$$\left|\Psi''\right\rangle=\left(\frac{-9\,|0\rangle}{41}+\frac{40\,|1\rangle}{41}\right)\otimes\left(\frac{9\,|0\rangle}{41}+\frac{40\,|1\rangle}{41}\right)\otimes\left(\frac{-9\,|0\rangle}{41}+\frac{40\,|1\rangle}{41}\right).$$

In other words, if we measure superposition $\left|\Psi''\right\rangle$, all qubits have an equal probability (of $40^2/41^2=0.9756$) to collapse into basis state $|1\rangle$.

The success of amplitude amplification depends on our ability to find a good superposition based on problem instance characteristics. The previous example, however, illustrates that this may not be that easy. In addition, note that amplitude amplification cannot be used to perform an unstructured search using less than $O(\sqrt{2^n})$ operations (this would contradict the result of Zalka (1999); see also Boyer et al. (1996) and Bennett et al. (1997)). As such, for an unstructured search, amplitude amplification cannot outperform Grover's algorithm. This (once more) indicates that it may not be that easy for algorithms that use amplitude amplification to outperform Grover-based algorithms (i.e., algorithms that assume a uniform superposition).

To further investigate the potential of amplitude amplification, we perform a test where 1000 random superpositions are compared to the uniform superposition for various values of n and m. To generate a random superposition, we assign a random number in $[0, 1]$ to each basis state, and normalize to get the probability to measure each basis state. Next, for each basis state, we determine its amplitude (i.e., the square root of its probability) and its sign (by drawing another random number in $[0, 1]$; the amplitude is negative if the random number is smaller than 0.5, and positive otherwise). For each random superposition, we assess the number of iterations required to reach a first extremum as well as the number of iterations required to be at least 95 percent certain to measure a valid solution. The results of the test are presented in Figures 6 and 7. Whereas Figure 6 presents the probability to measure any of the m valid solutions after a given number of iterations, Figure 7 presents the cumulative probability that a random superposition reaches a first extremum (gray full line) or has a probability of at least 95 percent to measure a valid solution (black dashed line) after a given number of iterations. In both figures, the performance of Grover's algorithm (i.e.,

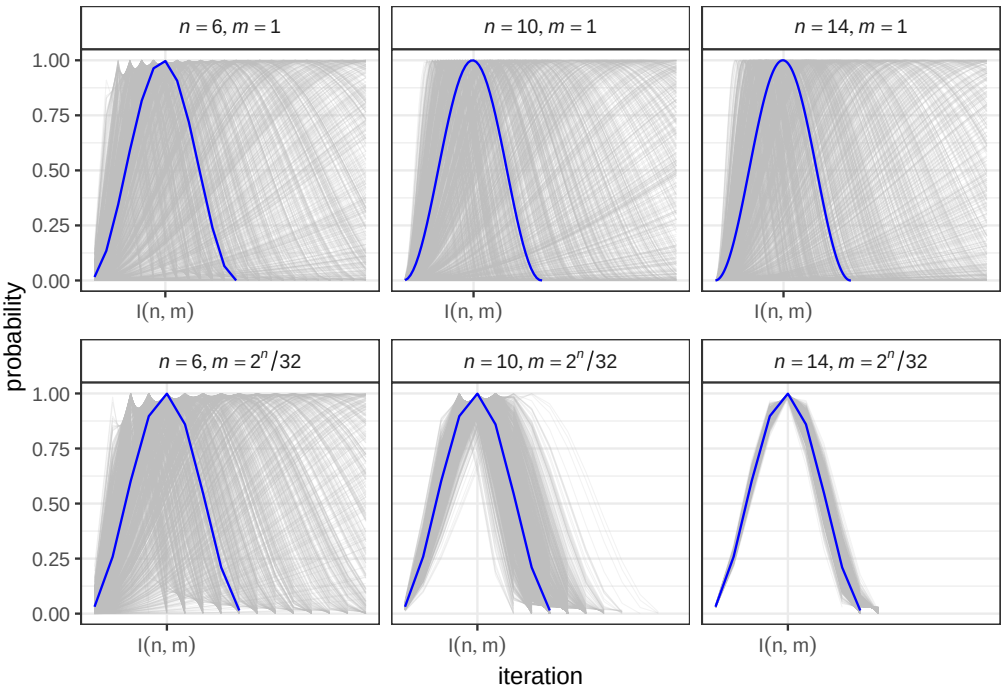
Figure 5: Comparison of a uniform superposition (used by Grover's algorithm), a "good" superposition (Ψ''), and a "bad" superposition (Ψ') when measuring a valid solution in a system that has 1 valid solution and 3 qubits.



a uniform superposition) is indicated by a blue line. If we consider the case where $m = 1$ (top rows of Figures 6 and 7), there are certainly superpositions that require less iterations than a uniform superposition. Figure 7, however, shows that a random superposition only has a 30 percent chance (roughly) to outperform the uniform superposition. In addition, the potential decrease in the number of required iterations (i.e., the upside potential) is far smaller than the potential increase in iterations (i.e., the downside risk). Note that finding $m = 1$ valid solution in a system that has $n = 6$ qubits (and 2^6 solutions) is far easier than finding $m = 1$ valid solution in a system that has $n = 14$ qubits (and 2^{14} solutions). To investigate the impact of the number of valid solutions on the performance of a random superposition, we observe the bottom rows of Figures 6 and 7. In the bottom rows, the proportion of valid solutions is kept constant (i.e., $m = 2^n/32$; one out of 32 solutions is a valid solution). It is clear that, as n increases, the performance of all superpositions converges. This leads us to conclude two things: (1) it does not make sense to use a random superposition if m is sufficiently large, and (2) if m is small, a superposition can be found that reduces the number of required iterations (when compared to a uniform superposition), however, randomly guessing this superposition may not be the best idea.

We perform an additional test to investigate the random superpositions that outperform the uniform superposition when $m = 1$ for various values of n (here, a random superposition outperforms the uniform superposition if it requires less iterations to be at least 95 percent certain to measure a valid solution). For this test, we collect 10000 random superpositions as well as 10000 random superpositions that outperform the uniform superposition. The results are summarized in Figure 8. Figure 8 shows boxplots for the average and standard deviation of the amplitudes of all random superpositions (dark gray boxplot) and of those superpositions that outperform the uniform superposition (light gray boxplot). Figure 8 also shows the boxplots for the correlation between the "optimal" superposition (where valid solutions have an amplitude of 1 and invalid solutions have an amplitude of 0) and all random superpositions (dark gray boxplot) and those superpositions that outperform the uniform superposition (light gray boxplot). If we observe the dark gray boxplots, we see (as expected) that the average amplitude as well as the average correlation equals 0. In addition, as n increases, we observe that average amplitude as well as correlation converges to 0. On the other hand, if we look at the random superpositions that outperform the uniform superposition (light gray boxplots), we observe almost identical results. In other words, superpositions that outperform the uniform

Figure 6: Comparison of the probability to measure a valid solution after a given number of iterations when using a uniform superposition (used by Grover's algorithm; indicated by the blue line) and 1000 random superpositions for various values of n and m.

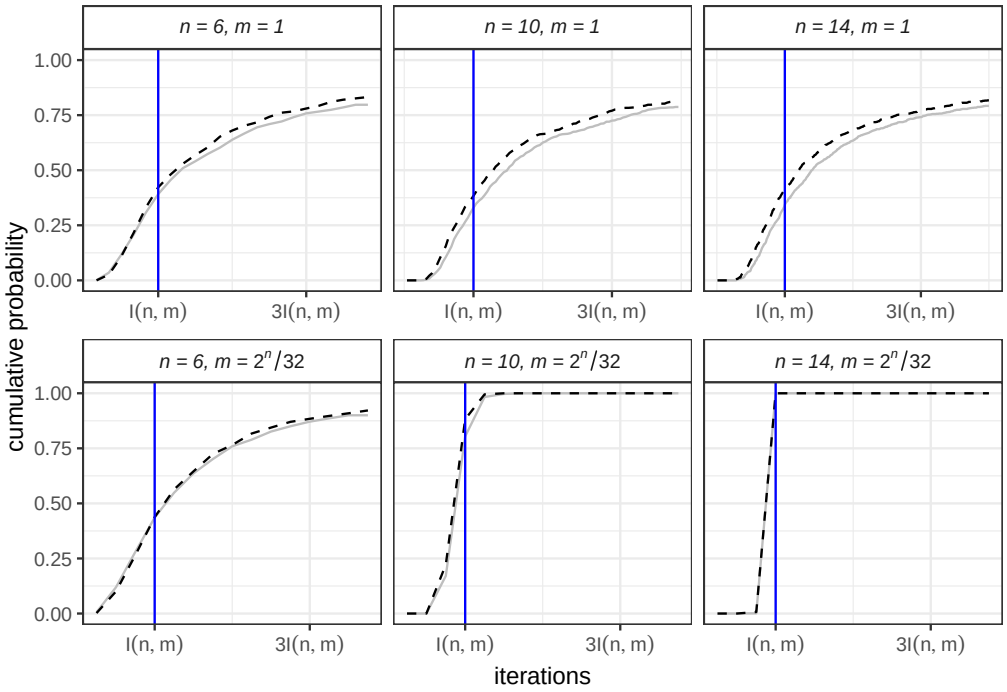


superposition do not seem to differ from superpositions that do not outperform the uniform superposition. This, once more, indicates that it may be difficult to identify a superposition that outperforms the uniform superposition.

In the previous tests, valid solutions where randomly selected. For some problems, however, valid solutions are clustered closely together. To verify the performance of random superpositions when valid solutions are clustered, we perform another test. In this test, we draw a random solution that is selected as the first of the m valid solutions that are all clustered together (e.g., if $m = 3$, and basis state i is drawn, the valid solutions are i, $i + 1$, and $i + 2$). Our results, however, show there is no difference in performance if valid solutions are clustered. Lastly, to verify the impact of randomly choosing the sign of the amplitude of a basis state, we perform yet another test in which the sign of all amplitudes is either positive or negative. The results of this test show that the performance of random superpositions deteriorates significantly if the sign of amplitudes is always positive or negative. This is somewhat surprising as a uniform superposition has amplitudes that are always positive.

To conclude, in order to outperform a uniform superposition, we can use amplitude amplification only if we are able to prepare a superposition that takes into account the structure of a discrete optimization problem. For instance, imagine a discrete optimization problem that has two sets of constraints C_1 and C_2. In addition, assume there are 2^n possible solutions and m_1 of these solutions satisfy the first set of constraints. On the one hand, if we apply Grover's algorithm to find one of the $m \leq m_1$ solutions that satisfy all constraints, we need $O(\sqrt{2^n m^{-1}})$ Grover iterations. If, on the other hand, we are able to prepare a superposition that has a non-zero probability amplitude only for those basis states that correspond to solutions that satisfy the first set of constraints, we only need to perform $O(\sqrt{2^{m_1} m^{-1}})$ Grover iterations to find one of the m valid solutions (as we are searching the set of solutions that already satisfy the first set of constraints). As such, if

Figure 7: Cumulative probability that a random superposition reaches a first extremum (gray full line) or has a probability of at least 95 percent to measure a valid solution (black dashed line) after a given number of iterations for various values of n and m. The number of iterations required by Grover's algorithm (i.e., a uniform superposition) is indicated by a blue line.



the right superposition can be prepared efficiently, amplitude amplification may allow a significant speedup when solving discrete optimization problems.

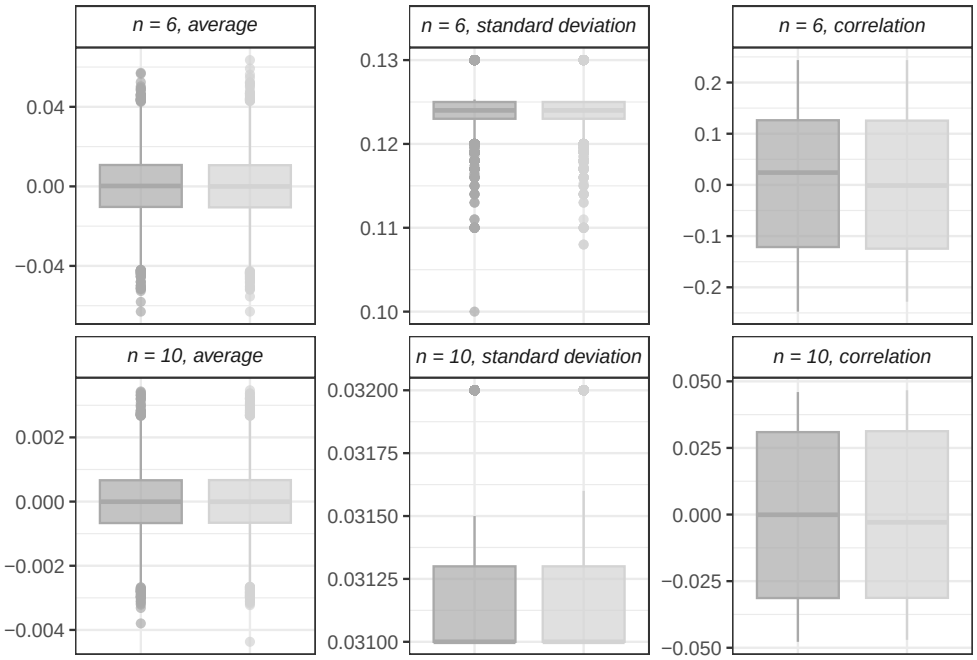
5. Conclusion

In this paper, we discuss Grover-based quantum algorithms and investigate their limitations when solving discrete optimization problems.

First, we discuss the quantum counting algorithm of Brassard et al. (1998; referred to as QCB). Similar to the prime-factorization algorithm of Shor (1994), QCB relies heavily on the quantum phase estimation algorithm (QPE). In contrast to the algorithm of Shor, however, QCB cannot exploit a technique that is known as "exponentiation by squaring". As a result, if $t = n/2$ counting qubits are used, QCB requires at least $O(\sqrt{2^n})$ operations to approximate the number of valid solutions in a set of 2^n solutions. Given the number of valid solutions (m), we can use Grover's algorithm to find one of the m valid solutions. This, however, requires another $O(\sqrt{2^n m^{-1}})$ operations. The expected performance of QCB, therefore, is close to its worst-case performance (as we need to perform at least $O(\sqrt{2^n})$ operations to approximate m). In contrast, even though GUM and QCB have the same asymptotic runtime, GUM (a simple Grover-based procedure) may have expected performance that is much better than its worst-case performance (because a valid solution may already be found in an early iteration of GUM). Therefore, a simple procedure such as GUM may be a sensible option to find a valid solution when solving discrete optimization problems. Also to verify whether a valid solution exists (i.e., to assess $P(m > 0)$), our results indicate that is may be a valid option to use a procedure such as GUM.

Next, we discuss the nested quantum search algorithm of Cerf et al. (2000; referred to as NQC). Rather than using Grover's algorithm to search the entire solution space to find an optimal solution, a nested quantum

Figure 8: Average amplitude and standard deviation of amplitudes over all superpositions (dark gray boxplots) and better superpositions (i.e., superpositions that improve the uniform superposition; light gray boxplots) as well as the mean correlation between the "optimal" superposition and all superpositions (dark gray boxplot) and better superpositions (light gray boxplot) for various values of n.



search "nests" one quantum search within another. This way, partial solutions (that are obtained efficiently using Grover's algorithm) are used to build other partial solutions that, in turn, can be used to build an optimal solution. The nested quantum search algorithm is the quantum counterpart of a classical nested search algorithm and allows to exploit the structure of a problem. Our analysis, however, seems to indicate that a nested quantum search algorithm cannot be used to solve all discrete optimization problems.

Last but not least, we investigate the potential of amplitude amplification. Amplitude amplification is a generalization of Grover's algorithm that tries to reduce the number of iterations (and hence operations) by using a "good" superposition. Finding such a "good" superposition, however, may prove to be difficult. We use amplitude amplification to solve an example problem and illustrate that the use of logical rules can result in a "bad" superposition (resulting in more iterations than those that are required by Grover's algorithm and/or in the impossibility to find an optimal solution). This finding is confirmed by an elaborate experiment that compares a large number of random superpositions. The experiment shows that "good" superpositions cannot be distinguished from "bad" superpositions. In addition, another experiment shows that only few random superpositions exist that can outperform the uniform superposition used by Grover's algorithm. To make matters worse, the upside potential of the random superpositions that outperform the uniform superposition is far smaller than the downside risk (i.e., while there are some random superpositions that require a slightly lower number of iterations than Grover's algorithm, the majority of random superpositions require far more iterations). Therefore, we conclude that a "good" superposition needs to be prepared with care and precision.

Data accessibility

The code and data are available upon request.

Authors' contributions

S.C.: conceptualization, investigation, writing—original draft, writing—review, and editing; L.F.P.A: conceptualization, writing—review, and editing.

Both authors gave their final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration

We declare that we have no competing interests.

Funding

This research benefited from the support of the FMJH Program PGMO.

A. Appendix

(a) Procedure GUM

Procedure GUM is a straightforward application of Grover's algorithm that may be used to assess whether a valid solution exists in a set of 2^n solutions if m is unknown. Because m is unknown, GUM evaluates several assumed numbers of valid solutions ($\hat{m}$). Starting with $\hat{m} = 2^n$, GUM calls Grover's algorithm, and performs $I(n, \hat{m})$ Grover iterations, where $I(n, \hat{m})$ rounds $\pi 4^{-1}\sqrt{\hat{m}^{-1}2^n}$ to the nearest integer. In each Grover iteration, a function $f_{\mathbf{x}}$ (that returns 1 if solution $\mathbf{x} = \{x_1, \ldots, x_n\}$ is valid and 0 otherwise) is called twice (once to determine whether solution $\mathbf{x}$ is valid and once to uncompute; refer to, for instance, Nielsen and Chuang (2010) for more details). If, after $I(n, \hat{m})$ Grover iterations, a valid solution $\mathbf{x}$ is found, GUM stops and returns $\mathbf{x}$. If no valid solution is found, we let $\hat{m} = \hat{m}/2$ and try again. If no solution is found after processing $\hat{m} = 1$, GUM stops, and we conclude that no valid solution exists (i.e., we conclude that $m = 0$). The number of times GUM expects to call function $f_{\mathbf{x}}$ is given by:

$$\sum_{i=0}^{n}\left((i+1)\sum_{j=0}^{i}2I(n,2^{(n-j)})\right)(1-\phi_{i-1})P(n,m,I(n,2^{(n-i)})), \tag{A 1}$$

where $P(n, m, I(n, 2^{(n-i)}))$ is the probability to measure any of the m valid solution if we use $I(n, 2^{(n-i)})$ Grover iterations to search a set of 2^n solutions, and where ϕ_i is the probability of having measured a valid solution after i GUM iterations and can be obtained through the following recursive relationship (with $\phi_{-1} = 0$):

$$\phi_i = \phi_{i-1} + (1-\phi_{i-1})P(n,m,I(n,2^{(n-i)})). \tag{A 2}$$

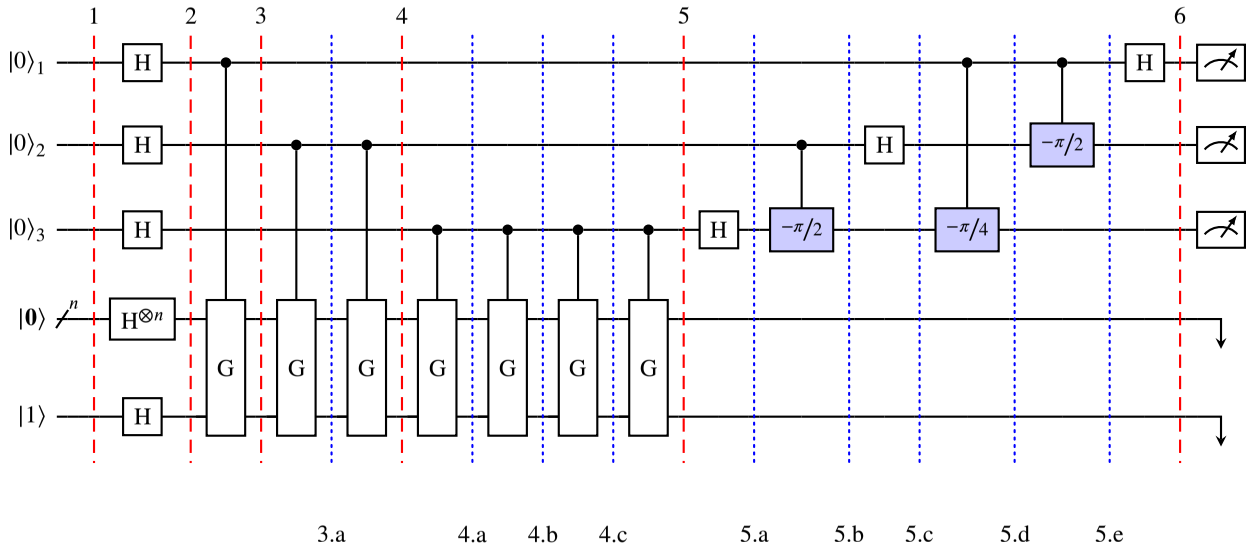
From Equation A 2, it is clear that, as $n \to \infty$, $\phi_n \to 1$. One can verify that, in the worst case, GUM requires $O(\sqrt{2^n})$ function calls. If, however, there are m valid solutions, GUM is conjectured to require $O(\sqrt{m^{-1}2^n})$ function calls (see also Creemers and Pérez Armas (2025) for a discussion). GUM can be used as a subroutine in procedures that solve (discrete) optimization problems. For instance, Creemers and Pérez Armas (2025) adopt GUM in several procedures that are used to solve the binary knapsack problem.

(b) Quantum phase estimation (QPE) and its use in the quantum counting algorithm of Brassard et al. (1998; QCB)

In this section, we illustrate how QCB can be used to approximate the number of valid binary knapsack solutions. For this purpose, we use the example binary knapsack instance that has 3 items, and assume we are counting the number of solutions that have a value V of at least 5 (i.e., only $\mathbf{x} = \{1, 0, 1\}$ is a valid solution). In addition, we use 3 counting qubits (i.e., $t = 3$).

The circuit of the quantum counting algorithm is presented in Figure 9 (whereas red dashed lines indicate the main steps of the algorithm, blue dashed lines indicate substeps). The circuit has three

Figure 9: QCB using $t = 3$ counting qubits to approximate the number of valid solutions in a quantum system that has n qubits. The blue gates represent controlled phase shift gates that shift the phase of the quantum state of a qubit by a period 2π divided by a power of 2.



Algorithm 2: Procedure that uses Grover's algorithm to find a valid solution when the number of valid solutions is unknown in a system that has n binary decision variables, and where $f_{\mathbf{x}}$ returns 1 if solution $\mathbf{x} = \{x_1, \ldots, x_n\}$ is valid (and 0 otherwise).

```
procedure GUM(n)
    m̂ = 2^n;
    do
        perform I(n, m̂) iterations of Grover's algorithm equipped with f_x;
        measure basis state |x⟩;
        if f_x = 1 then
            return x;
        else if m̂ > 1 then
            m̂ = m̂/2;
        else
            return ∅;
    while true;
```

counting qubits, three x qubits representing the binary decision variables, and one y qubit that is used to induce phase kickback. All qubits are initialized as $|0\rangle$ except for the y qubit that is initialized as $|1\rangle$. We have $|\Psi_1\rangle = |0000001\rangle$. In a first step, all qubits are passed through a Hadamard gate. We obtain $|\Psi_2\rangle = |+++++++-\rangle$. In a second step, we perform a controlled Grover iteration; we perform a Grover iteration if the first counting qubit is in state $|1\rangle$ (i.e., only if the first counting qubit is in state $|1\rangle$, we flip the phase of $|\mathbf{t}101y\rangle$ and apply the diffusion operator). As a result, we have:

$$\begin{aligned}|\Psi_3\rangle \;=\; & \frac{|0\rangle}{\sqrt{2}}\otimes|{+}{+}\rangle\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle\\ +\; & \frac{|1\rangle}{\sqrt{2}}\otimes|{+}{+}\rangle\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.\end{aligned}$$

In a third step, we perform two Grover iterations controlled by the second counting qubit. After a first Grover iteration, we obtain:

$$\begin{aligned}|\Psi_{3.a}\rangle \;=\; & \frac{|00\rangle}{2}\otimes|+\rangle\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle\\ +\; & \frac{|10\rangle}{2}\otimes|+\rangle\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle\\ +\; & \frac{|01\rangle}{2}\otimes|+\rangle\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle\\ +\; & \frac{|11\rangle}{2}\otimes|+\rangle\otimes\left(\frac{22}{8\sqrt{8}}|101\rangle-\frac{2}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.\end{aligned}$$

After a second Grover iteration, we obtain:

$$
\begin{aligned}
|\Psi_4\rangle \;=\; & \frac{|00\rangle}{2}\otimes|+\rangle\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|10\rangle}{2}\otimes|+\rangle\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|01\rangle}{2}\otimes|+\rangle\otimes\left(\frac{22}{8\sqrt{8}}|101\rangle-\frac{2}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|11\rangle}{2}\otimes|+\rangle\otimes\left(\frac{13}{8\sqrt{8}}|101\rangle-\frac{7}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.
\end{aligned}
$$

After the third step, all basis states where the first two qubits are in state $|11\rangle$ have undergone three Grover iterations, all basis states where the first two qubits are in state $|01\rangle$ have undergone two Grover iterations, all basis states where the first two qubits are in state $|10\rangle$ have undergone one Grover iteration, and all basis states where the first two qubits are in state $|00\rangle$ have not undergone any Grover iteration. In a fourth step, we perform four Grover iterations controlled by the third counting qubit. After step 4, one can verifiy that the state of the system is given by:

$$
\begin{aligned}
|\Psi_5\rangle \;=\; & \frac{|000\rangle}{\sqrt{8}}\otimes\left(\frac{1}{\sqrt{8}}|101\rangle+\frac{1}{\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|100\rangle}{\sqrt{8}}\otimes\left(\frac{20}{8\sqrt{8}}|101\rangle+\frac{4}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|010\rangle}{\sqrt{8}}\otimes\left(\frac{22}{8\sqrt{8}}|101\rangle-\frac{2}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|110\rangle}{\sqrt{8}}\otimes\left(\frac{13}{8\sqrt{8}}|101\rangle-\frac{7}{8\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|001\rangle}{\sqrt{8}}\otimes\left(\frac{-5}{16\sqrt{8}}|101\rangle-\frac{17}{16\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|101\rangle}{\sqrt{8}}\otimes\left(\frac{-67}{32\sqrt{8}}|101\rangle-\frac{23}{32\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|011\rangle}{\sqrt{8}}\otimes\left(\frac{-181}{64\sqrt{8}}|101\rangle-\frac{1}{64\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle \\
+\; & \frac{|111\rangle}{\sqrt{8}}\otimes\left(\frac{-275}{128\sqrt{8}}|101\rangle+\frac{89}{128\sqrt{8}}(|000\rangle+|100\rangle+|010\rangle+|110\rangle+|001\rangle+|011\rangle+|111\rangle)\right)\otimes|-\rangle.
\end{aligned}
$$

Where basis states with counting qubits in state $|000\rangle$, $|100\rangle$, $|010\rangle$, $|110\rangle$, $|001\rangle$, $|101\rangle$, $|011\rangle$, and $|111\rangle$ have undergone 0, 1, 2, 3, 4, 5, 6, and 7 Grover iterations, respectively. In what follows, we simplify notation, and let $|\mathbf{x}_o\rangle = (|000\rangle + |100\rangle + |010\rangle + |110\rangle + |001\rangle + |011\rangle + |111\rangle)$. As such, $|\Psi_5\rangle$ may be written as:

$$
\begin{aligned}
|\Psi_5\rangle \;=\; & \left(\frac{|000\rangle}{8}+\frac{5|100\rangle}{16}+\frac{11|010\rangle}{32}+\frac{13|110\rangle}{64}-\frac{5|001\rangle}{128}-\frac{67|101\rangle}{256}-\frac{181|011\rangle}{512}-\frac{275|111\rangle}{1024}\right)\otimes|101\rangle\otimes|-\rangle \\
+\; & \left(\frac{|000\rangle}{8}+\frac{|100\rangle}{16}-\frac{|010\rangle}{32}-\frac{7|110\rangle}{64}-\frac{17|001\rangle}{128}-\frac{23|101\rangle}{256}-\frac{|011\rangle}{512}+\frac{89|111\rangle}{1024}\right)\otimes|\mathbf{x}_o\rangle\otimes|-\rangle.
\end{aligned}
$$

In a fifth step, we perform an inverse quantum Fourier transform on the counting qubits (see also Figure 10 for a general implementation of the circuit of the inverse quantum Fourier transform). First, the inverse quantum Fourier transform applies a Hadamard gate to the last counting qubit. As a result, $|t_1 t_2\rangle \otimes |0\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle$

Table 5: Probability amplitude of different basis states and probability to measure any of the basis states after using QCB with $t = 3$ counting qubits to approximate the number of valid solutions in a system that has $n = 3$ qubits and $m = 1$ valid solution. For all basis states of the counting qubits, the table also reports the total probability to measure each of the basis states ($P_{\mathbf{t}}$), the rotation angle of the quantum state ($\theta_{\mathbf{t}}$), and the corresponding number of valid solutions ($m_{\mathbf{t}}$).

$\|\mathbf{t}\rangle$	$\|\mathbf{t}\rangle \otimes \|101\rangle \otimes \|-\rangle$		$\|\mathbf{t}\rangle \otimes \|\mathbf{x}_o\rangle \otimes \|-\rangle$		$P_{\mathbf{t}}$	$\theta_{\mathbf{t}}$	$m_{\mathbf{t}}$
	amplitude	prob.	amplitude	prob.			
$\|000\rangle$	-0.0321	0.0005	0.0218	0.0072	0.0077	0	0
$\|100\rangle$	0.0031	0.0010	0.0321	0.0001	0.0011	3.1416	8
$\|010\rangle$	$0.0090 + 0.0017i$	0.0028	$0.0338 - 0.0411i$	0.0006	0.0034	1.5708	4
$\|110\rangle$	$0.0090 - 0.0017i$	0.0028	$0.0338 + 0.0411i$	0.0006	0.0034	4.7124	4
$\|001\rangle$	$0.1783 + 0.0213i$	0.2651	$0.0836 - 0.5080i$	0.2257	0.4908	0.7854	1.1716
$\|101\rangle$	$0.0040 - 0.0006i$	0.0013	$0.0324 + 0.0150i$	0.0001	0.0014	3.9270	6.8284
$\|011\rangle$	$0.0040 + 0.0006i$	0.0013	$0.0324 - 0.0150i$	0.0001	0.0014	2.3562	6.8284
$\|111\rangle$	$0.1783 - 0.0213i$	0.2651	$0.0836 + 0.5080i$	0.2257	0.4908	5.4978	1.1716

and $|t_1 t_2\rangle \otimes |1\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle$ transform to:

$$|t_1 t_2\rangle \otimes \mathrm{H}\,|0\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle \;=\; |t_1 t_2\rangle \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes |\mathbf{x}\rangle \otimes |-\rangle\,,$$

$$|t_1 t_2\rangle \otimes \mathrm{H}\,|1\rangle \otimes |\mathbf{x}\rangle \otimes |-\rangle \;=\; |t_1 t_2\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes |\mathbf{x}\rangle \otimes |-\rangle\,.$$

For instance, this implies that $512^{-1}\,|011\rangle \otimes |\mathbf{x}_o\rangle \otimes |-\rangle$ transforms to:

$$\left(\frac{|010\rangle}{512\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle\right) - \left(\frac{|011\rangle}{512\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle\right).$$

After applying the Hadamard gate we get:

$$\begin{aligned} |\Psi_{5.a}\rangle \;=\; & \frac{88|000\rangle{+}52|100\rangle{-}10|010\rangle{-}67|110\rangle{+}168|001\rangle{+}588|101\rangle{+}714|011\rangle{+}483|111\rangle}{1024\sqrt{2}} \otimes |101\rangle \otimes |-\rangle \\ & + \frac{-8|000\rangle{-}28|100\rangle{-}34|010\rangle{-}23|110\rangle{+}264|001\rangle{+}156|101\rangle{-}30|011\rangle{-}201|111\rangle}{1024\sqrt{2}} \otimes |\mathbf{x}_o\rangle \otimes |-\rangle. \end{aligned}$$

Next, we apply a controlled phase-shift gate that shifts the phase of the last counting qubit with period $-\pi/2$ if the second counting qubit is in state $|1\rangle$ (in general, the inverse quantum Fourier transform shifts the phase of a qubit q_t with period $-\pi 2^{(t-t')}$ if control qubit $q_{t'}$ is in state $|1\rangle$; see also Figure 10). A phase-shift gate maps $|0\rangle \to |0\rangle$ and $|1\rangle \to e^{i\lambda}\,|1\rangle$, where λ is the period of the shift. As a result, in step 5.a, we map $|0\rangle \to |0\rangle$ and $|1\rangle \to -i\,|1\rangle$ (note that $e^{-i\pi/2} = -i$ in step 5.a). We obtain:

$$\begin{aligned} & |\Psi_{5.b}\rangle = \\ & \left(\frac{88|000\rangle{+}52|100\rangle{-}10|010\rangle{-}67|110\rangle{+}168|001\rangle{+}588|101\rangle}{1024\sqrt{2}} - \frac{714}{1024\sqrt{2}}i|011\rangle - \frac{483}{1024\sqrt{2}}i|111\rangle\right) \otimes |101\rangle \otimes |-\rangle \\ & -\left(\frac{8|000\rangle{+}28|100\rangle{+}34|010\rangle{+}23|110\rangle{-}264|001\rangle{-}156|101\rangle}{1024\sqrt{2}} - \frac{30}{1024\sqrt{2}}i|011\rangle - \frac{201}{1024\sqrt{2}}i|111\rangle\right) \otimes |\mathbf{x}_o\rangle \otimes |-\rangle. \end{aligned}$$

One can verify that, after performing step 5, we obtain $|\Psi_6\rangle$ as given by the probability amplitudes reported in Table 5. Table 5 also reports the probability to measure any of the basis states as well as the total probability to measure any of the basis states of the three counting qubits ($P_{\mathbf{t}}$). For instance, we have a 0.77 percent probability to measure a system where the three counting qubits are in basis state $|000\rangle$. The question now

is: what does this mean? To answer this question, we first determine the rotation angle for each of the basis states of the three counting qubits. The rotation angle is a measure of how much the phase of the quantum state has changed (when compared to the phase of its initial state $|\mathbf{0}\rangle$) and can be obtained as follows:

$$\theta_{\mathbf{t}} = \frac{2\pi}{2^t} \sum_{i=1}^{t} t_i 2^{(t-i)}.$$

Using $\theta_{\mathbf{t}}$, we can also determine the number of valid solutions that corresponds to a basis state $|\mathbf{t}\rangle$ using:

$$m_{\mathbf{t}} = 2^n \left(\sin \frac{\theta_{\mathbf{t}}}{2} \right)^2.$$

The total probability to measure any of the basis states of the three counting qubits gives us the probability to measure a given number of valid solutions. For instance, we are 98.16 percent certain to measure there are 1.1716 valid solutions in a system that has 3 qubits (see also Table 2).

(c) Limitations of exponentiation by squaring

Lemma A.1. *Let* G *denote the operator that performs a single iteration of Grover's algorithm. Exponentiation by squaring cannot be used to obtain* G^{2^i} *in less than* $O(\sqrt{2^n})$ *operations (for all* $i : 0 < i < n/2$*).*

PROOF. Imagine that we are using Grover's algorithm to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries. To maximize the probability to find $\mathbf{x}^*$, we need to perform $I(n, 1)$ Grover iterations. Rather than performing $I(n, 1)$ Grover iterations, however, we can use exponentiation by squaring to perform at most $\lceil \log_2(I(n, 1)) \rceil \approx \lceil \log_2(\sqrt{2^n}) \rceil = n/2 = t$ operations using t operators $G^{2^0}, G^{2^1}, \ldots, G^{2^{(t-1)}}$. For instance, assume $n = 14$. In this case, the corresponding database has 16384 entries, and Grover's algorithm requires $I(14, 1) = 101$ Grover iterations. Rather than performing 101 Grover iterations, we can use exponentiation by squaring, and perform only 4 operations using operators G^{2^0}, G^{2^2}, G^{2^5}, and G^{2^6} (where $2^0 + 2^2 + 2^5 + 2^6 = 101$). In other words, to find $\mathbf{x}^*$, we first need to use exponentiation by squaring to obtain $(n/2 - 1)$ operators G^{2^i} (i.e., one operator for each $i : 0 < i < n/2$). Next, we perform up to $n/2$ operations using operators G and G^{2^i}. If it takes $\sqrt{2^{(\gamma n)}}$ operations (with $0 < \gamma < 1$) to obtain operators G^{2^i}, we require $O(n\sqrt{2^{\gamma n}})$ operations to find $\mathbf{x}^*$. Even if $\gamma \to 1$, $O(n\sqrt{2^{\gamma n}}) < O(\sqrt{2^n})$ for sufficiently large n. This, however, contradicts the result of Zalka (1999), who shows that Grover's algorithm is exactly optimal and that at least $O(\sqrt{2^n})$ calls (and hence operations) are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries (see also Boyer et al. (1996) and Bennett et al. (1997)). As a result, we require at least $O(\sqrt{2^n})$ operations to obtain G^{2^i} (for all $i : 0 < i < n/2$). □

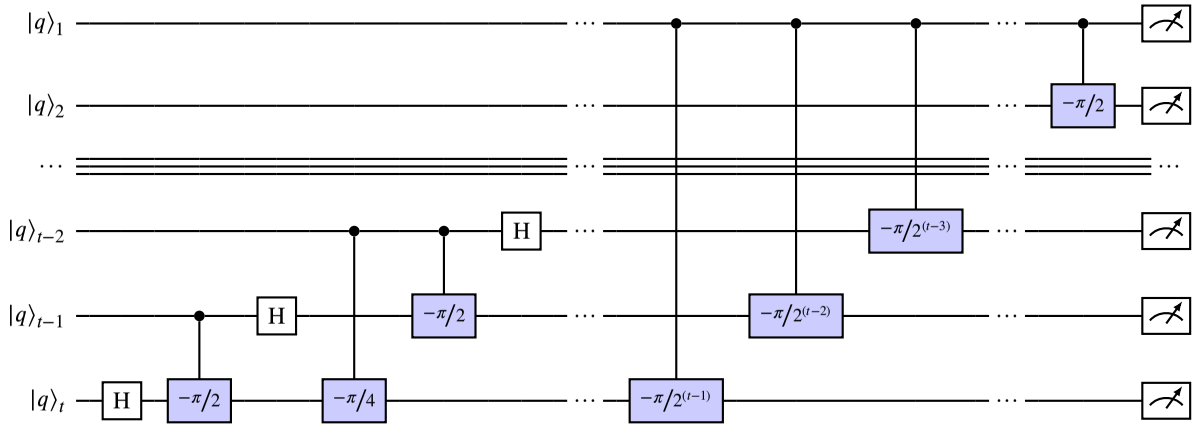
(d) Limitations of quantum counting

Theorem A.1. *Given an unstructured database that has* 2^n *entries, there is no quantum algorithm that can accurately approximate* $P(m > 0)$ *using less than* $O(\sqrt{2^n})$ *calls to a function* $f_{\mathbf{x}}$ *that evaluates whether entry* $\mathbf{x}$ *is the target entry.*

PROOF. Imagine that a quantum algorithm exists that can accurately approximate $P(m > 0)$ using $O\sqrt{2^{\gamma n}}$ calls (with $0 < \gamma < 1$). This algorithm can be used in a binary search procedure that identifies a single target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries using at most $O(log_2(2^n)\sqrt{2^{\gamma n}}) = O(n\sqrt{2^{\gamma n}})$ calls. Even if $\gamma \to 1$, $O(n\sqrt{2^{\gamma n}}) < O(\sqrt{2^n})$ for sufficiently large n. This, however, contradicts the result of Zalka (1999), who shows that Grover's algorithm is exactly optimal and that $O(\sqrt{2^n})$ calls are required to find a single target entry $\mathbf{x}^*$ in a database that has 2^n entries. As a result, any quantum algorithm that can accurately approximate $P(m > 0)$ requires at least $O(\sqrt{2^n})$ calls. □

To illustrate the binary search procedure used in the proof of Theorem A.1, consider a database that has $2^3 = 8$ entries. In a first step of the binary procedure, we evaluate whether target entry $\mathbf{x}^*$ can be found in the first half of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|0 + +\rangle$ and use a quantum algorithm to determine whether

Figure 10: Circuit of the inverse Quantum Fourier Transform ($QFT^{\dagger}$) using t qubits. The blue gates represent controlled phase shift gates that shift the phase of the quantum state of a qubit by a period 2π divided by a power of 2.



$m > 0$ for basis states $|000\rangle$, $|001\rangle$, $|010\rangle$, and $|011\rangle$). If $m > 0$, we enter a second step, and verify whether $m > 0$ in the first quarter of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|00+\rangle$ and evaluate basis states $|000\rangle$ and $|001\rangle$). If, once more, $m > 0$, we enter a third step and verify whether $|000\rangle$ is the target entry (if it is not, $|001\rangle$ is the target entry). If $m = 0$ in the first quarter, we know $m > 0$ in the second quarter of the database, and verify whether $|010\rangle$ or $|011\rangle$ is the target entry. If $m = 0$ in the first half of the database, we verify whether $m > 0$ in the third quarter of the database (i.e., we initialize $|\mathbf{x}\rangle$ as $|10+\rangle$ and evaluate basis states $|100\rangle$ and $|101\rangle$). If $m > 0$, we verify whether $|100\rangle$ or $|101\rangle$ is the target entry. If $m = 0$, we verify whether $|110\rangle$ or $|111\rangle$ is the target entry. In total, the binary procedure requires $\log_2(2^n) = n$ runs of the quantum algorithm. In each run $i : 1 \leq i \leq n$, we evaluate whether $m > 0$. As such, if a quantum algorithm exists that can approximate $P(m > 0)$ using $O(\sqrt{2^{(\gamma n)}})$ calls, the binary search procedure requires $O(n\sqrt{2^{\gamma n}})$ calls to identify target entry $\mathbf{x}^*$ in an unstructured database that has 2^n entries.

References

1. Aaronson S, Rall P. 2020.
   Quantum approximate counting, simplified.
   Symposium on Simplicity in Algorithms, 24–32.
2. Abbas A, Ambainis A, Augustino B, et al. 2024. Challenges and opportunities in quantum optimization, *Nat Rev Phys*, **6**, 718–735.
3. Bennett CH, Bernstein E, Brassard G, Vazirani U. 1997.
   Strengths and weaknesses of quantum computing.
   SIAM J. Comput., **26**(5), 1510–1523.
4. Brassard G, Høyer P, Tapp A. 1998.
   Quantum counting. In Larsen KG, Skyum S, Winksel G (Eds.), *Automata, Languages and Programming*, 820–831.
   Springer Berlin Heidelberg, Berlin, Heidelberg.
5. Brassard G, Høyer P, Mosca M, Tapp A. 2002.
   Quantum amplitude amplification and estimation. In Lomonaco SJ, Brandt HE (Eds.), *Quantum computation and information*, 53–74.
   Amer. Math. Soc., Providence, RI.
6. Bochkarev A, Heese R, Jäger S, Schiewe P, Schöbel A. 2024.
   Quantum computing for discrete optimization: A highlight of three technologies.
   https://arxiv.org/abs/2409.01373.
7. Boyer M, Brassard G, Høyer P, Tapp A. 1996.
   Tight bounds on quantum searching.
   Proceedings of the Fourth Workshop on Physics and Computation, New England Complex Systems Institute.
8. Cerf NJ, Grover LK, Williams CP. 2000.
   Nested quantum search and structured problems.
   Phys. Rev. A, **61**(3), 1–14.
9. Creemers S, Pérez Armas LF. 2025.
   Discrete optimization: A quantum revolution?
   Eur. J. Oper. Res., **323**(2), 378–408.
10. Dürr C, Høyer P. 1999.
    A quantum algorithm for finding the minimum.
    https://arxiv.org/abs/quant-ph/9607014.
11. Grinko D, Gacon J, Zoufal C, Woerner S. 2021.
    Iterative quantum amplitude estimation.
    npj. Quantum Inf., **7**(52).
12. Grover LK. 1996.
    A fast quantum mechanical algorithm for database search.
    Proc. Annu. ACM Symp. Theory Comput., 212–219.
13. Kitaev AY. 1995.
    Quantum measurements and the Abelian stabilizer problem, unpublished preprint at https://doi.org/10.48550/arXiv.quant-ph/9511026.
14. McKinsey. 2021. Quantum computing: An emerging ecosystem and industry use cases, *McKinsey*.
15. McKinsey. 2023. Quantum technology monitor, *McKinsey*.

16. Nielsen M, Chuang I. 2010.
Quantum Computation and Quantum Information: 10th Anniversary Edition.
Cambridge University Press, Cambridge.
17. Shor PW. 1994.
Algorithms for quantum computation: Discrete logarithms and factoring.
Proc. Annu. Symp. FOCS, 124–134.
18. Suzuki Y, Uno S, Raymond R, Tanaka T, Onodera T, Yamamoto N. 2020.
Amplitude estimation without phase estimation.
Quantum Inf Process, **19**(75).
19. Zalka C. 1999. Grover's quantum searching algorithm is optimal, *Phys. Rev. A*, **60**(4), 2746–2751.