Fault-ignorant quantum search

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG | IOP Institute of Physics

Fault-ignorant quantum search

Péter Vrana[1], David Reeb[2], Daniel Reitzner[2,3] and Michael M Wolf[2]
[1] Institute for Theoretical Physics, ETH Zürich, 8093 Zürich, Switzerland
[2] Department of Mathematics, Technische Universität München, D-85748 Garching, Germany
[3] Institute of Physics, Slovak Academy of Sciences, 845 11 Bratislava, Slovakia
E-mail: vranap@math.bme.hu, david.reeb@tum.de, reitzner@savba.sk and m.wolf@tum.de

Received 6 May 2014
Accepted for publication 25 June 2014
Published 24 July 2014
New Journal of Physics **16** (2014) 073033
doi:10.1088/1367-2630/16/7/073033

Abstract
We investigate the problem of quantum searching on a noisy quantum computer. Taking a *fault-ignorant* approach, we analyze quantum algorithms that solve the task for various different noise strengths, which are possibly unknown beforehand. We prove lower bounds on the runtime of such algorithms and thereby find that the quadratic speedup is necessarily lost (in our noise models). However, for low but constant noise levels the algorithms we provide (based on Grover's algorithm) still outperform the best noiseless classical search algorithm.

Keywords: quantum search, algorithms, fault tolerance, Grover search, computational complexity, error correction, runtime bounds

1. Introduction

Since the inception of quantum computing [1], a large effort has been devoted to making quantum computers function in a noisy environment and securing them against imperfections in the physical setup itself. The theoretical literature offers several ways to cope with such errors. The leading idea is to encode quantum states into a larger system [2, 3] such that noise hits can be recognized and subsequently corrected. A quantum computation can then be performed in a fault-tolerant way directly on these encoding systems [4, 5], and nested levels of error correction can make the computation error-free. The last statement assumes that the initial error rate is not too high and that the error hits are not too correlated, e.g. occur locally. Beyond these assumptions, quantum error correction schemes require significant resources in terms of circuit size and experimental control [6, 7]. Other approaches use decoherence-free subspaces, employing for the quantum computation those parts of a system's Hilbert space which are relatively unaffected by the noise [8, 9]. The latter approach works only in more limited circumstances and requires detailed knowledge of the noise.

Here, we follow a different idea which tries to avoid the disadvantages just mentioned. Rather than devoting large efforts to eliminating the errors at all costs, we accept them in the computation and try to design algorithms that eventually still find the desired result. The spatial size of the quantum circuit should however not be enlarged much beyond the noiseless version of the algorithm; one may e.g. allow only a number of extra qubits that remains bounded by a constant as the problem size becomes large [7] (whereas it seems reasonable to allow for an exponentially large noiseless classical memory).

Furthermore, the algorithms should find the desired result under any level of background noise—this level may actually be unknown to the algorithm, hence the term *fault-ignorant computation*. The algorithm is, however, allowed to take more time the larger the actual noise gets. In this sense, we are trading spatial resources (circuit size) for temporal resources (runtime). Still, when the actual noise level is low (but constant in the problem size), we want our algorithms to produce the desired result faster than any classical algorithm even in a noiseless environment—this indeed will be the case for the explicit algorithms we present. We describe the fault-ignorant approach and these desiderata in more detail in section 1.1.

While under noise the resulting fault-ignorant algorithms may not give the full quantum speedup for large problem sizes, they may still be useful for initial and medium-term realizations of quantum computers, in particular in the non-scalable low-qubit number regime which does not allow for full-blown quantum error correction schemes.

In this paper, we analyze the above *fault-ignorant idea* for the example of quantum search on a noisy quantum computer. The unstructured search problem, i.e. the search for a marked item in an unordered list with oracle access, can be solved on a noiseless quantum computer by Grover's algorithm quadratically faster than by any classical algorithm [1, 10]. This quantum speedup is optimal [11–13].

Good algorithms or optimality issues for the noisy case are however much less clear. Many studies in this direction investigate how (different models of) background noise or a faulty oracle affect specifically Grover's algorithm (e.g. [14–19]), and it is often found that the quadratic speedup persists as long as the noise stays below a certain threshold depending on the database size. Only the paper [20] by Regev and Schiff gives, for a specific model of a faulty oracle, a general lower bound on the number of oracle invocations necessary to find the marked item by *any* quantum algorithm; [21] proves the analogous result for any continuous-time implementation of oracle search. A concrete algorithm that functions under a faulty oracle is briefly suggested in [19], advocating the avoidance of active error correction as well, as we do.

In the quantum search part of this paper, we address the strengths and weaknesses of general oracular search algorithms under noise in more detail. Notably, we state all our upper and lower runtime bounds with explicit prefactors as e.g. in theorems 2–5, which in particular allows a comparison with the performance of classical search algorithms already for 'small' (i.e. non-asymptotic) database sizes *N*. On the one hand we give fault-ignorant algorithms, on the other we investigate their optimality. More detailed comparisons to the works [19–21] are made in section 3.3. The Hilbert space in our setup consists of a search space, possibly supplemented by an ancillary quantum system and both suffering from noise; our constructive algorithms will actually not use ancillary systems, but we allow for them in the proofs of our lower runtime bounds, which strengthens these. Additionally, the algorithms may have access to a noiseless classical memory, which is a technologically realistic assumption. The noise itself is modelled as discrete hits of some noise channel that is to be applied between any two oracle invocations.

The problem of fault-ignorant quantum search is then as follows: devise a quantum algorithm that, except for some specified maximal failure probability ε, returns the marked item under any decoherence rate p, using as few oracle calls as possible. The fastest (noiseless) classical algorithm needs $\lceil (1 - \varepsilon) N \rceil$ table lookups for this task, examining the database entries one by one. We exhibit quantum algorithms which, under low but constant depolarizing level p, require fewer oracle calls than this classical algorithm, see e.g. theorem 3.

The paper is organized as follows. We give a more detailed description of the fault-ignorant idea in section 1.1, while referring to appendix A for a general and precise mathematical definition of *fault-ignorant algorithms*. Those readers interested mainly in the problem of quantum search on a noisy quantum computer are directed to the remaining sections. In section 2 we introduce quantum search in the presence of decoherence, and develop a fault-ignorant algorithm (algorithm 1) that consists of 'quantum building blocks' and uses the noiseless classical memory merely to store the marked item in case of a previous successful round. A matching lower bound on the runtime of such algorithms is given by theorem 2. We expand this analysis in section 3 and allow for a noiseless classical memory that can store all previously falsified items, which enhances search efficiency (algorithm 2). In appendix B we discuss in more detail the noise models for which our results from the main text apply.

1.1. Fault-ignorant algorithms—definition and basic properties

Here we describe the main desiderata on fault-ignorant algorithms, with particular emphasis on oracular algorithms that are the main topic of the following sections. For a precise mathematical formalization of fault-ignorance, which also includes algorithms computing probabilistic functions as in sampling problems and computational algorithms such as factoring, we refer the reader to appendix A (in particular, definitions 1 and 2 therein).

The tasks we consider consist in the computation of a classical output $o \in O$ as a function of an oracle index $x \in X$ and an input $i \in I$, which we assume is given via a specified encoding ρ_{i} in the quantum register at the start of the algorithm. The specification of the task contains already the available size of the quantum register, i.e. the Hilbert space $\mathcal{H}$ on which the ρ_{i} acts and which is assumed to be fixed throughout the computation; in early implementations of quantum computers this size may be severely limited and is thus assumed to be part of the problem specification. Besides the quantum register, we allow for a classical register that may serve several purposes: *(i)* to be used during the computation, *(ii)* to store the output, *(iii)* to hold a binary flag indicating whether the output has already been written into the register, such that it can be read out by an outside agent without disturbing the (quantum) computation that may still be ongoing (such an indication is necessary since the noise level and the algorithm runtime may not be known in advance, see below).

For the task of oracle search among N items, we thus have $x \in \{1, \ldots, N\}$ and would like to produce the output $o = x$, whereas the quantum register can be initialized in any fixed state ρ_0 as there is no other input to the task, i.e. $I = \{0\}$. We can allow for any quantum register $B(\mathcal{H})$; $\mathcal{H} = \mathbb{C}^N$ would for example enable us to perform Grover's algorithm using the oracle equation (5) (see section 2.1), but we may also allow for additional quantum registers such that e.g. $\mathcal{H} = \mathbb{C}^N \otimes \mathbb{C}^M$.

We have not yet specified in what way fault-ignorant algorithms should behave with respect to noise. For this we need two more ingredients. The first is a family of noise channels, denoted by D_p and acting only on the quantum register, that should model the effect of the noise per unit time (see also appendix B), and we think of the index $p \in [0, 1]$ as a noise strength parameter that is *not* known to the agent executing the algorithm. Classical registers are assumed noise-free. For our results on noisy quantum search in the following sections, we will for example choose the noise models in equations (7) or (8). The second ingredient is the specification of the set S of operations that an algorithm can perform per unit time. S may be any subset of all quantum channels acting jointly on the quantum and classical register. Actually, for oracular algorithms, each element $T \in S$ also depends on the oracle index x that is not directly accessible to the algorithm; rather, when an algorithm 'calls' the operation T, then the quantum channel $T(x)$ is executed. Again for the oracular search case below, we allow any

$$T(x) = C_2 \circ O_x \circ C_1 \qquad (\text{for all } x \in X) \tag{1}$$

that calls the oracle O_x from equation (5) once, possibly preceded and followed by any quantum channels C_1, C_2 acting on the quantum and classical registers (and not depending on x). For oracular problems, such a specification of the allowed operations per unit time is quite natural, but other choices are possible; for computational problems one can for example impose locality restrictions on the operations per time step, such as in example 3 in appendix A. The specification of what constitutes one time step will be implicit in equation (2) below, saying that the noise channel D_p is to be applied once between any two operations from S.

An *algorithm* is now simply a sequence $(T_n)_{n\in\mathbb{N}}$ of operations $T_n \in S$. Algorithms may depend on an accuracy parameter $\varepsilon \in (0, 1)$, denoting the maximum probability with which we allow a wrong answer to be output.

Crucially however, for a *fault-ignorant algorithm* $(T_n)_n$, we *do not* allow the operations T_n to depend on the above noise level p, but still require that, for *each* noise level $p \in [0, 1]$, after the execution of any number $t \geqslant t(p)$ of time steps

$$T_t(x) D_p T_{t-1}(x) D_p \ldots D_p T_2(x) D_p T_1(x) \left(\rho_i \otimes |0\rangle_{\text{class}}\langle 0|\right), \tag{2}$$

the classical output register holds the correct output o, up to failure probability ε (see also figure A1 in appendix A for illustration). We call an algorithm $(T_n)_n$ *fault-ignorant* if $t(p)$ can be chosen finite for all $p \in [0, 1]$, and we will actually denote by $t(p)$ the smallest time such that the previous requirement is satisfied for all $t \geqslant t(p)$ (see definition 2 in appendix A for an exact statement). In other words, a fault-ignorant algorithm is ignorant of the noise level p, but should nevertheless output the correct answer o irrespective of the noise p, within a time $t(p) < \infty$ that may depend on the unknown noise level p.

Note that any computation which can be performed in a noiseless classical register of sufficient size alone constitutes a fault-ignorant algorithm, as we assume classical memory to be unaffected by the noise.

As a further instructive example of a fault-ignorant algorithm, assume the following: *(a)* $(T_n)_{n=1}^k$ is a finite-step algorithm such that, after executing the sequence (2) for k time steps, there is for *any* noise level $p \in [0, 1]$ a non-zero success probability $p_s(p) > 0$ of obtaining the correct output o in the classical register, *(b)* the set S contains operations that allow us to check whether a tentative result o' in the classical register is correct, *(c)* the input encoding ρ_i is 'classical' in the sense that there are operations in S which can extract from ρ_{i} the index i onto a classical register, and conversely allow preparation of the quantum state ρ_{i} given the classical value of i (requirement *(c)* is easy when $|I| = 1$, as in the search task). If these three conditions hold, one can construct a fault-ignorant algorithm that solves the task for any fixed accuracy parameter $\varepsilon > 0$, as follows:

- repeatedly run the k-step sequence $(T_n)_{n=1}^{k}$ with noise interspersed as in (2) on the initial state $\rho_i \otimes |0\rangle_{\mathrm{class}}\langle 0|$ (note that, after the value i has been saved in a classical register at the start, this initial state can be reproduced before each iteration due to condition *(c)*),
- after each iteration, check for the correctness of the answer proposed in this iteration according to *(b)*, and store the correct result in an overall output register once it has been found.

If the actual noise level is p, then after r iterations of $(T_n)_{n=1}^{k}$, the failure probability equals $(1 - p_s(p))^r$, which will become smaller than any specified $\varepsilon \in (0, 1)$ if only $r \geqslant r(p)$ is large enough. More precisely, since each round including verification consumes $k + 1$ time steps, the composite check-and-repeat algorithm has a runtime

$$t(p) = (k+1)\left\lceil \frac{\log \frac{1}{\varepsilon}}{\log \frac{1}{1 - p_s(p)}} \right\rceil. \tag{3}$$

For example, in searches using the quantum oracle (5), one can satisfy the above conditions *(a)*–*(c)* by randomly selecting in each iteration an index $o' \in \{1, \ldots, N\}$ (see also the beginning of section 3.1) and then using the oracle O_x once to check whether $o' = x$. In this example, $k = 0$ and $p_s(p) = 1/N$ for all p. For another similar example, see the beginning of section 2.4. In fact, all the constructive algorithms we present in this paper will be variations of the above check-and-repeat algorithm, with possibly varying numbers k_g of oracle uses in each round $g = 1, \ldots, r$ (section 2.4, see figure 2 for illustration) and possibly leaving out previously falsified items in future rounds (section 3, see also figure 3).

Due to this simple way of constructing fault-ignorant algorithms, the meaningful question, which we investigate in the following sections for noisy quantum search, is about the efficiency of fault-ignorant algorithms. Notice for example that the first algorithm proposed in the previous paragraph has a runtime of $t(p) \approx N \log \frac{1}{\varepsilon}$, proportional to the problem size N and *independent* of $p \in [0, 1]$. However, one might hope that there exist fault-ignorant algorithms which for small actual noise level p need fewer oracle calls, because at least when the noise is *known* to vanish (i.e. $p = 0$) then Grover's algorithm solves the problem with at most roughly $\frac{\pi}{4}\sqrt{N} \log \frac{1}{\varepsilon}$ oracle calls (see section 2.1). Furthermore, both these runtimes diverge in the limit of perfect accuracy, i.e. $\varepsilon \to 0$, whereas the classical algorithm checking the N items one after another needs only a finite number N of steps for perfect accuracy. In fact, in the following sections we will develop a fault-ignorant quantum search algorithm which, under the noise models (8) and up to a constant overall factor, for any N, p, ε requires fewer oracle calls than the ones just mentioned (see theorem 4).

When given a fault-ignorant algorithm solving one specific task (e.g. one specified problem of size N, of specified quantum register size $\dim(\mathcal{H})$, noise model $\mathcal{N}_p$, and accuracy goal ε), one can compare its runtime $t(p)$ to the runtime of other algorithms that can be implemented (e.g. to the classical search algorithm above, or to any algorithm that 'knows' the noise level p as one of its inputs, or to any algorithm that may use a quantum register of some larger size, etc). On the basis of this comparison one can then decide whether to consider this fault-ignorant algorithm useful w.r.t. the competitor. Instead of solely the runtime, one may also take into account other factors in this comparison, such as the size of the quantum register used by either algorithm. It does not seem possible to give general criteria for such a decision. However, due to the prefactors given in the runtime bounds for our concrete algorithms, one can for example use our theorems 3 and 4 to compare these fault-ignorant algorithms in such a way to other algorithms (which we for example do below in theorem 3 and section 3.3).

Instead of performing such a comparison for one specific task, one may consider a whole family of tasks (e.g. one for each problem size $N \in \mathbb{N}$ and accuracy $\varepsilon \in (0, 1)$, possibly also allowing the size of the quantum register to vary independently of N, etc) and fault-ignorant algorithms solving them. In this situation one can then investigate the *scaling* of the runtime $t(p)$ with these parameters N, ε, etc, as is usual in complexity theory, and investigate various tradeoffs, e.g. between the runtime and the size of the quantum register. Again, the concrete questions seem to depend highly on the tasks at hand.

Nevertheless, since the main feature of fault-ignorant algorithms is to find the correct answer *without knowing* the noise level p in advance, we can introduce a distinguished notion of *efficient fault-ignorant algorithm*: a family of fault-ignorant algorithms (each solving a task from a given family of tasks) is called *efficient* if there exists a constant $C > 0$ such that for each fixed $p \in [0, 1]$ the runtime $t(p)$ (which depends on the member of the family) is at most a factor of C larger than the runtime of any algorithm that solves the same task while *knowing* the noise value p as one of its inputs (i.e. need *not* be fault-ignorant). In other words, we call a fault-ignorant algorithm *efficient* if knowing the actual noise level p would shorten its runtime by at most an overall factor $1/C$. Our theorem 5 can thus be seen as a statement that algorithm 2 is efficient w.r.t. to the class of algorithms considered and, furthermore, an upper bound on the constant C is apparent together with theorem 4. Independently of this efficiency notion, for low enough (but unknown) noise level p, the runtime of algorithm 2 compares favorably with noiseless classical search (see section 3.3).

2. Quantum search under noise—memoryless approach

2.1. Setup: noiseless and noisy quantum search

The quantum search problem [1, 11] asks for an algorithm of short runtime to identify (up to some small error probability ε) one out of N oracles, i.e. to return the index x of the 'black box' implementing the unitary transformation

$$\widetilde{O}_x : \mathbb{C}^N \otimes \mathbb{C}^2 \to \mathbb{C}^N \otimes \mathbb{C}^2, \quad |x', b\rangle \mapsto \left|x', b \oplus \delta_{xx'}\right\rangle, \tag{4}$$

where $x \in \{1, \ldots, N\}$ and $\oplus$ denotes addition modulo 2. Here, we assume the oracle x to have been chosen uniformly at random (often referred to as unstructured database). It is customary to take the input of the oracle $\widetilde{O}_x$ at each step to be of the product form $|\varphi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ so that the output is also of this form with the *sign* of the coefficient of $|x\rangle$ flipped in the first tensor factor $|\varphi\rangle$. In this case one can neglect the second subsystem and concentrate on the effective unitary transformation

$$O_x : \mathbb{C}^N \to \mathbb{C}^N, \quad |x'\rangle \mapsto (-1)^{\delta_{xx'}}|x'\rangle. \tag{5}$$

As usual, we measure the runtime of an oracular algorithm by counting the number of queries (oracle uses), which are viewed as the expensive or time-consuming operations, and disregard all the other quantum channels which are independent of the oracle.

Grover [10] found a solution to this problem which makes use of the equal superposition state $|\psi\rangle \equiv \frac{1}{\sqrt{N}}\sum_{x=1}^{N}|x\rangle$—this state reflects the initial lack of knowledge about the oracle and will be used frequently in the following. At the beginning of the algorithm the state $|\psi\rangle$ is prepared and then the oracle black box O_x and the unitary $I - 2|\psi\rangle\langle\psi|$ are applied alternately. (In this description, and in our whole paper, we disregard any subsystem structure of $\mathbb{C}^N$; if, for example, $\mathbb{C}^N = (\mathbb{C}^2)^{\otimes n}$, then the 'inversion about the mean' $I - 2|\psi\rangle\langle\psi|$ can be implemented efficiently in the number n of qubits [1, 10]). After k applications of both operators, a von Neumann measurement is performed in the standard basis. The outcome of this measurement will give the correct index x of the oracle with probability [1]

$$\sin^2\left((2k+1)\arcsin\frac{1}{\sqrt{N}}\right), \tag{6}$$

independently of which oracle $x \in \{1, \ldots, N\}$ was implemented by the black box. In particular, if we choose $k = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$, the success probability is $1 - O(N^{-1/2})$. Alternatively, if we fix some small maximal error probability $\varepsilon \gtrsim N^{-1/2}$, with which the algorithm may return an incorrect oracle index, then we may stop after $k = \left\lceil \frac{1}{4}\sqrt{N}\arccos(2\varepsilon - 1) \right\rceil$ iterations. In fact, it can be shown, for any $0 \leqslant k < \frac{\pi}{4}\sqrt{N}$, that Grover's algorithm yields the highest probability of success which can be achieved by any quantum algorithm using the oracle k times [13].

The above analysis is valid when the unitaries and measurements etc can be implemented perfectly and the quantum computer is not subject to noise. In more realistic settings, however, these idealizations have to be lifted, and some such extensions have been considered in the literature before, cf section 1. In this work, we consider the specific case where the quantum computer is continually affected by noise, e.g. coming from the environment.

Our aim is not to approach this problem by implementing quantum error correction, which may be expensive in terms of the required control and size of the quantum computer. Rather, we aim to find (optimal) algorithms which succeed even under the influence of—known, or ideally even *unknown*—noise, in such a way that their runtime may depend on the noise level; see section 1.

Throughout this paper the term 'noise' will mean the application of a certain quantum channel to the state of the quantum register in discrete time steps. This is supposed to model, within the discrete-time setting of oracle algorithms, that the quantum computer is continually affected by noise. More precisely, we will impose that the noise channel has to act *once between any two invocations of the oracle*.

Our paradigmatic example of noise will be the family of *partial depolarizing* channels

$$D_p(\varrho) := p\frac{I_d}{d}\,\mathrm{Tr}\,(\varrho) + (1-p)\varrho\,, \tag{7}$$

acting on states ϱ on a d-dimensional Hilbert space. We can interpret these channels as acting on the system in a completely depolarizing way if a biased coin toss yields heads, which happens with probability $p \in [0, 1]$, and otherwise leaving the quantum computer undisturbed. Intuitively speaking, the partially depolarizing noise (7) discards the whole quantum register with probability p between any two successive oracle invocations. In particular, quantum error correction cannot be applied to this noise model (cf appendix B); but so it serves to illustrate our idea of *fault-ignorant computing*, one of whose rationales actually is to avoid costly error correction procedures. In appendix B we will further introduce partially dephasing noise (B2), which has an additional interpretation as modelling the transitioning from quantum to classical algorithms, and we relate the different noise models.

The lower bounds on the runtime of noisy quantum search algorithms which we prove (theorems 2 and 5) rely on partial depolarizing (7), which is a very drastic and in some implementations quite pessimistic noise model, as it acts in a strongly correlated way across the whole quantum register (somewhat similar to a noisy oracle [19–21], see also section 3.3). For initial implementations of quantum computing this may in some cases indeed be a reasonable assumption, e.g. when p denotes the occurrence probability of a noise event requiring the restarting of the whole quantum computer. The noise strength p could for example be related to the timescale of a drifting laser or of collective hits by external stray magnetic fields. Nevertheless, it would be desirable to prove similar lower runtime bounds for weaker noise models, in particular incorporating some kind of locality, but for now our bounds provide at least a (pessimistic) point of comparison.

On the other hand, the concrete algorithms we provide will function with the guaranteed upper runtime bounds given in theorems 3 and 4 even under any more general noise of the form

$$D_p(\varrho) := pT(\varrho) + (1-p)\varrho\ , \tag{8}$$

with T being an arbitrary quantum channel; i.e. it is necessary only with probability $(1-p)$ that at each step no fatal noise event occurs. Partial depolarizing and partial dephasing are special cases of this.

2.2. Building block for noisy search algorithms

Let us see how the probability of a successful measurement, i.e. returning the correct oracle index x, looks when we include an application of the noise channel after each query. As a preparation, we first consider Grover's algorithm under noise. Introducing the channel $G_x(\varrho) = ((I - 2|\psi\rangle\langle\psi|)O_x)\varrho((I - 2|\psi\rangle\langle\psi|)O_x)^\dagger$, the final state can be written as $(D_pG_x)^k(|\psi\rangle\langle\psi|)$, so that the success probability is then

$$p_s(N, k, p) := \sum_{x=1}^{N}\frac{1}{N}\langle x|\left(D_pG_x\right)^k(|\psi\rangle\langle\psi|)|x\rangle, \tag{9}$$

where we take an average over the N possible oracles, since the search is unstructured and we assume equal a priori probabilities. In this paper, we choose to consider the *average success probability* of algorithms, i.e. averaged over all possible oracles with equal weight (see e.g. [13]), as opposed to the minimal success probability, i.e. minimized over all oracles (e.g. [1, 12]). Both figures of merit agree for 'symmetric algorithms', e.g. for Grover's algorithm [13] and for the constructive algorithms we propose in this paper. But our choice strengthens the lower bounds derived in the following on the required number of oracle invocations.

Now, for all above noise models $D_p = pD_1 + (1-p)\text{id}$ (see equation (8)), the evolution $(D_pG_x)^k$ can be written as a sum of 2^k histories. Since each term gives a non-negative contribution to the sum, we can find a lower bound by keeping only the noise-free term $(1-p)\text{id}$ in each factor:

$$p_s(N, k, p) \geqslant (1-p)^k \sin^2\left((2k+1)\arcsin\frac{1}{\sqrt{N}}\right), \qquad (10)$$

which is quite sharp unless $kp \gg 1$, cf appendix B; compare this also to noiseless case, equation (6). (The convention $0^0 = 1$ is understood throughout this paper.)

We are now interested in how well this simple algorithm, and other algorithms that we shall consider below (i.e., not necessarily consisting of Grover steps), perform. That is, we would now like to derive *upper* bounds on the success probability p_s depending on N, k, and p. As the starting point we use the implicit bound on p_s derived by Zalka [13] for the average success probability p_s after k oracle calls:

$$2N - 2\sqrt{N}\sqrt{p_s} - 2\sqrt{N}\sqrt{N-1}\sqrt{1-p_s} \leqslant 4k^2. \qquad (11)$$

This bound has been established in [13] for the following situation: we start from any pure state $|\phi\rangle \in \mathbb{C}^K$; the oracle O_x $(1 \leqslant x \leqslant N)$ inverts the coefficients of the basis states within a subset $S_x \subseteq \{1, \ldots, K\}$ where $S_x \cap S_y = \varnothing$ for $x \neq y$; we let arbitrary unitaries $U_1, \ldots, U_k \in \mathbb{C}^{K\times K}$ act after each oracle use; in the end we perform a von Neumann measurement in some basis, and our guess for x is an arbitrary function of the measurement outcome.

The same bound (11) holds thus when we start from any mixed state over $\mathbb{C}^N \otimes \mathbb{C}^M$ $(M \geqslant 1)$, apply arbitrary channels between the oracle uses, and our guess for x comes from measuring a POVM $(E_x)_{x=1}^N$. This holds because mixed states, quantum channels and POVM measurements can be dilated to pure states, unitary evolutions and von Neumann measurements on a larger system [1], and the oracles (5) tensored by the identity on all other subsystems still invert coefficients of disjoint sets of basis states.

Even for the more general algorithms described above, we can thus use inequality (11) together with lemma 1 (see appendix D), to prove the following *explicit upper bound* on the average success probability p_s of any quantum algorithm using k oracle calls (with or without noise):

$$p_s \leqslant \frac{(2k+1)^2}{N}. \qquad (12)$$

This bound does not depend on the noise strength p, and thus gives no further restrictions on *noisy* search compared to the noiseless case. But it does enable us to prove a result on the limitations of algorithms employing a noisy quantum register $\mathbb{C}^N \otimes \mathbb{C}^M$ for computation, with the oracle acting on the first subsystem. Note that the computational steps T_i, T_i' in any algorithm covered by the following upper bounds on the success probability are nowhere assumed to be necessarily unital. If they all were unital, then after the occurrence of a noise hit of partial depolarizing (corresponding to the term $p\,\mathrm{Tr}\,(\varrho)I_d/d$ in (7)) the success probability of finding the marked item would be fixed at $1/N$, whereas non-unital actions might try to correct a partial depolarizing error and increase the success probability (we will comment on a particular non-unital error-detection-and-correction strategy below equation (30)). And, actually, the following result holds even for noise channels D_p^τ that may be somewhat more general than the partial depolarizing given in equation (7):

Theorem 1. *(Bound on success probability of building block). Let* $\varrho_0 \in \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M)$ *be the initial state of the algorithm,* $\widehat{O}_x\colon \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M) \to \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M)$ *defined by* $\widehat{O}_x(|y\rangle\langle y'| \otimes \sigma) = (-1)^{\delta_{xy}+\delta_{xy'}}|y\rangle\langle y'| \otimes \sigma$ *be the quantum channels implementing the oracles* *on the first subsystem, and* $D_p^\tau(\varrho) := p\tau \operatorname{Tr} \varrho + (1-p)\varrho$ *the noise channel that is to be applied between any two oracle calls, with* $p \in [0, 1]$ *and* $\tau \in \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M)$ *any quantum state. Let* $T_1, T_1', T_2, T_2', \dots, T_k, T_k' \colon \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M) \to \mathcal{B}(\mathbb{C}^N \otimes \mathbb{C}^M)$ *denote steps in the algorithm, such that the state after k uses of the oracle* $\widehat{O}_x$ *is*

$$\varrho_k^x = T_k' \widehat{O}_x T_k D_p^\tau \dots D_p^\tau T_2' \widehat{O}_x T_2 D_p^\tau T_1' \widehat{O}_x T_1 D_p^\tau(\varrho_0), \tag{13}$$

and let the final measurement be given by the POVM $(E_y)_{y=1}^N$. *Then the average success probability of this algorithm is upper bounded as follows:*

$$p_s := \sum_{x=1}^{N} \frac{1}{N} \operatorname{Tr}\left[\varrho_k^x E_x\right] \leqslant \frac{1}{N} + \frac{8}{Np^2}\,, \tag{14}$$

and

$$p_s \leqslant 8\frac{k+1}{Np}\,. \tag{15}$$

Proof. Introduce the following states:

$$\sigma_i^x := T_k' \widehat{O}_x T_k T_{k-1}' \widehat{O}_x T_{k-1} \dots T_{k-i+1}' \widehat{O}_x T_{k-i+1}(\tau) \tag{16}$$

for $1 \leqslant i < k$, and $\sigma_k^x := T_k' \widehat{O}_x T_k \dots T_1' \widehat{O}_x T_1(\varrho_0)$. With these we can write

$$\varrho_k^x = \sum_{i=1}^{k} p(1-p)^{i-1} \sigma_i^x + (1-p)^k \sigma_k^x\,, \tag{17}$$

and hence

$$Np_s = \sum_x \operatorname{Tr}\left[\varrho_k^x E_x\right] = \sum_{i=1}^{k} p(1-p)^{i-1} \sum_x \operatorname{Tr}\left[\sigma_i^x E_x\right] + (1-p)^k \sum_x \operatorname{Tr}\left[\sigma_k^x E_x\right]. \tag{18}$$

As the 'computation' of σ_i^x involved i oracle calls, from (12) we have $\sum_x \operatorname{Tr}\left[\sigma_i^x E_x\right] \leqslant (2i+1)^2$, and thus

$$\begin{aligned} Np_s &\leqslant \sum_{i=1}^{k} p(1-p)^{i-1}(2i+1)^2 + (1-p)^k(2k+1)^2 \\ &= 1 + \frac{8}{p^2}\left(1-(1-p)^k\right) - \frac{8}{p}k(1-p)^k\,, \end{aligned} \tag{19}$$

which trivially leads to (14). Furthermore, we can use $(1-p)^k \geqslant 1 - kp$ from the Bernoulli inequality to obtain (15):

$$Np_s \leqslant 1 + \frac{8}{p^2}kp \leqslant 8\frac{k+1}{p}\,. \tag{20}$$

□

In the following text we shall address a sequence of operations as in (13) acting on an initial state ϱ_0 as **Algorithm 0** (or $\mathbf{Alg}^0$), which is also depicted in figure 1.

It is worthwhile to mention two ways of using bounds as in theorem 1:

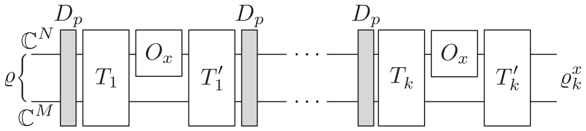

Figure 1. Algorithm 0, performing k steps of quantum search. Noise is acting between any two oracle invocations O_x (see equation (5)), before and after which one may however apply arbitrary channels T_i, T_i'. The computation uses a quantum register of dimension N, on which the oracle acts, and an ancillary (quantum) register $\mathbb{C}^M$, both noisy. A measurement, described by some POVM (E_y), is applied to the final state ϱ_k^x to guess the marked element x.

(i) one can consider the noise level p to be fixed and examine the scaling behaviour (e.g. of the algorithm runtime) with respect to the number N of search items, or

(ii) one can consider p to scale in some way with N.

Point (ii) provides a way to compare the results with other works (e.g. [19]) where this kind of scaling was analyzed. Our results, however, apply to any values of N and p (and later, of ε) in the stated ranges, but we are implicitly often imagining the case $N \to \infty$, p = const (and ε = const), which is a sensible limit as explained in appendix B.

Theorem 1 has two important implications. The first is that, for any fixed $p > 0$, the growth of the success probability is at most *linear* in the number k of queries, as opposed to the quadratic growth in the noiseless case (cf (6) for $k \ll \frac{\pi}{4}\sqrt{N}$). This may come as a surprise as one might have guessed that the quadratic speedup of Grover's algorithm may persist for small enough noise levels $p > 0$ (i.e., fixed p and $N \to \infty$). In other words, theorem 1 says that there exists *no* algorithm with success probability $p_s \sim k^2$ whenever partial depolarizing acts. The inequality (15) in particular implies that quantum error correction *cannot* be done for partially depolarizing noise D_p with $p > 0$.

The second implication is that the success probability is bounded by $\frac{1}{N} + \frac{8}{Np^2}$ *independently of* k, cf (14). For growing $N \to \infty$, this goes to 0 unless $p = O(N^{-1/2})$; in general we cannot reach a prescribed success probabilty $1 - \varepsilon$ with an algorithm as described in theorem 1. The straightforward solution to this problem is that we repeat the algorithm (including the final measurement) until the probability of failure in all the repetitions combined drops below ε. This strategy is detailed in the following section and shows the basic structure for all further algorithms.

2.3. Search algorithm by repeating the basic building block—known noise level p

Now we consider algorithms that consist of a number of repetitions (rounds) of the basic building block described in the previous section: repeatedly preparing states like (13), using the oracle O_x a number of times, and trying to infer the oracle index x by a measurement. In our concrete constructions we shall, as in Grover's algorithm [1, 10], specify the channels T_i' in (13) to perform unitary inversions $(I - 2|\psi\rangle\langle\psi|)$ about the mean, and T_i to equal the identity channel (whereas for our lower bounds T_i, T_i' remain arbitrary).

Note that these repetitions require a *noiseless* classical memory in order to reliably store the correct result x after one of the rounds has been successful (any noisy classical register may be part of the subsystem $\mathbb{C}^M$ in theorem 1 and is subject to noise D_p). Furthermore, in order to test whether the measurement after any one of the repetitions has returned the correct index x

and in order not to overwrite the correct result in subsequent rounds, one needs to perform a verification of the measurement outcome—using *one* oracle call or classical table lookup—immediately after each measurement.

In this section we will first develop and analyze an algorithm that finds, except for some specified failure probability $\varepsilon > 0$, the marked element x among N elements, on a noisy quantum computer. Secondly, we will prove (theorem 2) that, up to a constant, the runtime of this algorithm is optimal among a certain class of algorithms, when the noise level p is known in advance.

Our algorithm will consist of m rounds of the procedure described in theorem 1, each time checking whether the concluding measurement gave the correct x and, if so, storing the result in a noiseless classical register. Specifically, in each round we perform Grover's algorithm for some number k (to be determined) of steps—this has been described at the beginning of section 2.2. We now first give a motivating 'derivation' of the algorithm.

Clearly, the events 'the noise did not hit and the measurement was unsuccessful in the ith round' and 'the noise did not hit and the measurement was unsuccessful in the jth round' are independent for $i \neq j$ when we use Grover iterations, and similarly for more than two rounds, since Grover's algorithm is symmetric with respect to permutation of the oracles (the probability for the complement of each such event is given by (10), which we will substitute below for $1 - p_s$). This means that if we perform m rounds with k Grover iterations in each round, then the probability of failure and the total number of queries will be $(1 - p_s(N, k, p))^m$ and $(k + 1)m$, respectively (including the verification step after each of the m measurements). If we are to reach the target error bound ε in the least number of steps, we need to minimize $(k + 1)m$ subject to the condition $(1 - p_s(N, k, p))^m \leqslant \varepsilon$. The latter gives a lower bound on m depending on N, k and p, namely

$$(k+1)m \geqslant \frac{k+1}{-\log\left(1 - p_s(N, k, p)\right)} \log\frac{1}{\varepsilon}. \tag{21}$$

This means that for given values of N and p one needs to minimize

$$R(N, k, p) := \frac{k+1}{-N \log\left(1 - p_s(N, k, p)\right)}, \tag{22}$$

i.e. the number of queries per decrease of error probability by a factor of e, where the factor N^{-1} is included for normalization. Let the optimal k be denoted by $k_{\text{opt}}(N, p)$. Note that minimizing the rate function R from (22) originates from minimizing the number of oracle calls from independent rounds to get the failure probability below ε. This is different than optimizing the *expected* number of oracle calls, cf [12, 13] for a comparison.

Then our new algorithm consists of $m = \lceil (\log \varepsilon)/\log(1 - p_s(N, k_{\text{opt}}(N, p), p)) \rceil$ repetitions of rounds with $k_{\text{opt}}(N, p)$ Grover steps each, measurement in the standard basis plus verification step and storing the result in the classical output register when successful. The discussion in the following example will motivate an easy-to-compute quantity to be used instead of $k_{\text{opt}}(N, p)$.

Example 1. (Asymptotically optimal number of Grover iterations). If we are interested in large databases, we can simplify by taking $N \to \infty$, yielding, with (10) instead of (B5) (giving virtually identical results):

$$R(\infty, k, p) := \lim_{N \to \infty} R(N, k, p) = \frac{(k+1)}{(1-p)^k (2k+1)^2}. \tag{23}$$

To find the optimal $k = k_{\text{opt}}(\infty, p)$, we compare ($k \geqslant 1$):

$$\frac{R(\infty, k-1, p)}{R(\infty, k, p)} = \frac{k(2k+1)^2}{(k+1)(2k-1)^2}(1-p). \tag{24}$$

This is a decreasing function of p, so $R(\infty, k-1, p)$ and $R(\infty, k, p)$ intersect at

$$p_k(\infty) := \frac{4k^2 + 4k - 1}{k(2k+1)^2} = \frac{1}{k} + O\left(k^{-3}\right), \tag{25}$$

which is appropriate for large k, and we have $k(\infty, p) = k$ whenever $p_{k+1}(\infty) < p \leqslant p_k(\infty)$ with the convention $p_0(\infty) := 1$. By inverting the series expansion around $k = \infty$ one can get an *explicit* approximate expression for $k_{\text{opt}}(\infty, p)$: the inverse is $\frac{1}{p} + O(p)$, and thus $k_{\text{opt}}(\infty, p) \approx \lfloor \frac{1}{p} \rfloor$ is a good approximation especially for small p.

Intuitively, this behaviour of the optimal length k_{opt} of a quantum round can be understood by noting that the quantum register remains undisturbed with reasonably high probability (of order $O(1)$) for time $\sim 1/p$, whereas with a probability approaching 1 exponentially the quantum register will be disturbed by noise when $k \gtrsim (const)/p$. This is because the noise will hit at each step independently with probablity p (see equation (7)). Thus, roughly $\sim 1/p$ Grover steps provide an advantage (see also, e.g., [19]). Plugging $k = \lfloor \frac{1}{p} \rfloor = \frac{1}{p} + O(1)$ into $R(\infty, k, p)$ and considering small noise level $p \ll 1$, we see

$$R\left(\infty, \frac{1}{p} + O(1), p\right) = \frac{e}{4}p + \frac{e}{4}p^2 + O\left(p^3\right), \tag{26}$$

so $R(\infty, k_{\text{opt}}(\infty, p), p)$ vanishes linearly in p around $p = 0$. The fact that this is non-zero and finite for $p > 0$, means that the normalization by $1/N$ in (22) was the 'correct' one. This suggests that the number of steps $m(k+1)$ necessary is proportional to $Np \log(1/\varepsilon)$, i.e. linear in N for any fixed $p > 0$ and $\varepsilon \in (0, 1)$, meaning that the quadratic speedup is lost under depolarizing noise (7); see theorem 2 for a more rigorous and general lower bound on the runtime.

Example 1 motivates a more rigorous analysis of the case of finite $N < \infty$. In this finite case, one has to be careful for small values of p, since it is clearly not a good idea to do more than $\frac{\pi}{4}\sqrt{N}$ Grover iterations in one round. Let us first assume that $p > \frac{4}{\pi}\frac{1}{\sqrt{N}}$, and suppose we perform $m = \lceil cNp^2 \log \varepsilon^{-1} \rceil$ rounds with $k = \lfloor \frac{1}{p} \rfloor$ Grover steps in each round, for some $c > 0$ (this ansatz is motivated by example 1, and will below turn out to be good). Then the probability of failing in all rounds can be bounded as (see 10)

$$\begin{aligned}
\left[1 - p_s\left(N, \left\lfloor \frac{1}{p} \right\rfloor, p\right)\right]^{\left\lceil cNp^2 \log \frac{1}{\varepsilon}\right\rceil} &\leqslant \left[1 - (1-p)^{\left\lfloor \frac{1}{p} \right\rfloor} \sin^2\left(\left(2\left\lfloor\frac{1}{p}\right\rfloor+1\right)\arcsin\frac{1}{\sqrt{N}}\right)\right]^{\left\lceil cNp^2 \log \frac{1}{\varepsilon}\right\rceil} \\
&\leqslant \left[1 - (1-p)^{\frac{1}{p}} \sin^2\left(\left(\frac{2}{p}-1\right)\arcsin\frac{1}{\sqrt{N}}\right)\right]^{\left\lceil cNp^2 \log \frac{1}{\varepsilon}\right\rceil} \\
&\leqslant \left[1 - (1-p)^{\frac{1}{p}}(1-\delta)\left(\frac{2}{p}-1\right)^2\frac{1}{N}\right]^{\left\lceil cNp^2 \log \frac{1}{\varepsilon}\right\rceil} \\
&\leqslant \exp\left\{-c(1-p)^{\frac{1}{p}}(1-\delta)\left(\frac{2}{p}-1\right)^2 p^2 \log\frac{1}{\varepsilon}\right\} \\
&= \varepsilon^{c(1-\delta)(1-p)^{\frac{1}{p}}(2-p)^2}
\end{aligned} \tag{27}$$

for some $0 < \delta \leqslant 1 - 4/\pi^2$ depending on N and p.

As we want to guarantee a failing probability of at most ε, we need to choose c such that the exponent in the final expression (27) is at least 1—independently of p for the following statements to be valid. However, for large values of p the exponent goes to 0, which is a consequence of vanishing (10) for large p and $k = \lfloor \frac{1}{p} \rfloor \geqslant 1$ Grover steps; this can be avoided by introducing a cutoff $p^* \in (0, 1)$ into the specification of the algorithm, such that for $p \geqslant p^*$ we use $k = 0$ iterations in each round, i.e. only perform verification steps on randomly chosen elements. The failure probability in this range is

$$\left(1-\frac{1}{N}\right)^{\left\lceil cNp^2\log\frac{1}{\varepsilon}\right\rceil} \leqslant \left(1-\frac{1}{N}\right)^{cNp^{*2}\log\frac{1}{\varepsilon}} \leqslant \varepsilon^{cp^{*2}}. \tag{28}$$

Numerically one finds that viable values for c and p^* in the specification of a concrete algorithm, i.e. such that both (27) and (28) do not exceed ε, are, for example, $c = 5$ and $p^* = 1/2$ (even when setting $\delta = 1 - 4/\pi^2$). We have not optimized these constants, as our main interest for now is in the scaling for large N, small ε, and all noise levels p. The number of oracle invocations during such an algorithm is upper bounded by

$$\left(\left\lfloor\frac{1}{p}\right\rfloor+1\right)\left\lceil cNp^2\log\frac{1}{\varepsilon}\right\rceil \leqslant \frac{2}{p}\left(cNp^2\log\frac{1}{\varepsilon}+1\right) \leqslant \frac{\pi}{2}\sqrt{N} + 2cNp\log\frac{1}{\varepsilon}, \tag{29}$$

where we used $p > \frac{4}{\pi}\frac{1}{\sqrt{N}}$.

For small noise levels $p = \frac{\beta}{\sqrt{N}}$ (where $0 < \beta \leqslant 4/\pi$), one can do $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ Grover iterations in each of the m rounds, i.e. before each measurement, which gives a success probability of at least roughly $e^{-\beta\pi/4}$ by the lower bound (10); therefore, $m = \lceil e^{\beta\pi/4}\log\frac{1}{\varepsilon}\rceil \leqslant \lceil e\log\frac{1}{\varepsilon}\rceil$ rounds are sufficient to get the failure probability below ε, putting an upper bound on the number of oracle calls similar to (29) with a term proportional to $\sim\sqrt{N}\log(1/\varepsilon)$ instead of the term $\sim Np\log(1/\varepsilon)$.

Summing up, our algorithm finds the marked element x, except with probability $\varepsilon \in (0, 1]$, on a quantum computer with noise level $p \in [0, 1]$ using the oracle at most

$$c_1\sqrt{N} + c_2(Np + \sqrt{N}) \log \frac{1}{\varepsilon} \tag{30}$$

times, e.g. for (non-optimal) constants $c_1 = 2$, $c_2 = 10$. We omit here a formal statement of the algorithm, which should have become resonably clear from the description above, but will remedy this in section 2.4 for the more general case of an unknown noise level.

The algorithm just described performs a number m of quantum rounds, each of identical length k given by an ansatz that is based on example 1. A cleverer algorithm might try to detect when a noise event has happened and then immediately abort the present round in such a case and start a fresh round in order not to 'waste' oracle uses. One way to accomplish this would be to adjoin to the quantum register $\mathbb{C}^N$ used above another quantum register $(\mathbb{C}^2)^{\otimes r}$ of r qubits that is initialized to $|0\rangle^{\otimes r}$ at the beginning of each round and is left untouched by the above algorithm. In case of a partial depolarizing noise event, given by the term $pI_{2^rN}(\mathrm{Tr}\,\varrho)/(2^rN)$ in (7), the r-qubit register will then be reset to a computational basis state other than $|0\rangle^{\otimes r}$ which can be detected by a projective measurement on this auxilliary system with probability $1 - 2^{-r}$. Thus, by expending a small number r of extra qubits (e.g. a number r that is constant in the problem size N, or chosen as $r \sim \log(1/\varepsilon)$) one can detect a noise hit with high probability and abort the present round to gain a saving in the number of oracle calls compared to the algorithm outlined above.

While this is a viable strategy in the noise model used above, it is actually extremely dependent on the noise model (7). If, for example, the noise would replace the whole quantum state with probability p by $|0_N\rangle \otimes |0\rangle^{\otimes r}$ (instead of $I_{2^rN}/(2^rN)$), then the exact strategy would not work anymore. In particular, any such strategy relies strongly on the fact that the noise is correlated across the whole quantum register. While we allow such strongly correlated noise as a pessimistic assumption from the outset, in particular to prove our *lower* runtime bounds, one would probably not want the actual *constructive* algorithms to rely on this assumption. In contrast, our algorithm developed below example 1 as well as the upper runtime bound (30) work for any noise model $D_p^T(\varrho) = pT(\varrho) + (1-p)\varrho$ with T *any* quantum channel T (see appendix B). This is because we only use equation (10), which merely relies on the fact that with probability $1 - p$ the quantum register remains undisturbed. Furthermore, even when relying on an exactly known noise model as e.g. in equation (7), one would at most save a constant factor of order 1 by the error-detection-strategy compared to the runtime (30) of the algorithm outlined above. This is due to the exponential first factor in (10), which implies that only with small probability $\sim 1 - e^{-C}$ will the noise hit occur before executing C/p steps in one round (where $C < 1$ here, such that there would be a saving).

We would like to point out that there are at least two different interpretations of runtime complexity results like equation (30). Firstly, one can run the algorithm indefinitely (i.e.,without any a priori bound on the number of rounds) until the marked element is found. Then we can guarantee that the algorithm gives the correct result with probability 1, and the number of oracle calls required is at most $c_1\sqrt{N} + c_2(Np + \sqrt{N}) \log \frac{1}{\varepsilon}$ except with probability ε. Alternatively, one can decide in advance to use the oracle $c_1\sqrt{N} + c_2(Np + \sqrt{N}) \log \frac{1}{\varepsilon}$ times before terminating the above algorithm, and after any successful measurement store the result in a classical memory; then, in the end, the marked element will have been found with probability at least $1 - \varepsilon$.

With the runtime bound (30) at hand, one can look at the case where p is fixed and independent of N. Then we see that, unless $p = 0$, the leading term is $c_2 Np \log(1/\varepsilon)$, i.e.

proportional to N. On the other hand, if one supposes that p scales decreasingly with N, the other terms may dominate. In particular, if $p \lesssim 1/\sqrt{N}$, the leading term is $c_2\sqrt{N}\log(1/\varepsilon)$.

Next we show that our algorithm presented above is essentially optimal within a certain class of algorithms (a wider class of algorithms will be considered in section 3). Namely, we assume that the algorithms employ a quantum register $\mathbb{C}^N \otimes \mathbb{C}^M$ (as above in theorem 1), consist of several 'rounds' where in each round we prepare some state, apply arbitrary channels and use the oracle an arbitrary number of times (possibly different for each round, but applying the noise channel between any two consecutive queries), do any measurement yielding an element of $\{1, \ldots, N\}$, and verify the result with one oracle use, writing it into a (noiseless) classical register reserved for storing the output if correct. Crucially, we assume that the events of success in each round are independent of each other. This assumption is valid in particular if the noise is symmetric (under permutations of the basis vectors $|x\rangle$ of $\mathbb{C}^N$, which partial depolarizing from equation (7) satisfies) and if the steps between measurements are Grover iterations (as for example in our algorithm above).

Theorem 2. *(Lower runtime bound on memoryless algorithms). Consider a sequence of algorithms, one for each size of the search space $N = 1, 2, 3, \ldots$, satisfying the assumptions just stated and subject to partial depolarizing noise (equation (7)). If the success probabilities are $1 - \varepsilon_N$, then, asymptotically, the number of queries q_N in the Nth algorithm is lower bounded by the level $p \in [0, 1]$ of depolarizing noise:*

$$\liminf_{N\to\infty} \frac{q_N}{N\log\left(1/\varepsilon_N\right)} \geqslant \frac{p}{8}. \tag{31}$$

More precisely, for any $p, \varepsilon \in (0, 1]$ and any finite $N > 9/p^2$, the number of queries q_N satisfies:

$$q_N \geqslant \frac{Np\log(1/\varepsilon)}{8}\left[-\frac{Np^2}{9}\log\left(1-\frac{9}{Np^2}\right)\right]^{-1}. \tag{32}$$

Proof. Suppose that the Nth algorithm consists of m_N rounds with $k_N^1, \ldots, k_N^m$ queries in each round (abbreviating $m \equiv m_N$), with failure probability ε_N^i in the ith round. Then $q_N = (k_N^1 + 1) + \ldots + (k_N^m + 1)$, and by the independence condition

$$\begin{aligned}\frac{q_N}{N\log\left(1/\varepsilon_N\right)} &= \frac{\left(k_N^1+1\right)+\ldots+\left(k_N^m+1\right)}{N\log\left(1/\left(\varepsilon_N^1\ldots\varepsilon_N^m\right)\right)}\\ &= \frac{\dfrac{k_N^1+1}{N\log\left(1/\varepsilon_N^1\right)}\log\left(1/\varepsilon_N^1\right)+\ldots+\dfrac{k_N^m+1}{N\log\left(1/\varepsilon_N^m\right)}\log\left(1/\varepsilon_N^m\right)}{\log\left(1/\varepsilon_N^1\right)+\ldots+\log\left(1/\varepsilon_N^m\right)},\end{aligned} \tag{33}$$

which is a weighted average of expressions of type $\frac{k+1}{N\log(1/\varepsilon(N,k,p))}$. Lower-bounding this expression thus automatically lower-bounds (33), and therefore it is enough to consider the $m = 1$ case only.

Since, from (14), $1 - \varepsilon(N, k, p) \leqslant \frac{1}{N} + \frac{8}{Np^2} \to 0$ as $N \to \infty$ (for any $p > 0$), one has $\log(1/\varepsilon(N, k, p)) \leqslant \delta_N(1 - \varepsilon(N, k, p))$ for some positive sequence $\delta_N \to 1$. Using that, from (15), also $1 - \varepsilon(N, k, p) \leqslant 8(k+1)/(Np)$ we get

$$\liminf_{N\to\infty} \frac{k+1}{N \log(1/\varepsilon(N, k, p))} \geqslant \liminf_{N\to\infty} \frac{k+1}{N\delta_N \cdot 8(k+1)/(Np)} = \frac{p}{8}. \tag{34}$$

The finite-N bound follows from $1 - \varepsilon(N, k, p) \leqslant \frac{9}{Np^2}$ and setting δ_N equal to the quantity inside the square brackets in (32). □

Theorem 2 shows that, under any non-zero noise $p > 0$ (and $\varepsilon \in (0, 1)$), our algorithm from above has asymptotically optimal runtime, up to a constant factor: in our algorithm, $\varepsilon_N \equiv \varepsilon$ was chosen independent of N and (30) shows that asymptotically $q_N \lesssim 10Np \log(1/\varepsilon)$, which matches (31) up to a factor of 80. In the noiseless case $p = 0$, our algorithm reduces to repeated Grover searches, whose optimality for $p = 0$ was shown in [11–13].

One other implication is noteworthy: on the one hand, theorem 2 says that, at fixed positive noise $p > 0$ and asymptotically for $N \to \infty$, the number of oracle queries $\gtrsim Np \log(1/\varepsilon)$ *has* to grow at least linearly in N, so that the quadratic speed of noiseless Grover search is lost (at least for the class of algorithms considered above, and for the depolarizing noise model, equation (7). On the other hand, however, the prefactor in this linear growth is $O(p)$, which is actually achieved by the explicit algorithm above, see equation (30); thus, for small enough noise p, the number of oracle calls to solve the search task by a quantum algorithm is *much less* than the minimal number $\sim N(1 - \varepsilon)$ of oracle calls required by any classical algorithm, even in a noiseless environment.

In the following section, we will extend the above algorithm so that it works in a noisy environment even when its noise level p is *not* known beforehand (algorithm 1 and theorem 3).

2.4. Fault-ignorant search composed from basic building blocks

We are now ready to turn to the *'fault-ignorant' setting*—the algorithm should be ignorant of the actual noise level under which the quantum computer operates. More precisely, the goal is to find an algorithm for which not the ability to give the correct result depends on the level of noise, but rather only its runtime may depend on the actual noise level. Actually, the algorithm described in the previous section has this property for any fixed number k of oracle calls per round; however, the runtime can then become large unless $k \approx k_{\text{opt}}(N, p)$. For example, if we choose $k \approx \frac{\pi}{4}\sqrt{N}$ in order to get a quadratic speedup for $p = 0$, then for $p \approx 1$ the number of oracle calls grows as fast as $N^{3/2}$, which is clearly unsatisfactory.

In order to overcome this problem we allow the number of queries to change from round to round. Thus, for each N and ε, we need to find a sequence $k_1(N, \varepsilon)$, $k_2(N, \varepsilon)$, …, where $k_i(N, \varepsilon)$ denotes the number of Grover iterations performed in the ith round. Again, for our constructive algorithm, we employ the usual Grover iterations; and again, theorem 2 will later show that this algorithm is nearly optimal.

One idea can be as follows. In the first round, we do a Grover search with $k_1(N, \varepsilon) := k_{\text{opt}}(N, 0) \approx \frac{\pi}{4}\sqrt{N}$ oracle calls (for the definition of k_{opt} see below (22)). For $N \gtrsim \varepsilon^{-2}$ this is enough to get the error probability below ε as long as $p = 0$; the set $\left\{p \in [0, 1] \mid p_s(N, k_{\text{opt}}(N, 0), p) > 1 - \varepsilon\right\}$ is open and therefore

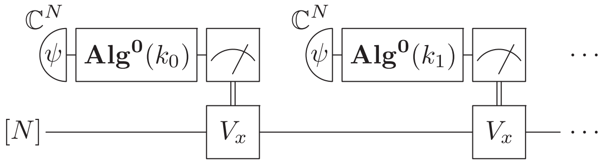

Figure 2. The fault-ignorant algorithm 1 searches the marked element x in consecutive rounds of k_0, k_1, k_2, … Grover steps plus one verification step each. Each round starts by preparing the equal superposition state ψ, which is follwed by Grover steps (a special case of algorithm 0, see figure 1), and a measurement in the computational basis; finally, the output is checked against the oracle (possibly by list lookup) and, in case of success, stored (by V_x) in a noiseless classical register with N states, so that the result is ready for readout by an external agent (the algorithm may however continue). No ancillary system $\mathbb{C}^M$ is used by the algorithm ($M = 1$).

$$p_2 := \inf \left\{ p \in [0, 1] \;\middle|\; 1 - p_s\left(N, k_1, p\right) \geqslant \varepsilon \right\} \tag{35}$$

exists and is larger than 0 (if the set is empty, e.g. when $\varepsilon \ll N^{-1/2}$, we set the infimum to $p_2 := 0$). Suppose that the measurement after the first round fails to find the marked element x. There are now two possibilities: either the actual noise level was below p_2, in which case the probability of this failure was at most ε (i.e., as required); or the actual noise level exceeded p_2, in which case the function k_{opt} gives an upper bound on the optimal number of Grover iterations to perform in the next round, so we set $k_2(N, \varepsilon) := k_{\text{opt}}(N, p_2)$. We then proceed similarly by iteratively setting $p_i := \inf \left\{ p \in [0, 1] \mid \prod_{j=1}^{i-1} \left(1 - p_s(N, k_j, p)\right) \geqslant \varepsilon \right\}$ and $k_i := k_{\text{opt}}(N, p_i)$, giving the number of Grover iterations to be performed in the ith round.

The sequences $\{k_i\}_i$ obtained this way are difficult to analyze, but by examining the results of numerical simulations for various values of N and ε one can get an idea about their behaviour. This turns out to be enough to get an approximation which still achieves the asymptotically optimal performance, up to a multiplicative factor in the runtime (see below). Specifically, we arrive at the following algorithm (see also figure 2) for *fault-ignorant quantum search*:

Algorithm 1. (Quantum search from basic building blocks). For suitably chosen $c > 0$, define

$$\alpha_g(\varepsilon) := \frac{1}{\sqrt{1 + \frac{g}{c \log(1/\varepsilon)}}}. \tag{36}$$

Repeat the following steps for $g = 0, 1, 2, \ldots$:

1. Prepare the equal superposition state $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$ on a quantum register $\mathbb{C}^N$,
2. Perform $k_g := \left\lfloor \alpha_g(\varepsilon) \frac{\pi}{4} \sqrt{N} \right\rfloor$ Grover iterations,
3. Measure in the standard basis, verify the result using one oracle invocation; if correct then store in classical output register.

The following theorem proves that algorithm 1 is fault-ignorant, i.e. finds the marked element independently of the actual noise level (in particular, the algorithm does not need p as an input), and gives a bound on its runtime which, however, depends on the actual noise level.

Theorem 3. *(Fault-ignorance of algorithm 1). Let $p \in [0, 1]$ be the actual noise level (i.e. noise $D_p(\varrho) := (1-p)\varrho + pT(\varrho)$ with any quantum channel T) acting on the quantum register when executing algorithm 1, and let $N \geqslant 100$ and $\varepsilon \in (0, 1/2]$. Then algorithm 1, with c = 10, finds the marked element after at most*

$$100\left(Np + \sqrt{N}\right)\log\frac{1}{\varepsilon} \tag{37}$$

oracle queries except with probability at most ε.

If one wants algorithm 1 to be fault-ignorant only for noise levels $p \in [0, 0.1]$, and if one presupposes reasonable values $N \geqslant 1000$ and $\varepsilon \in (0, 0.1]$, then one can guarantee the constant prefactor to be 20 instead of 100 as in (37), when using $c = 4.5$; cf appendix C.

As the proof of theorem 3 is rather technical, we present here only a sketch; for details, see appendix C.

Proof. *Sketch of the proof.* As the noise acts symmetrically with respect to the different oracles and since the Grover steps of algorithm 1 are symmetric as well, the success events for different rounds g are independent, so that we will be able to upper bound the failure probability after round g^*:

$$p_{\text{fail}} = \prod_{g=0}^{g^*}\left(1 - p_s\left(N, k_g, p\right)\right) \leqslant \exp\left\{-\sum_{g=0}^{g^*}(1-p)^{k_g}\sin^2\left(\left(2k_g + 1\right)\arcsin\frac{1}{\sqrt{N}}\right)\right\}. \tag{38}$$

When the sum $\sum_{g=0}^{g^*}$ in the last expression is greater than $\log(1/\varepsilon)$ then we can guarantee the failure probability to be at most ε, as desired. To get the statement about the number of oracle calls, we upper bound it by

$$\sum_{g=0}^{g^*}\left(k_g + 1\right) \leqslant g^* + 1 + \frac{\pi}{4}\sqrt{N} + \frac{\pi}{2}\sqrt{N}\left(c\log\frac{1}{\varepsilon}\right)\sqrt{1 + \frac{g^*}{c\log(1/\varepsilon)}}. \tag{39}$$

The proof of theorem 3 now consists in showing that there exists a number g^* (of rounds) such that the failure probability (38) is at most ε, while the number of oracle calls (39) is at most (37). Similar to our analysis leading up to (30), this argument will be split into three different cases: for $p \leqslant \pi/(4\sqrt{N})$ the first few rounds ($g = 0, 1, 2, \ldots$) are the important ones; for $p \geqslant 0.3$ we take into account only the rounds with $k_g = 0$ (i.e., large g); and for $\pi/(4\sqrt{N}) \leqslant p \leqslant 0.3$ our proof relies on an intermediate regime of g. Details are given in appendix C. □

Theorem 2 actually shows that the runtime of algorithm 1 (which we just proved to be at most equation (37)) is *optimal up to a constant*: to see this, note that algorithm 1 is contained in the class of algorithms to which the bound from theorem 2 on the number of oracle queries $q_N \gtrsim \frac{1}{8}Np\log\frac{1}{\varepsilon}$ applies. For any fixed noise level $p > 0$ and up to a constant factor, this equals the upper bound (37) on the number of queries needed by algorithm 1. In particular, even if one does *not* know the actual noise level in advance, one only loses a constant factor in the number of queries, compared to the runtime in case of known p (given in equation (30)).

We re-emphasize here the last point from section 2.3, that for small enough but constant noise levels p and in the limit $N \to \infty$, the quantum algorithm 1 needs *fewer* oracle calls than even the best classical algorithm in a noiseless environment.

As the runtime depends on an unknown parameter, it is necessary to have the ability to stop the algorithm as soon as the result is found. Theorem 3 then states a 'probabilistic bound' on the number of oracle uses up to the point when the element is found; this bound is probabilistic in the sense that in a fraction of at most ε of all runs of algorithm 1, the actual runtime may exceed this bound.

When considering more general algorithms, namely for which the events of failure in different rounds are not independent, the derivation of the lower bound on the necessary number of queries from theorem 2 is no longer valid. This dependency can arise either from asymmetric noise or from an asymmetry in the algorithm itself. Indeed, it is useful to consider such 'asymmetric algorithms': already classically one can find the marked element using $\lceil N(1 - \varepsilon) \rceil$ queries, except with error probability ε, by simply testing a subset of $\lceil N(1 - \varepsilon) \rceil$ elements using one oracle call each. This feature of not considering previously falsified items again is absent from algorithm 1 whose runtime may therefore exceed that of classical search, through the factor $\log(1/\varepsilon)$ in theorem 2.

The algorithms considered in section 3 will make use of this asymmetry, which can also be conceived of as conditioning the actions in future rounds on previous measurement outcomes that are being stored in a classical memory. This will be done by incorporating a noiseless classical memory which we will allow the algorithms to use in a limited way, namely by excluding oracle indices that have been falsified in previous rounds.

3. Search algorithms employing noiseless classical memory

3.1. Search with exclusion

Classical search algorithms can find the marked element with maximal failure probability ε using at most $\lceil N(1 - \varepsilon) \rceil$ steps, by excluding falsified oracle indices. Here we aim to achieve an upper bound of N on the runtime—independently of ε and of p—for our quantum algorithms as well, whereas in section 2 we have only presented algorithms whose runtime may exceed N parametrically due to the factor $\log(1/\varepsilon)$, e.g. in (37).

On a quantum computer a random choice may be implemented by preparing the equal superposition state $|\psi\rangle$ over a subset of basis states followed by a measurement in that basis. This in turn can be viewed as a Grover search with zero iterations (cf section 2.1). This leads to the idea of replacing the uniform random choices by proper Grover searches (each including several Grover steps plus a concluding measurement) over decreasing subsets, i.e. $\{1, \ldots, N\} \setminus \{i_1, \ldots, i_{m'}\}$ after round m'. For this, the classical noiseless memory that previously stored only the correct search outcome will be exanded by a register of 2^N states (N bits) to mark the previously excluded items.

We shall now develop and sketch a search algorithm based on this idea of excluding previously tested elements; the following procedure is applicable if the noise level p is known beforehand. If one fixes the number N of database entries, the noise parameter p and the target error bound ε, then the question is how to choose the number of iterations in each round in order to consume the least number of queries. Suppose that in the ith round we perform k_i queries and we do m rounds in total. Then the number of queries is $\sum_{i=1}^{m} (k_i + 1)$, while the probability of

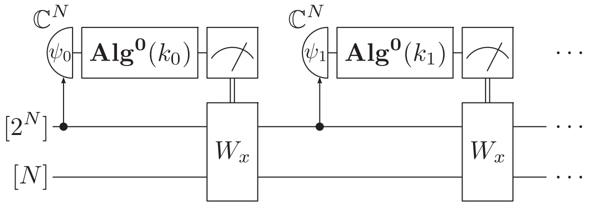

Figure 3. algorithm 2 uses exclusion in searching for an element in consecutive rounds of k_0, k_1, k_2, … Grover steps each, supplemented by one verification query. Each round starts by preparing the equal superposition state ψ_g of the previously not excluded elements, noted in the classical memory $[2^N]$, and is concluded by a measurement in the computational basis. The output is then verified against the oracle (list lookup) and stored in the classical noiseless memory $[N]$ if the element is found and marked in the memory $[2^N]$ if the round was unsuccessful (W_x).

not finding the marked element is at most $\prod_{i=1}^{m}(1 - p_s(N - i + 1, k_i, p))$; thus, the minimal number of queries for which one can guarantee success (up to failure probability ε) in the general noise model is

$$\min\left\{\sum_{i=1}^{m}\left(k_i + 1\right)\middle|\, m \in \mathbb{N},\ i_1, \ldots, i_m \in \mathbb{N},\ \prod_{i=1}^{m}\left(1 - p_s\left(N - i + 1, k_i, p\right)\right) \leqslant \varepsilon\right\}, \tag{40}$$

e.g. letting $p_s(N, k, p) \equiv (1 - p)^k \sin^2((2k + 1)\arcsin(1/\sqrt{N}))$ equal the lower bound in (10) (alternatively, (B5)). For given (N, p, ε), the minimum (40) and the corresponding sequence $\{k_i\}$ of Grover steps can be found by dynamic programming. Clearly, $\lceil N(1 - \varepsilon)\rceil$ is an upper bound on the number of oracle calls and for any fixed $p_0 > 0$ we can bound this as $\left\lceil N(1 - \varepsilon)\right\rceil \leqslant Np(1 - \varepsilon)/p_0$ for all $p \geqslant p_0$. Similarly, by (30) or theorem 3, for $\varepsilon \geqslant \varepsilon_0 > 0$ there is also an upper bound of the form $cNp(1 - \varepsilon)$, since $\log(1/\varepsilon) \leqslant c(1 - \varepsilon)$ where c is determined by ε_0. Hence, an upper bound on the runtime of the form $c'Np(1 - \varepsilon)$ holds for the complement of any neighbourhood of $(p, \varepsilon) = (0, 0) \in [0, 1]^2$, at least asymptotically for $N \to \infty$.

In the following section we simplify the above algorithm, based on typical behaviour of the sequences $\{k_i\}_i$ found in numerical experiments.

3.2. Fault-ignorant quantum search with exclusion

In this section we present a more explicit algorithm to solve the search problem in the *fault-ignorant setting*, i.e. an algorithm which can be specified and works even for *unknown* noise level p, using the exclusion described above to obtain faster runtime (cf also figure 3):

Algorithm 2. (Quantum search with exclusion). For suitably chosen $c > 0$, define $S_0 := \{1, \ldots, N\}$ and

$$\alpha_g(\varepsilon) := \frac{1}{\sqrt{1 + \frac{g}{c\log(1/\varepsilon)}}}. \tag{41}$$

Repeat the following steps for $g = 0, 1, 2, \ldots$:

1. Prepare the equal superposition state ψ_g over the set S_g,

2. Perform k_g Grover iterations (with $I - 2|\psi_g\rangle\langle\psi_g|$ as reflection), where

$$k_g := \begin{cases} \left\lfloor \alpha_g(\varepsilon) \frac{\pi}{4} \sqrt{N - g} \right\rfloor, & \text{if } k_0 + k_1 + \cdots + k_{g-1} + g \leqslant (1 - \varepsilon)N \,, \\ 0 \,, & \text{otherwise} \,, \end{cases} \tag{42}$$

3. Measure in the standard basis, verify the result r_g using one oracle invocation, store if correct,
4. Let $S_{g+1} := S_g \setminus \{r_g\}$.

Similarly as theorem 3 for algorithm 1, the following theorem proves fault-ignorance of algorithm 2 and provides a bound on its runtime:

Theorem 4. *(Fault-ignorance of algorithm 2). Let* $p \in [0, 1]$ *be the actual noise level (i.e. noise* $D_p(\varrho) := (1 - p)\varrho + pT(\varrho)$ *with any quantum channel T) acting on the quantum register when executing algorithm 2, and let* $N \geqslant 100$ *and* $\varepsilon \in (0, 1/2]$. *Then algorithm 2, with c = 10, finds the marked element after at most*

$$\min \left\{ 100(Np + \sqrt{N}) \log \frac{1}{\varepsilon}, \;\; 2(1 - \varepsilon)N + \sqrt{N} \right\} \tag{43}$$

oracle queries except with probability at most ε.

Proof. The success probability in each round of algorithm 2 is at least as large as in algorithm 1 because we are excluding elements; thus, the runtime of algorithm 1 puts an upper bound on the runtime by (37). Before that, however, algorithm 3 may switch to testing (and excluding) elements in random order (the second selector in (42)); this switch happens after at most $(1 - \varepsilon)N + (\pi/4)\sqrt{N} + 1$ oracle calls, and after the switch algorithm 2 needs at most $(1 - \varepsilon)N + 1$ additional calls to find the marked element except with probability ε. □

The constant 100 in (43) can be improved to 20 for a restricted range of parameters, following the remark below theorem 3.

Again, similar to theorem 2, we can show that, despite restricting it to Grover's specific steps, algorithm 2 is essentially optimal within a wider class of algorithms. Namely, we extend the class of algorithms considered in and before theorem 2 in such a way that, instead of requiring independence of the failure probabilities in different rounds, we assume that after each unsuccessful round we exclude the tested element and thereby reduce the search space as well as the state space of the computation. We also need to ensure that failure probabilities are multiplicative, which is the case e.g. if both the noise and the algorithm treat the search elements uniformly (this in particular applies to Grover iterations and partial depolarizing noise, see equation (7)).

This wider class of algorithms is qualitatively different from the algorithms considered in theorem 2, as it now contains algorithms succeeding with $(1 - \varepsilon)N \leqslant N$ oracle calls, *independently* of p, for example the classical verification-and-exclusion algorithm described at the beginning of section 3.1. This is reflected in the fact that the lower bound in the following theorem never exceeds N, unlike the bounds on q_N in the memoryless setting from theorem 2.

Theorem 5. *(Lower runtime bound on exclusion algorithms). For any quantum search algorithm (that may or may not have the noise level p as an input) satisfying the above* *constraints and whose quantum register is subject to depolarizing noise (see equation (7)) with fixed strength* $p \in (0, 1)$*, the number* q_N *of queries to find the marked element up to fixed failure probability* $\varepsilon \in (0, 1)$ *is lower bounded as*

$$q_N \geqslant \frac{N}{1 + \frac{8C_N}{p \log(1/\varepsilon)}}, \tag{44}$$

for some sequence $C_N = C_N(\varepsilon, p)$ *with* $\lim_{N\to\infty} C_N = 1$.

Proof. We can assume that $\lim\ \sup_{N\to\infty} q_N/N \leqslant 1 - \varepsilon$, because there does exist an algorithm with this limit being $1 - \varepsilon$ (see above). For now, fix N, and by letting $N \to \infty$ later we will make sure that all following expressions are well-defined (e.g. no logarithms of negative arguments occur, etc).

Let the number of queries in round g $(0 \leqslant g \leqslant r)$ be $k_g + 1$ (i.e., including the verification-exclusion step), so $r < q_N = \sum_{g=0}^{r} (k_g + 1)$. For the success probabilty in round g we have by (14) and (15)

$$p_s\left(N - g, k_g, p\right) \leqslant \frac{h\left(p, k_g\right)}{N - g} \quad \text{with} \quad h(p, k) \leqslant 1 + \frac{8}{p^2} \quad \text{and} \quad h\left(p, k_g\right) \leqslant \frac{8\left(k_g + 1\right)}{p}, \tag{45}$$

where the latter inequality implies that $h(p, k)$ is bounded independently of k (and of N), since p is given. Thus we can lower bound the failure probability (using $\log(1 - x) \geqslant -x/(1 - x)$):

$$\begin{aligned}
\varepsilon &\geqslant \textstyle\prod_{g=0}^{r} \left(1 - p_s\left(N - g, k_g, p\right)\right) \geqslant \exp\left\{\sum_{g=0}^{r} \log\left(1 - \frac{h\left(p, k_g\right)}{N - r}\right)\right\}, \\
&\geqslant \exp\left\{\sum_{g=0}^{r} - \frac{h\left(p, k_g\right)}{N - r} \frac{1}{1 - \frac{h\left(p, k_g\right)}{N - r}}\right\}, \\
&\geqslant \exp\left\{- \sum_{g=0}^{r} \frac{8\left(k_g + 1\right)}{(N - r)p} \left[1 - \frac{1}{N - r}\left(1 + \frac{8}{p^2}\right)\right]^{-1}\right\}, \\
&\geqslant \exp\left\{- \frac{8}{p} \frac{q_N}{N - q_N} C_N\right\},
\end{aligned} \tag{46}$$

where we used $g \leqslant r \leqslant q_N$ from above and defined $C_N := \left[1 - \frac{1}{N - q_N}\left(1 + \frac{8}{p^2}\right)\right]^{-1}$, which converges to 1 as $N \to \infty$. Inverting (46) to get an explicit bound on q_N finally gives (44). □

With the usual conventions in treating $1/0$ and $1/\infty$, theorem 5 is valid for all $p, \varepsilon \in [0, 1]$. Similar to equation (32) in theorem 2, one could explicitly specify a sequence C_N in the bound (44) which would however complicate the expression.

In summary, algorithm 2, which uses the exclusion strategy and Grover iterations, is *fault-ignorant* and theorem 4 provides an upper bound on its runtime. Conversely, theorem 5 provides a lower bound on the number of oracle calls for any symmetric fault-ignorant algorithm using the exclusion strategy. And the inequalities from lemma 2 (see appendix D) show that algorithm 2 is basically optimal within this class of algorithms, in the sense that for any $p, \varepsilon \in (0, 1)$ its runtime is at most a constant factor (independent of p and ε) above the lower bound from theorem 5.

And even stronger, the lower bound on the runtime in theorem 5 is proven for algorithms that may 'know' the noise level p as one of their inputs (such as the algorithm resulting from (40)), whereas our algorithm 2 basically saturates this lower bound *without* actually being dependent on the actual noise level p; the latter feature is the characteristic of *fault-ignorant algorithms*. Thus, not knowing the noise level inflicting upon the quantum computation extends the runtime at most by a constant factor, which was observed in the memoryless setting following theorem 3 as well.

3.3. Our search algorithms, and comparision to other work [19–21]

In section 2 we did not allow for a classical memory (except to store the correct output), whereas in section 3 we allowed a noiseless classical register in order to exclude falsified items from future search rounds. This is obviously not the most general class of algorithm. One may for example perform non-projective measurements which could result in a non-uniform distribution over oracles (cf [22]) after the measurement. Or one may abandon the division into 'rounds' altogether, and rather use the noisy quantum register and noiseless classical memory in a more general way (cf appendix A). While these possibilities are rather vague, at least in the noiseless case ($p = 0$) Grover's algorithm is exactly optimal [13]. A similar general proof eludes us in the noisy case considered in this paper; theorems 2 and 5 give such a bound under more restrictive qualifications.

Nevertheless, the results obtained here may suggest that any non-zero noise level $p > 0$ (in our noise models, cf appendix B) prolongs the runtime beyond the noiseless lower bound (in [13]), necessitating it to be proportional to the number of search items as $N \to \infty$. However, for small but constant noise level $p > 0$, the runtime bound $\lesssim Np \log(1/\varepsilon)$ on our algorithms (cf theorems 3 and 4) can be far below the $N(1 - \varepsilon)$ oracle calls required by the best noiseless classical algorithm. In this regard, see [23] for a treatment of *locally* acting noise and questions about optimality in this case.

Similar in spirit to our theorems 1, 2 and 5, a lower bound of $\gtrsim Np/(1 - p)$ on the runtime of general noisy quantum search algorithms was obtained in [20], whose faulty oracle model is somewhat similar to our partial depolarizing noise (7) (with roughly the same noise parameter p; they fixed $\varepsilon \simeq 1/10$); see also [21] for a continuous-time analogue of this result. One difference is that, in theorems 2 and 5, we allow error-free (e.g. classical) verification steps, whereas every oracle use in [20] is potentially faulty, leading to a diverging bound as $p \to 1$. Also, [20, 21] does not include a noiseless classical memory. Due to these extensions, our lower bounds are restricted to 'symmetric' algorithms consisting of 'rounds', whereas [20, 21] applies to *all* algorithms within their memoryless setting. These works do not consider achievability of the bound.

The work [19] specifically investigated Grover's algorithm under phase noise (see also [21]), again somewhat analogous to our noise model (7). It was observed that Grover's algorithm gives an advantage only if it runs for $k \lesssim 1/p$ steps, and it was hinted that at this time one may perform a measurement and start a new Grover round. In sections 2.2 and 2.3, we give more rigorous arguments (and prefactors) for the scaling $k \sim 1/p$, also for algorithms not necessarily consisting of Grover steps. Our algorithms 1 and 2 do indeed use the division into Grover rounds, but they even function *fault-ignorantly*. The avoidance of active error correction [6] is advocated by [19] as well.

A more technical difference of our work to most of the literature is that we consider the *average* success probability of an algorithm, i.e. averaged over all N oracles with equal weight, whereas the literature most often only investigates the *minimum* success probability of any of the N oracles. This makes our lower bounds stronger than the ones obtained in the literature. (As our constructive algorithms are symmetric, the minimum and average success probabilities coincide for those.)

4. Conclusion

In this paper we have investigated the idea of *fault-ignorant quantum algorithms*. Such algorithms should output the correct result even in the presence of noise of potentially unknown strength, in such a way that the actual noise level p may affect the runtime it takes to arrive at the correct answer (up to some specified failure probability ε), but should not affect that fact that the correct answer is found eventually. This approach allows us to reduce the required spatial circuit sizes, for example in comparison to using full-scale quantum error correction, however at the expense of increased runtime.

Following this general idea, we have provided fault-ignorant algorithms for quantum searching that function under depolarizing or dephasing noise of unknown strength p. We find the 'quadratic speedup' to be achievable only for low decoherence rates $p \lesssim 1/\sqrt{N}$. Otherwise, our best algorithm's runtime scales asymptotically like $\min\{Np \log 1/\varepsilon,\ N(1-\varepsilon)\}$ as $N \to \infty$. This is linear in N, but for low enough noise levels p it nevertheless outperforms the optimal classical search algorithm. Our algorithms may thus be useful for initial uses of quantum computing, when unlimited scalability of the size of quantum computers is not yet achievable due to technological limitations.

We moreover proved that, up to a constant factor, our algorithms' runtimes are optimal within wide classes of noisy quantum search algorithms. Remarkably, for the searching task, it turned out that ignorance of the actual noise level will extend the runtime by only a constant factor compared to the case of known noise level p.

Due to the novelty of the approach, our algorithms and lower bounds leave questions for further research. On the side of concrete algorithms, one may ask for them to be independent not only of the noise level p but also of the desired accuracy ε; then one could continue running the algorithm for longer to increase the success probability or accuracy.

Concerning lower bounds on the complexity of noisy quantum search, it would be worthwhile to establish an analogue of theorem 1 for the case of local noise models or even partial dephasing or general partial entanglement-breaking noise. The latter would immediately extend the validity of our lower bound in theorem 5 to the class of quantum algorithms that use a noiseless classical register in an arbitrary way and need not be divided into 'quantum rounds'. In a similar vein, it may also be possible to prove that the essentially optimal runtime for quantum searching under partial depolarizing noise, which we mainly investigated, can always be achieved by an algorithm divided into such rounds (see section 3.3). If not, it would be very interesting to find fault-ignorant algorithms that are not of this simple check-and-repeat form.

Finally, it would be desirable to investigate whether and how the fault-ignorant idea could possibly be applied to computational models other than the quantum circuit model. This would in particular be desirable in computational models for which quantum error correction techniques are less developed, such as adiabatic quantum computing, and where other methods to achieve fault tolerance are needed. Generally, we hope that, beyond unstructured search, the fault-ignorant idea will be fruitfully applied to algorithmic tasks, such as sampling algorithms.

Acknowledgments

We thank the two anonymous referees for their insightful comments which helped to improve the paper. This research was initiated at a workshop of the FP7 project COQUIT, which also supported P Vrana and D Reitzner. D Reeb was supported by the Marie Curie Intra-European Fellowship QUINTYL. M Wolf acknowledges support from the CHIST-ERA/BMBF project CQC and the Alfried Krupp von Bohlen und Halbach-Stiftung.

Appendix A. Fault-ignorance—a mathematical framework

The aim of this appendix is to provide a rigorous mathematical definition of fault-ignorant computing (see section 1.1 for a less formal discussion). We do this in a way which enables us to include a fairly broad class of algorithmic problem into this framework in a unified manner, while keeping the definition reasonably simple. The definitions are supposed formalize algorithms that do not need to know the actual noise level in order to accomplish their task—they should be *ignorant* of the noise. A fault-ignorant algorithm should be robust enough to provide the answer (up to some specified failure probability) under any level of noise, the latter affecting only its runtime.

In our formalization, we want to allow the desired and the actual output of the algorithm to be probabilistic (as is usual in sampling and quantum simulation problems), and to depend on an input (as for example in computational problems) and on an oracle (as in search problems). Given the discrete-time nature of the computation as well as of our noise models (cf appendix B), it is necessary to explicitly refer to an allowed class of quantum operations (the set S in the following definition). This can be done most conveniently if one also specifies the (spatial) resources available for performing the computation, i.e. the size of the quantum computer available or of any accompanying classical register. We thus view the specification of the size of the available quantum register as part of the task to be solved; and indeed, since early realizations of quantum computers will be limited in the number of qubits, it will be a part of the challenge to solve a desired task on the *available* hardware, esp. under noise influence because full-scale quantum error correction may be prohibited. Further, we consider only the quantum register to be noisy, whereas noiseless classical memory is today a reasonable technological assumption.

In light of this, we propose the following definitions, which we explain and supplement with examples afterwards.

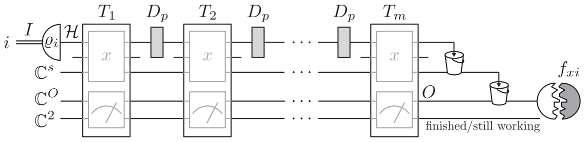

Figure A1. A fault-ignorant algorithm (definition 2), specified to solve a noisy quantum computational task (definition 1): an input state $\varrho_{\rm i}$ is affected in turn by devised operations T_j (which may include an oracle indexed by x, or other coherent operations, or measurement/verification procedures that store information in noiseless classical registers $\mathbb{C}^s$, $\mathbb{C}^O$, $\mathbb{C}^2$) and noise D_p acting on the quantum register. After some number of steps, the probability distribution of the output $o \in O$ should approximate the desired distribution f_{xi}, up to error ε.

Definition 1. (Noisy quantum computational task). A *noisy quantum computational task* is a tuple $(X, I, O, f, \mathcal{H}, \varrho, D, s, S)$ where

- X, I, O are sets,
- $f \in \mathbb{R}^{X \times I \times O}$ is a stochastic matrix, i.e. has non-negative entries and for any $x \in X$ and $i \in I$ we have $\sum_{o \in O} f_{xio} = 1$,
- $\mathcal{H}$ is a Hilbert space,
- ϱ: $I \to \mathcal{B}(\mathcal{H})$ is a function with density operators as values,
- $D_\cdot$: $[0, 1] \to \mathrm{CPT}(\mathcal{B}(\mathcal{H}))$ is a function with quantum channels on $\mathcal{B}(\mathcal{H})$ as values,
- $s \in \mathbb{N}$,
- $S \subseteq \mathrm{CPT}(\mathcal{B}(\mathcal{H}) \otimes \mathbb{C}^s \otimes \mathbb{C}^O \otimes \mathbb{C}^2)^X = \left\{T : X \to \mathrm{CPT}(\mathcal{B}(\mathcal{H}) \otimes \mathbb{C}^s \otimes \mathbb{C}^O \otimes \mathbb{C}^2)\right\}$.

We interpret X as the set labelling the different oracles, I and O as the sets of possible inputs and outputs, respectively (see also figure A1). For a task which does not make use of an oracle, we let X be any singleton set, and similarly, if the computation does not need an input, we let $|I| = 1$. The stochastic matrix f describes the desired distribution on the output set depending on the input and the oracle. The computation is performed using the Hilbert space $\mathcal{H}$ and a classical memory of s states, with the output being written into an additional classical register with $|O|$ states, corresponding to the possible output states in O. The additional classical bit $\mathbb{C}^2$ is to have value 1 iff the algorithm wants to signal that the result is available in the register $\mathbb{C}^O$. The reason for this is that in the fault-ignorant setting the runtime depends on an unknown parameter (namely p, see below), and therefore the algorithm needs a way to tell whether the computation is already done, without destroying the quantum state.

The map ϱ: $I \to \mathcal{B}(\mathcal{H})$ plays the role of input encoding in the sense that the physical initial state $\varrho_{\rm i}$ on the register $\mathcal{B}(\mathcal{H})$ represents the abstract input value $i \in I$. The quantum register $\mathcal{H}$ is subject to noise modeled by the quantum channels D_p (as specified in equation (A2) below) depending on a parameter $p \in [0, 1]$, which we think of as a strength parameter.

Finally, the set $S \subseteq \mathrm{CPT}(\mathcal{B}(\mathcal{H}) \otimes \mathbb{C}^s \otimes \mathbb{C}^O \otimes \mathbb{C}^2)^X$ represents the set of allowed elementary steps. An element in this subset is understood as a quantum channel depending on the oracle $x \in X$, whereby the quantum channel acts on the quantum register $\mathcal{B}(\mathcal{H})$ as well as on the classical (diagonal) registers $\mathbb{C}^s$, $\mathbb{C}^O$ and $\mathbb{C}^2$ described above (our specification below will be such that all these classical registers are initialized in the state $|0\rangle\langle 0|$ at the start of an algorithm). This gives a way to impose conditions on how 'complicated' the elementary operations are, e.g. in terms of oracle use or locality requirements (see examples below), and at the same time it maps the abstract oracle $x \in X$ to actual physical transformations $T(x)$ it may perform.

Definition 2. (Fault-ignorant algorithm). A *fault-ignorant algorithm* solving the noisy quantum computational task $(X, I, O, f, \mathcal{H}, \varrho, D, s, S)$ is a family $((T_n^{\varepsilon})_n)_{\varepsilon \in (0,1)}$ of finite or infinite sequences with $T_n^{\varepsilon} \in S$ such that for all $\varepsilon \in (0, 1)$ and for all $p \in [0, 1]$ the value

$$t^{\varepsilon}(p) := \min\left\{ t_0 \in \mathbb{N} \;\middle|\; \forall t \geqslant t_0\colon \forall x, i\colon \; \frac{1}{2} \left\| \hat{f}_{xi} - s_{xi}^{\varepsilon,t}(p) \right\|_1 \leqslant \varepsilon \right\} \tag{A1}$$

is finite, where $\hat{f} \in \mathbb{R}^{X \times I \times O \times \{0,1\}}$ is defined by $\hat{f}_{xio0} := 0$ and $\hat{f}_{xio1} := f_{xio}$ for $x \in X, i \in I, o \in O$, and

$$s_{xi}^{\varepsilon,\, t}(p) := \operatorname{Tr}_{\mathcal{B}(\mathcal{H}) \otimes \mathbb{C}^s} \left[T_t^{\varepsilon}(x) D_p T_{t-1}^{\varepsilon}(x) D_p \dots D_p T_2^{\varepsilon}(x) D_p T_1^{\varepsilon}(x) \left(\varrho_i \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0| \right) \right] \tag{A2}$$

is a probability distribution on $O \times \{0, 1\}$.

Thus, the sequence of operations in (A2) models a t-step noisy quantum computation, in the sense that between any two elementary operations from S a noise channel D_p is to be applied on the quantum register $\mathcal{B}(\mathcal{H})$ (figure A1). The sequence $(T_n^{\varepsilon})_n$ itself describes the algorithmic operations, which may depend on the required accuracy ε, i.e. on the maximally tolerable distance from the desired output distribution $\hat{f}_{xi}$, cf (A1).

The requirement for $t^{\varepsilon}(p)$ to be finite for *any* p, even though the algorithm $((T_n^{\varepsilon})_n)_{\varepsilon \in (0,1)}$ does *not* depend on p, justifies the term *fault-ignorant algorithm*. The condition '$\forall t \geqslant t_0$' in (A1) requires the result to be available in the classical memory at any later time when the outside agent, ignorant of the noise level p and thus of the necessary computation time $t^{\varepsilon}(p)$, may check the $\mathbb{C}^2$ flag to see whether the computation has already finished and wants to read out the result. Note that definition 2 does *not* put any requirements on the efficiency of the algorithm, which however in some circumstances may be quantified by $t^{\varepsilon}(p)$, i.e. the minimal number of invocations of $T_k^{\varepsilon}(x)$ (e.g. oracle calls); see section 1.1.

We now illustrate the definitions above by two examples.

Example 2. (Quantum search). As an example we now show how the noisy quantum search problem considered in sections 2 and 3 fits into this framework. In this case we have a set of N oracles $X = \{1, \dots, N\}$, and the algorithm is required to identify the oracle, so $O = X$. Since no input is needed, we set $I = \{0\}$. Now the function to be computed is deterministic, so f will be a 0-1 matrix, more specifically $f_{xio} = \delta_{xo}$. The Hilbert space we use is $\mathcal{H} = \mathbb{C}^N \otimes \mathbb{C}^M \otimes \mathbb{C}^2$ for some M setting the size of the ancillary quantum system and $\mathbb{C}^2$ standing for the ancillary system used by the oracle, cf equation (4). The noise acting on it is for example partial depolarizing, $D_p(\varrho) = p(\operatorname{Tr} \varrho)\frac{I_{2NM}}{2NM} + (1 - p)\varrho$. Since there is no input, ϱ_0 is just any fixed initial state, e.g. we may take $\varrho_0 = \frac{I_{2NM}}{2NM}$. In the version without classical memory we set $s = 1$ (section 2), while if we are to exclude previously tested elements, we may set $s = 2^N$ corresponding to an N-bit classical memory (section 3).

The set of allowed elementary operations to be applied between two noise hits is

$$S = \left\{ T \middle| \ \exists \ C_1, C_2 \in \mathrm{CPT}\left(\mathcal{B}(\mathcal{H}) \otimes \mathbb{C}^s \otimes \mathbb{C}^O \otimes \mathbb{C}^2\right) : \ \forall x \in X : \ T(x) \right. \\ \left. = C_2 \circ \widehat{O}_x \circ C_1 \right\}, \tag{A3}$$

where $\widehat{O}_x$ first prepares the pure state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ on the $\mathbb{C}^2$-subsystem of $\mathcal{H}$, and then acts as $|x', b\rangle \mapsto |x', b + \delta_{x,x'}\rangle$ on $\mathcal{H}$ (cf equations (4) and (5)) and as the identity on the classical registers. This choice of S means that an elementary step consists of a single use of the oracle, possibly applying an arbitrary (but oracle-independent) channel before and afterwards.

Finally, half of the trace-distance in (A1) gives, when the ready-flag $\mathbb{C}^2$ has been set to 1, exactly the probability of not outputting the correct oracle index in the classical output register, and it is this failure probability which we wanted to be smaller than ε in sections 2 and 3.

Example 3. (Binary addition). This example illustrates the possibility to introduce some kind of 'locality structure'. The task is the addition of two n-bit numbers given their binary representation using local gates on a $2n$-bit quantum register with local dephasing noise. Such a task is given by $X = \{0\}$, $I = \{0, 1, \ldots, 2^n - 1\} \times \{0, 1, \ldots, 2^n - 1\}$, $O = \{0, 1, \ldots, 2^{n+1} - 1\} \simeq \{0, 1\}^{n+1}$, $f_{x(i_1,i_2)o} = \delta_{i_1+i_2,o}$, the Hilbert space is $\mathcal{H} = \left(\mathbb{C}^2\right)^{\otimes 2n}$, $\varrho_{(i_1,i_2)} = |i_1, i_2\rangle$ (with i_1, i_2 considered as a $2n$-bit string), $D_p = d_p^{\otimes 2n}$ with d_p: $\mathcal{B}(\mathbb{C}^2) \to \mathcal{B}(\mathbb{C}^2)$ the partial dephasing with strength p, $s = 1$, and $S \subseteq \mathrm{CPT}(\mathcal{B}(\mathbb{C}^{2^{\otimes 2n}}) \otimes \mathbb{C}^O \otimes \mathbb{C}^2)$ consisting of 1- and 2-(qu)bit gates, i.e. channels which act as the identity on all but at most two bits (quantum or classical), remembering the subsystem structure of $\mathbb{C}^O \simeq \left(\mathbb{C}^2\right)^{\otimes n+1}$.

An algorithm that works only for *known* noise level p is *not* fault-ignorant; such algorithms may be formalized by assuming a p-dependence $(T_n^{\varepsilon,p})_n$ in the family of sequences in definition 2. On the other hand, if these sequences do *not* depend on the desired accuracy ε, i.e. $(T_n^\varepsilon)_n \equiv (T_n)_n$, then the algorithm does have another feature: the level of accuracy ε need not be specified before starting the algorithm; when higher accuracy is desired (i.e. smaller ε), one only needs to continue running the algorithm for longer.

Returning to efficiency questions, one may call a fault-ignorant algorithm (or rather, a family of fault-ignorant algorithms, parametrized by some 'problem size' N) *efficient* if, for any p, any ε and any N, its runtime is within a constant factor times the runtime of the best algorithm that may depend on ε *and* on p (see section 1.1). In this sense, our theorems 2 and 5 can be seen as statements that algorithms 1 and 2 are efficient (within restricted classes of algorithm).

It should be clear that there is nothing special about the set [0, 1] parametrizing the noise channels apart from the possibility to interpret it as 'strength' or to use it directly as a coefficient in a convex combination. One could instead consider a family $(D_p)_{p\in P}$ of noise channels indexed by an arbitrary set P parametrizing wider classes of noise, and so allowing for 'more' ignorance about the faults. Another obvious extension of definition 1 would be to allow for time-dependent noise.

Appendix B. Noise models

Here we elaborate on different kinds of noise which may be acting on the quantum computer during its runtime, and in particular on the noise models to which our results apply.

Partial depolarizing,

$$D_p^I(\varrho) = p\frac{I_d}{d}\operatorname{Tr}(\varrho) + (1-p)\varrho \tag{B1}$$

for noise level $p \in [0, 1]$, has been defined in equation (7), and corresponds to erasing the state of the quantum register with probability p (between any two oracle calls). Somewhat similar is *partial dephasing*,

$$D_p^{\varphi}(\varrho) := p\sum_{x=1}^{d}\langle x|\varrho|x\rangle|x\rangle\langle x| + (1-p)\varrho, \tag{B2}$$

acting on states ϱ on a d-dimensional Hilbert space equipped with a distinguished orthonormal basis $\{|x\rangle\}_{x=1}^{d}$ (for these, we imagine the basis states with respect to which the oracles (4) act). For $p = 1$, all quantum coherence is lost between any two oracle calls, but one can still perform a classical algorithm (on the basis states $|x\rangle$); in this sense, the noise level p of partial dephasing parametrizes how 'quantum' a search algorithm may be. Our constructive algorithms also work with the runtimes guaranteed by theorems 3 and 4 under the more general noise model

$$D_p^T(\varrho) := pT(\varrho) + (1-p)\varrho\ , \tag{B3}$$

where T may be any quantum channel, see discussion below equation (7).

Our formalization of noisy search algorithms (sections 2 and 3) does allow to *noiselessly check* whether a given index x' equals the marked element x, since immediately before and after an oracle call one may perform any quantum operation without noise (cf equation (13)) and thus one action of $\widetilde{O}_x$ from (4) on a suitably prepared quantum register can accomplish this check and write the result into the (noiseless) classical memory. This fact is important, as it allows the verification/falsification step at the conclusion of each round (cf algorithms 1 and 2). Alternatively, such a noiseless check may be implemented by a classical table lookup.

The above noise models are formulated in discrete time, but our prescription for the noise D_p to act between any two oracle calls is supposed to model the continuous action of noise in a real-world situation. For example, since in (B1) the probability to 'lose' the quantum computer between any two consecutive oracle calls is p, its lifetime is roughly $1/p$ (measured in the time between two oracle calls); and indeed, the time scale $k \sim 1/p$ appeared often in the analysis in section 2.3.

Note that quantum error correction [1, 4, 5] does not work for partial depolarizing (B1) or dephasing (B2), as these noises affect the whole quantum computer 'collectively'. This means that the whole quantum computer is subjected to a 'flash' of noise, such as drifting lasers or an external hit by a magnetic field. These may be reasonable noise models for not-too-large quantum computers.

Discussing the noise models more quantitatively, we first notice that the lower bound (10) on the success probability after k steps of Grover's algorithm under noise applies to all three noise models (B1)–(B3). For partial depolarizing (B1) one can compute the success probability in (10) exactly: the $2^k - 1$ omitted terms are of the form

$$\sum_{x=1}^{N} \frac{1}{N} p^m (1-p)^{k-m} \langle x | \frac{I_N}{N} | x \rangle = \frac{1}{N} p^m (1-p)^{k-m}, \tag{B4}$$

where m is the number of noise hits. These terms correspond to events when the maximally mixed state is prepared at some point due to noise acting and since both D_p and G_x (see before equation (9)) are unital. As the coefficients of these terms sum up to $(1-(1-p)^k)$, the exact success probability for this model is

$$p_s^{\text{pol}}(N, k, p) = \left(1 - (1-p)^k\right)\frac{1}{N} + (1-p)^k \sin^2\left((2k+1)\arcsin\frac{1}{\sqrt{N}}\right). \tag{B5}$$

Using this exact success probability for partial depolarizing improves the runtime bounds for this specific noise model (e.g. theorem 3 for large noise level p), but the lower bound (10) is quite tight unless $kp \gg 1$. The drawbacks of relying on a too specific noise model are furthermore discussed below equation (30).

An exact computation of the success probability can also be done for partial dephasing (B2), but is much more involved. Furthermore, one can prove that the success probability for partial dephasing is not smaller than for depolarizing at the *same* noise level: $p_s^{\varphi}(N, k, p) \geqslant p_s^{pol}(N, k, p)$. This inequality is, however, not immediate, as for identical noise parameters $p \in (0, 1)$, partial depolarizing D_p^I *cannot* be obtained by post-processing D_p^{φ}, i.e. $D_p^I \neq P \circ D_p^{\varphi}$ for all quantum channels P.

Our proofs of the general lower bounds on the number of oracle calls (theorems 2 and 5) require partial depolarizing (B1), as theorem 1 was proved only for generalized partial depolarizing noise $\varrho \mapsto p\tau \operatorname{Tr} \varrho + (1-p)\varrho$ and the proofs (and presuppositions) of theorems 2 and 5 require furthermore a symmetry between the oracle indices, limiting further to $\tau = I_d/d$.

Finally, we argue that it makes sense in sections 2 and 3 to perform efficiency analyses by keeping the noise parameter p fixed while the size of the quantum register N (or NM) varies, possibly even tending to infinity. Phrased another way, we ask whether, for example, the strength of partial depolarizing (B1) with parameter p on an d-dimensional quantum system is comparable to the strength of partial depolarizing with the same parameter p in d' dimensions, even if d and d' are widely different.

First, both partial depolarizing (B1) and partial dephasing (B2) (the latter with respect to a tensor product of bases) are compatible under tracing out subsystems when the *same* parameter p is used on the tensor product system and on the subsystem:

$$\operatorname{tr}_B\left[D_p^{I,\varphi}(\varrho_{AB})\right] = D_p^{I,\varphi}\left(\operatorname{tr}_B[\varrho_{AB}]\right) \qquad \forall p \in [0, 1]. \tag{B6}$$

With this parametrization of the noise, it does therefore not help for algorithm performance to introduce larger and larger ancillary systems or 'innocent bystanders': the noise on the 'Grover part' of the algorithm cannot be made small in this way, which is a reasonable requirement.

Secondly, both for partial depolarizing and dephasing, one can obtain the noise in $d - 1$ dimensions by post-processing the noise on a d-dimensional system:

$$D_p^{\varphi}(\varrho) \oplus 0 = P \circ D_p^{\varphi}(\varrho \oplus 0) \qquad \forall p \in [0, 1] \ \ \forall \varrho \in \mathcal{B}\left(\mathbb{C}^{d-1}\right), \tag{B7}$$

where $P = \text{id}$ for dephasing noise (and the additional dimension $|d\rangle$ has to correspond to one of the basis vectors in (B2)), and $P(X) := (I - |d\rangle\langle d|)X(I - |d\rangle\langle d|) + (I - |d\rangle\langle d|)\langle d|X|d\rangle/(d-1)$ for depolarizing. This compatibility under restrictions of the Hilbert space to subspaces is important and sensible in the context of exclusion algorithms (algorithm 2, and proof of theorem 4), where the effective dimension of the quantum register is reduced by 1 in each round.

Appendix C. Proof of theorem 3

Proof. As we assume the noise to act symmetrically with respect to the different oracles (which both partial depolarizing and dephasing do) and since the Grover steps of algorithm 1 are symmetric as well, the success events in different rounds g are independent. Thus, with (10), we can upper bound the failure probability after round g^* by

$$\begin{aligned} p_{\text{fail}} \equiv \textstyle\prod_{g=0}^{g^*}\left(1 - p_s\left(N, k_g, p\right)\right) &= \exp\left\{\sum_{g=0}^{g^*} \log\left(1 - p_s\left(N, k_g, p\right)\right)\right\} \\ &\leqslant \exp\left\{-\sum_{g=0}^{g^*} p_s\left(N, k_g, p\right)\right\} \\ &\leqslant \exp\left\{-\sum_{g=0}^{g^*}(1-p)^{k_g}\sin^2\left(\left(2k_g+1\right)\arcsin\frac{1}{\sqrt{N}}\right)\right\}. \end{aligned} \tag{C1}$$

To show that the failure probability is at most ε, as desired, below we will lower bound the sum $\sum_{g=0}^{g^*}$ and adjust parameters such that it is at least $\log(1/\varepsilon)$. The sum can be further bounded by assuming it to start at some $g = g_*$ with $0 \leqslant g_* \leqslant g^*$:

$$\sum_{g=0}^{g^*}(\ldots) \geqslant (1-p)^{k_{g_*}} \sum_{g=g_*}^{g^*} \sin^2\left(\left(2k_g+1\right)\arcsin\frac{1}{\sqrt{N}}\right). \tag{C2}$$

The number of oracle calls can be upper bounded as follows:

$$\begin{aligned} \sum_{g=0}^{g^*}\left(k_g+1\right) &\leqslant g^* + 1 + \frac{\pi}{4}\sqrt{N}\sum_{g=0}^{g^*}\left(1 + \frac{g}{c\log(1/\varepsilon)}\right)^{-1/2} \\ &\leqslant g^* + 1 + \frac{\pi}{4}\sqrt{N} + \frac{\pi}{4}\sqrt{N}\int_{g=0}^{g^*}\left(1 + \frac{g}{c\log(1/\varepsilon)}\right)^{-1/2} dg \\ &\leqslant g^* + 1 + \frac{\pi}{4}\sqrt{N} + \frac{\pi}{2}\sqrt{N}\left(c\log\frac{1}{\varepsilon}\right)\sqrt{1 + \frac{g^*}{c\log(1/\varepsilon)}}. \end{aligned} \tag{C3}$$

The proof of the theorem now consists in showing that there exists a number g^* (of rounds) such that the failure probability (C1) is at most ε, while the number of oracle calls (C3) does not exceed the value given in (37). This argument will be split into three cases, as sketched in the main text. We make abundant use of the fact that $c\log(1/\varepsilon) \geqslant 1$, since $\varepsilon \leqslant \varepsilon_0 := 1/2$ and

$c = 10$. We also define $N_0 := 100$ and assume $N \geqslant N_0$ throughout, in accord with the statement of theorem 3.

Case 1: $p \leqslant 4/(\pi\sqrt{N})$. In this case, the actual decoherence rate p is small, and we take only the first few rounds g into account to obtain an upper bound on p_{fail}. By using $\arcsin(1/\sqrt{N}) \geqslant 1/\sqrt{N}$, $\sin(x) \geqslant x \sin(x_0)/x_0$ for $0 \leqslant x \leqslant x_0 \leqslant \pi$, and setting

$$Q(N_0) := \frac{\sin^2\left[\left(\frac{\pi}{2}\sqrt{N_0} + 1\right)\arcsin\left(1/\sqrt{N_0}\right)\right]}{\left[\left(\frac{\pi}{2}\sqrt{N_0} + 1\right)\arcsin\left(1/\sqrt{N_0}\right)\right]^2},$$

we continue in bounding (C2):

$$\begin{aligned}\sum_{g=0}^{g^*}(\ldots) &\geqslant (1-p)^{k_{g_*}} Q(N_0) \sum_{g=g_*}^{g^*}\left(2\left\lfloor \alpha_g(\varepsilon)\frac{\pi}{4}\sqrt{N}\right\rfloor + 1\right)^2 \frac{1}{N},\\ &\geqslant \frac{\pi^2}{16}(1-p)^{k_{g_*}} Q(N_0) \sum_{g=g_*}^{g^*}\alpha_g^2(\varepsilon),\\ &\geqslant \frac{\pi^2}{16}(1-p)^{k_{g_*}} Q(N_0) \int_{g\,=\,g_*}^{g^*}\alpha_g^2(\varepsilon)dg\ ,\end{aligned} \tag{C4}$$

where we used $2\lfloor x\rfloor + 1 \geqslant x$ for $x \geqslant 0$. Choosing $g_* := 0$ and $g^* := \lceil c_0 \log(1/\varepsilon)\rceil$, we evaluate the integral in (C4) to find

$$\begin{aligned}\sum_{g=0}^{g^*}(\ldots) &\geqslant \frac{\pi^2}{16}(1-p)^{k_0} Q(N_0) c \log\left(1 + \frac{c_0}{c}\right)\log\frac{1}{\varepsilon},\\ &\geqslant \frac{\pi^2}{16}\left(1 - \frac{4}{\pi}\frac{1}{\sqrt{N_0}}\right)^{\frac{\pi}{4}\sqrt{N_0}} Q(N_0) c \log\left(1 + \frac{c_0}{c}\right)\log\frac{1}{\varepsilon},\end{aligned} \tag{C5}$$

where we used $p \leqslant \pi/(4\sqrt{N})$. When the prefactor of $\log(1/\varepsilon)$ is at least 1, then the failure probability p_{fail} will be at most ε by equation (C1); this happens e.g. for the choice $c_0 := 170$. The number of oracle calls from (C3) is then

$$\leqslant 2 + \frac{\pi}{4}\sqrt{N} + \left(\frac{\pi}{2}\sqrt{2c^2 + c_0 c} + \frac{c_0}{\sqrt{N_0}}\right)\sqrt{N}\log\frac{1}{\varepsilon}, \tag{C6}$$

which is less than $\sqrt{N} + 86\sqrt{N}\log(1/\varepsilon)$ due to $2 + \pi\sqrt{N}/4 \leqslant \sqrt{N}$. We notice that the term linear in N (cf equation (37)) is absent from the runtime (C6) in case 1; intuitively speaking, such small noise levels p still allow for quadratic speedup in the quantum search.

Case 2: $4\big/(\pi\sqrt{N}) \leqslant p \leqslant p^*$, where we define $p^* := 0.3$. In this intermediate region of the actual decoherence rate (the need for $p^* < 1$ will become evident later), we define $g^* := \left\lceil c_2(\pi^2/16)Np^2 \log(1/\varepsilon)\right\rceil$ and $g_* := \left\lceil c_1(\pi^2/16)Np^2 \log(1/\varepsilon)\right\rceil$ with $c_2 > c_1 > c$ to be determined later. Our choice $c_1 > c$ will in particular imply $g_* \geqslant c \log(1/\varepsilon)$, so that we can continue lower-bounding (C4) by bounding the integrand,

$$\begin{aligned}\sum_{g=0}^{g^*}(\dots) &\geqslant \frac{\pi^2}{32}(1-p)^{k_{g_*}}Q\left(N_0\right)\left(c\log\frac{1}{\varepsilon}\right)\int_{g=g_*}^{g^*}\frac{dg}{g} \geqslant \frac{\pi^2}{32}(1-p)^{k_{g_*}}Q\left(N_0\right)c\log\frac{g^*}{g_*}\log\frac{1}{\varepsilon}\\ &\geqslant \frac{\pi^2}{32}\exp\left\{\frac{\pi}{4}\sqrt{N}\left(1+\frac{g_*}{c\log(1/\varepsilon)}\right)^{-1/2}\log(1-p)\right\}\\ &\quad\times Q\left(N_0\right)c\log\frac{c_2}{c_1+\left(\frac{\pi^2}{16}Np^2\log(1/\varepsilon)\right)^{-1}}\log\frac{1}{\varepsilon}\\ &\geqslant \frac{\pi^2}{32}\exp\left\{\frac{\log\left(1-p^*\right)}{p^*}\sqrt{\frac{c}{c_1}}\right\}Q\left(N_0\right)c\log\frac{c_2}{c_1+\log^{-1}\left(1/\varepsilon_0\right)}\log\frac{1}{\varepsilon},\end{aligned}\tag{C7}$$

where we used $p\log(1-p^*)\leqslant p^*\log(1-p)$ (due to $p\geqslant p^*$) and $(\pi^2/16)Np^2\log(1/\varepsilon)\geqslant\log(1/\varepsilon_0)$ (due to $p\geqslant 4/(\pi\sqrt{N})$). Again, the prefactor of $\log(1/\varepsilon)$ can be made larger than 1 by choosing $c_1 := 20$ and $c_2 := 180$. The number of oracle calls from (C3) is then

$$\leqslant 2+\frac{\pi}{4}\sqrt{N}+\frac{\pi}{\sqrt{2}}\sqrt{N}c\log\frac{1}{\varepsilon}+\frac{\pi^2}{16}\left(p_2^*+2\sqrt{cc_2}\right)Np\log\frac{1}{\varepsilon},\tag{C8}$$

which again is less than $\sqrt{N}+86(Np+\sqrt{N})\log(1/\varepsilon)$.

Case 3: $p\geqslant p^*$. For large actual noise levels p, it is enough to consider only those rounds g for which $k_g(\varepsilon)=0$; in each such round, a measurement is performed on the equal superposition state, leading to a success probability of exactly $1/N$. This leads to the choice $g_* := \left\lfloor c(\pi^2/16)N\log(1/\varepsilon)\right\rfloor$ and $g^* := \left\lfloor c_3(\pi^2/16)N\log(1/\varepsilon)\right\rfloor$, and we can lower-bound (C2):

$$\sum_{g=0}^{g^*}(\dots)\geqslant\sum_{g=g_*}^{g^*}\frac{1}{N}=\frac{1}{N}\left(g^*-g_*+1\right)\geqslant\frac{\pi^2}{16}(c_3-c)\log\frac{1}{\varepsilon}.\tag{C9}$$

By choosing $c_3 := 12$, the prefactor of $\log(1/\varepsilon)$ exceeds 1. The number of oracle calls from (C3) is then

$$\leqslant 1+\frac{\pi}{4}\sqrt{N}+\frac{\pi}{2}c\sqrt{N}\log\frac{1}{\varepsilon}+\frac{\pi^2}{16p^*}\left(c_3+2\sqrt{cc_3}\right)Np\log\frac{1}{\varepsilon},\tag{C10}$$

which is again less than $\sqrt{N}+86(Np+\sqrt{N})\log(1/\varepsilon)$.

So far we have proved that the algorithm is fault-tolerant with runtime at most $\sqrt{N}+86(Np+\sqrt{N})\log(1/\varepsilon)$. Due to $\varepsilon\leqslant\varepsilon_0=1/2$, (37) is an upper bound on the runtime. □

For the proof of the prefactor 20 mentioned below theorem 3, the Case 3 in the proof above can be neglected, and we set $p^* := \varepsilon_0 := 0.1$, and alter the lower bound (C4) a bit, such that constants c_0, c_1, c_2 etc can be found to yield the lower guaranteed runtime.

Appendix D. Technical Lemmata

By the following lemma, we convert Zalka's implicit bound, equation (11), into an explicit one (see after the proof below):

Lemma 1. *Let* $0 \leqslant x \leqslant 1$ *and* $0 \leqslant y \leqslant 1$*. Then, for any* $0 < \alpha < \infty$*:*

$$\sqrt{xy} + \sqrt{(1-x)(1-y)} \leqslant 1 + \frac{1}{2}(\alpha - 1)x + \frac{1}{2}\left(\frac{1}{\alpha} - 1\right)y \,. \tag{D1}$$

Proof. The left-hand side is a concave function of $(x, y) \in [0, 1]^2$, smooth in the interior, and hence its graph stays under its tangent plane drawn at any point (x_0, y_0) for $0 < x_0 < 1$ and $0 < y_0 < 1$. The partial derivatives of $h(x, y) = \sqrt{xy} + \sqrt{(1-x)(1-y)}$ at (x, y) are

$$h_x(x, y) = \frac{\sqrt{y}}{2\sqrt{x}} - \frac{\sqrt{1-y}}{2\sqrt{1-x}} \quad \text{and} \quad h_y(x, y) = \frac{\sqrt{x}}{2\sqrt{y}} - \frac{\sqrt{1-x}}{2\sqrt{1-y}} \,. \tag{D2}$$

Writing $y_0 = \alpha^2 x_0$ for $0 < \alpha < \infty$, we have

$$\begin{aligned} h(x, y) &\leqslant h(x_0, y_0) + h_x(x_0, y_0)(x - x_0) + h_y(x_0, y_0)(y - y_0) \\ &= h\left(x_0, \alpha^2 x_0\right) + \frac{1}{2}\left(\alpha - \sqrt{\frac{1-\alpha^2 x_0}{1-x_0}}\right)(x - x_0) \\ &\quad + \frac{1}{2}\left(\frac{1}{\alpha} - \sqrt{\frac{1-x_0}{1-\alpha^2 x_0}}\right)\left(y - \alpha^2 x_0\right) . \end{aligned} \tag{D3}$$

Now taking the limit $x_0 \to 0$ yields (D1). □

We apply this lemma to Zalka's bound [13] (equation (11) above) with $x = p_s$ and $y = \frac{1}{N}$:

$$\begin{aligned} 4k^2 &\geqslant 2N - 2\sqrt{N}\sqrt{p_s} - 2\sqrt{N}\sqrt{N-1}\sqrt{1-p_s} \\ &= 2N\left[1 - \left(\sqrt{p_s \frac{1}{N}} + \sqrt{\left(1-p_s\right)\left(1 - \frac{1}{N}\right)}\right)\right] \\ &\geqslant 2N\left[1 - \left(1 + \frac{1}{2}(\alpha - 1)p_s + \frac{1}{2}\left(\frac{1}{\alpha} - 1\right)\frac{1}{N}\right)\right] \\ &= (1 - \alpha)Np_s - \left(\frac{1}{\alpha} - 1\right). \end{aligned} \tag{D4}$$

One can easily see that the sharpest bound on p_s is obtained for $\alpha = (2k+1)^{-1}$, yielding $Np_s \leqslant (2k+1)^2$.

The following lemma shows that algorithm 2 in section 3 is optimal within the class of algorithms considered in theorem 5, up to a constant factor in the runtime:

Lemma 2. *For* $0 < p < 1$ *and* $0 < \varepsilon < 1$ *the following inequalities hold:*

$$\frac{1}{1+\frac{1}{p\log(1/\varepsilon)}} \leqslant \min\left\{1-\varepsilon,\ \ p\log\frac{1}{\varepsilon}\right\} \leqslant \frac{2}{1+\frac{1}{p\log(1/\varepsilon)}}. \tag{D5}$$

Proof. First, $p\log(1/\varepsilon) > 0$ implies

$$\frac{1}{1+\frac{1}{p\log(1/\varepsilon)}} \leqslant p\log\frac{1}{\varepsilon}. \tag{D6}$$

Now let $h(x) = x\log\frac{1}{x}$ for $x > 0$. Then $h'(x) = -1 - \log x$ and $h''(x) = -1/x$, so h is concave, and the tangent at $x = 1$ is $1 - x$. This gives $\varepsilon p\log(1/\varepsilon) \leqslant \varepsilon\log(1/\varepsilon) \leqslant 1-\varepsilon$, which implies

$$\frac{1}{1+\frac{1}{p\log(1/\varepsilon)}} \leqslant 1-\varepsilon, \tag{D7}$$

concluding the left inequality. For the right inequality, if $\min\left\{1-\varepsilon,\ p\log\frac{1}{\varepsilon}\right\} = 1-\varepsilon \leqslant 1$, then

$$\frac{2}{1+\frac{1}{p\log(1/\varepsilon)}} \geqslant \frac{2}{1+\frac{1}{1-\epsilon}} \geqslant 1-\epsilon. \tag{D8}$$

Lastly, if $\min\left\{1-\varepsilon,\ p\log\frac{1}{\varepsilon}\right\} = p\log\frac{1}{\varepsilon}$ then in particular $0 < p\log\frac{1}{\varepsilon} \leqslant 1$. Thus, finally,

$$p\log\frac{1}{\varepsilon} \leqslant \frac{2}{1+\frac{1}{p\log(1/\varepsilon)}}. \tag{D9}$$

□

References

[1] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)

[2] Shor P 1995 Scheme for reducing decoherence in quantum memory *Phys. Rev.* A **52** 2493

[3] Steane A M 1996 Error correcting codes in quantum theory *Phys. Rev. Lett.* **77** 793

[4] Shor P 1996 *Fault-tolerant Quantum Computation Proc. of the XXXVII Annual Symp. on Fundamentals of Computer Science* (Los Alamitos, CA: IEEE Press)

[5] Knill E, Laflamme R and Zurek W H 1998 Resilient quantum computation: error models and thresholds *Proc. R. Soc.* A **454** 365

[6] Preskill J 1998 Reliable quantum computers *Proc. R. Soc.* A **454** 385

[7] Gottesman D 2013 Fault-tolerant quantum computation with constant overhead (arXiv:1310.2984 [quant-ph])

[8] Zanardi P 1997 Dissipative dynamics in a quantum register *Phys. Rev.* A **56** 4445

[9] Lidar D A, Chuang I L and Whaley K B 1998 Decoherence-free subspaces for quantum computation *Phys. Rev. Lett.* **81** 2594

[10] Grover L K 1997 Quantum mechanics helps in searching for a needle in a haystack *Phys. Rev. Lett.* **79** 325

[11] Bennett C H, Bernstein E, Brassard G and Vazirani U 1997 Strengths and weaknesses of quantum computing *SIAM J. Comput.* **26** 1510

[12] Boyer M, Brassard G, Hoyer P and Tapp A 1998 Tight bounds on quantum searching *Fortschr. Phys.* **46** 493

[13] Zalka C 1999 Grover's quantum searching algorithm is optimal *Phys. Rev.* A **60** 2746

[14] Pablo-Norman B and Ruiz-Altaba M 1999 Noise in Grover's quantum search algorithm *Phys. Rev.* A **61** 012301

[15] Long G L, Li Y S, Zhang W L and Tu C C 2000 Dominant gate imperfection in Grover's quantum search algorithm *Phys. Rev.* A **61** 042305

[16] Azuma H 2002 Decoherence in Grover's quantum algorithm: perturbative approach *Phys. Rev.* A **65** 042311

[17] Shapira D, Mozes S and Biham O 2003 Effect of unitary noise on Grover's quantum search algorithm *Phys. Rev.* A **67** 042301

[18] Hsieh J, Li C and Chuu D 2004 An improved phase error tolerance in quantum search algorithm *Chin. J. Phys.* **42** 585

[19] Shenvi N, Brown K R and Whaley K B 2003 Effects of a random noisy oracle on search algorithm complexity *Phys. Rev.* A **68** 052313

[20] Regev O and Schiff L 2008 Impossibility of a quantum speed-up with a faulty oracle *Proc. of the XXXV Int. colloquium on Automata, Languages and Programming* **1** 773

[21] Temme K 2014 A note on the runtime of a faulty Hamiltonian oracle (arXiv:1404.1977 [quant-ph])

[22] Montanaro A 2010 *Quantum Search with Advice Proc. of the 5th Conf. on Theory of Quantum Computation, Communication and Cryptography* (Berlin: Springer) p 77

[23] Ben-Or M, Gottesman D and Hassidim A 2013 Quantum refrigerator (arXiv:1301.1995 [quant-ph])