Quantum search with advice

Ashley Montanaro[*]

Department of Computer Science, University of Bristol,
Woodland Road, Bristol, BS8 1UB, UK.

September 14, 2009

Abstract

We consider the problem of search of an unstructured list for a marked element x, when one is given advice as to where x might be located, in the form of a probability distribution. The goal is to minimise the expected number of queries to the list made to find x, with respect to this distribution. We present a quantum algorithm which solves this problem using an optimal number of queries, up to a constant factor. For some distributions on the input, such as certain power law distributions, the algorithm can achieve exponential speed-ups over the best possible classical algorithm. We also give an efficient quantum algorithm for a variant of this task where the distribution is not known in advance, but must be queried at an additional cost. The algorithms are based on the use of Grover's quantum search algorithm and amplitude amplification as subroutines.

1 Introduction

Grover's algorithm for search of an unstructured list is one of the greatest successes of the nascent field of quantum computation [6]. The algorithm operates in the *black box* model: given access to a function $f : \{1, \dots, n\} \to \{0, 1\}$, where f is promised to take the value 1 on precisely one input x, it finds x with certainty using $O(\sqrt{n})$ queries to f, whereas any classical algorithm requires $\Omega(n)$ queries to perform the same task. However, it is rarely necessary to search the type of databases that we encounter in real life in a completely unstructured fashion. Instead, there is often some prior information about the location of the sought ("marked") item x, which can be used to guide the search. We can formalise this intuition by considering a search problem where the searcher is given access to a probability distribution μ, which hints where the marked item is likely to be. We term this problem Search with Advice, and give a definition below.

Problem: Search with Advice
Input: A function $f : \{1, \dots, n\} \to \{0, 1\}$ that takes the value 1 on precisely one input x, and an "advice" probability distribution $\mu = (p_y)$, $y \in \{1, \dots, n\}$, where p_y is the probability that $f(y) = 1$.
Output: The marked element x.

It is clear that knowledge of μ can enable a classical algorithm to achieve a significant reduction in the average number of queries to f (with respect to μ) required to find the marked element x.

[*]montanar@cs.bris.ac.uk

This paper is concerned with the development of *quantum* algorithms for the SEARCH WITH ADVICE problem which also use μ, and which obtain significant speed-ups over any classical algorithm.

We distinguish two models for the complexity of this problem. In the first model – the *known* model – μ is known completely beforehand, and can be used to help design an algorithm to find the marked element. The complexity of the problem is given by the minimum expected number of queries to f required to find x, under the distribution μ. In the second model – the *unknown* model – μ is not known before the algorithm starts, but the algorithm is also given access to a black box which outputs samples from μ, at unit cost. In the case of quantum algorithms, the black box outputs a coherent superposition corresponding to μ (a "quantum sample").

In both cases, note that we are interested in the *average* number of queries with respect to μ required to find the marked element, rather than the worst-case number of queries. Previous work has shown that, if one considers the worst-case number of queries to the input required to compute any total function, there can only be at most a polynomial separation between quantum and classical computation [2]. Considering the average number of queries required (over the input) allows one to sidestep these results and hope to obtain *exponential* speed-ups.

Indeed, previous work of Ambainis and de Wolf [1] has shown that quantum algorithms *can* achieve exponential (or even super-exponential) reductions in average-case query complexity over classical algorithms. The model that these authors considered was that of computing a particular boolean function $f : \{0,1\}^n \to \{0,1\}$, with a particular (known) distribution on the inputs. Among other results, they exhibited a (function, distribution) pair with a super-exponential separation between quantum and classical query complexity, and even gave a function whose quantum and classical query complexity were exponentially separated under the *uniform* distribution.

1.1 New results

The main results of this paper are as follows. First, in the known model, we give a quantum algorithm for SEARCH WITH ADVICE which is optimal up to constant factors. Assuming without loss of generality that the probability distribution $\mu = (p_x)$ is given in non-increasing order, the algorithm uses an expected number of queries to f which is of the order of

$$\sum_{x=1}^{n} p_x \sqrt{x},$$

which should be compared with the optimal classical expected number of queries,

$$\sum_{x=1}^{n} p_x \, x.$$

For certain probability distributions, this represents an exponential (or even super-exponential) improvement in the expected number of queries used. The quantum algorithm is based on the use of an exact variant of Grover's algorithm [6, 8, 4] as a subroutine. Known lower bounds on the query complexity of quantum search are used to show that this algorithm is optimal, up to constant factors, for any probability distribution μ.

In the unknown model, we give a quantum algorithm that uses a expected number of queries of the order of

$$\left(\sum_{x, p_x > 1/n} \sqrt{p_x} \right) + \sqrt{n} \left(\sum_{x, p_x \leq 1/n} p_x \right).$$

Again, this algorithm is sometimes significantly more efficient than the best possible classical algorithm. The algorithm is based on the amplitude amplification algorithms proposed by Boyer et al [3]

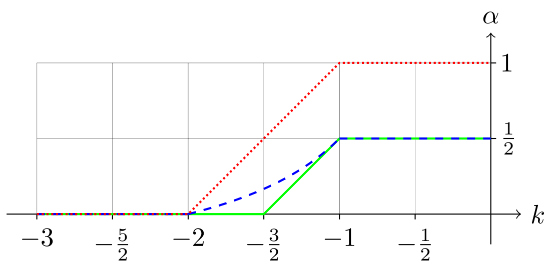

Figure 1: Query complexity of Search with Advice in different models, for power law distributions $p_x \propto x^{-k}$. For each k, the query complexity of the algorithms given in this paper is $\Theta(n^\alpha)$ for some α (ignoring log factors); the graph plots the exponent α against k. Dotted red line: best classical algorithm; solid green line: quantum, known probability distribution; dashed blue line: quantum, unknown probability distribution.

and Brassard et al [4]; the main difference being that after performing a certain number of iterations of amplitude amplification, it reverts to exact Grover search. This can considerably improve the average query complexity.

These results in the two different models are applied to the natural class of power law distributions $p_x \propto x^{-k}$, for some constant $k > 0$. We will see that for certain values of k, quantum algorithms deliver very significant reductions in the average number of queries used. In particular, when $-2 < k < -3/2$, a super-exponential separation between quantum and classical computation is obtained in the known model ($O(1)$ vs. $\Omega(n^{k+2})$). The results for power law distributions are summarised in Figure 1.1 (with details given in Propositions 2.5 and 3.4 below).

The paper is organised as follows. After defining the models and notation used, Section 2 contains the results on the known model, while Section 3 studies the unknown model. Various proofs are deferred to an appendix.

1.2 Models and notation

In this section, we set up concepts and notation that will be used throughout the paper. We assume familiarity with quantum computation [10], and in particular the concept of query complexity [5]: the number of queries to the input which a classical or quantum algorithm requires to compute some function. Let $[n]$ denote the integers $\{1, \dots, n\}$, and consider an oracle function $f : [n] \to \{0, 1\}$ which is promised to take the value 1 on precisely one input $x \in [n]$ (that is, $f(y) = \delta_{xy}$). We say that x is the *marked* element. Also consider a quantum or classical algorithm $\mathcal{A}$ which, given access to f, attempts to output x. We say that $\mathcal{A}$ is a *valid* algorithm if, for any f satisfying the above constraint, $\mathcal{A}$ outputs x with certainty. Let $\mathcal{D}$ denote the set of valid deterministic classical algorithms, and let $\mathcal{Q}$ denote the set of valid quantum algorithms.

Let $\mu = (p_x)$ be a distribution on $[n]$ giving the probability for the marked element to be found at each location. We will be concerned with understanding the average number of queries to the input required to find x. This will only depend on μ, and will hence be termed the (average-case) query complexity of μ. The general model of average-case query complexity used here will be similar to that in [1], with some minor differences. We distinguish two models for the Search with Advice problem: a *known* model and an *unknown* model. In the known model, the probability distribution μ is known beforehand, and can be used to design the algorithm. In the unknown model, μ is not known, and the algorithm must query an oracle to gain information about μ.

We first define the known model. Let $\mathcal{A}$ be a valid algorithm, and let $T_\mathcal{A}(x)$ denote the expected number of queries to f used by $\mathcal{A}$, when x is the marked element. Note that, in order for this model to be interesting in the case where $\mathcal{A}$ is a quantum algorithm, intermediate measurements during the search process are allowed; otherwise, $\mathcal{A}$ would always use the same number of queries. Further, let $T_\mathcal{A}(\mu)$ be the expected number of queries to f used by $\mathcal{A}$, where the expectation is taken over both the distribution μ and (potentially) $\mathcal{A}$'s internal randomness. That is,

$$T_\mathcal{A}(\mu) = \sum_{x=1}^{n} p_x T_\mathcal{A}(x).$$

Finally, we define the main quantities of interest, the deterministic and quantum (respectively) average-case query complexities of μ.

$$\begin{aligned} D(\mu) &= \min_{\mathcal{A}\in\mathcal{D}} T_\mathcal{A}(\mu), \\ Q(\mu) &= \min_{\mathcal{A}\in\mathcal{Q}} T_\mathcal{A}(\mu). \end{aligned}$$

The restriction to algorithms that succeed with certainty makes this a zero-error (*Las Vegas*) notion of average-case query complexity. It is common to consider an alternative *Monte Carlo* model of query complexity where $\mathcal{A}$ is allowed to err with some constant probability (e.g. 1/3). Note that this would not change the model significantly in the case of the current problem: given a (classical or quantum) Monte Carlo search algorithm that uses t queries and outputs x with probability p, one can produce an algorithm that succeeds with certainty and uses an expected number of queries of at most $(t+1)/p$ [9, Exercise 1.3].

We now turn to the unknown model. In this scenario, as well as querying f, we allow $\mathcal{A}$ to sample from μ using an oracle. In the case of quantum algorithms, we allow the preparation of *quantum* samples; that is, in the quantum case we define an oracle O_μ, which performs the mapping

$$O_\mu|0\rangle = |\mu\rangle := \sum_{x=1}^{n} \sqrt{p_x}|x\rangle.$$

The algorithm is also given access to the inverse operation, O_μ^{-1}. We define $T^*_\mathcal{A}(\mu)$ as the expected total number of queries to f, O_μ and O_μ^{-1} used by $\mathcal{A}$ (a query to each oracle being counted as unit cost). The oracle O_μ may appear somewhat unrealistic. However, it can be implemented if one has the ability to sum the distribution μ over arbitrary ranges [7]. That is, given an efficient means of computing $\sum_{x=a}^{b} p_x$ for arbitrary a, b, one can implement O_μ efficiently.

Finally, we will make use of an exact variant of Grover's quantum search algorithm throughout this paper.

Theorem 1.1 (Grover [6], Høyer [8], Brassard et al [4])**.** *Given an unstructured list of n elements that contains a unique marked element, there is a quantum algorithm that finds the marked element with certainty using $\lceil \frac{\pi}{4}\sqrt{n}\rceil$ queries to the list. If the list is promised to contain either one or zero marked elements, the marked element can be found (or "no marked element present" returned) with certainty using one extra query.*

2 Search with a known probability distribution

In this section, we will assume that p_x is non-increasing with x (so the most likely place for the marked element to be is at the start of the list, etc.). With this assumption, the optimal classical algorithm to find x is simply to query $f(1)$ through $f(n)$ in turn, so the classical average-case query complexity can be written down as

$$D(\mu) = \sum_{x=1}^{n} p_x\, x. \tag{1}$$

Note that, classically, the algorithm obtains no benefit from the use of randomness. When μ is the uniform distribution, corresponding to having no information about the location of the marked item, Grover's algorithm (Theorem 1.1) achieves a quadratic reduction in average-case query complexity. However, naïve use of this algorithm does not give an advantage in the average-case setting in general. In the next section, we give a quantum algorithm which *does* significantly improve on the trivial classical algorithm above.

2.1 Geometric search algorithm

We now give an algorithm for the Search with Advice problem, which will turn out to be asymptotically optimal. The quantum component of this algorithm is in fact simply Grover search (Theorem 1.1). Informally, the algorithm consists of splitting the input into blocks which increase in size geometrically (hence its name) and performing Grover search on each block. Interestingly, the algorithm does not need to know the precise advice probability distribution to achieve its near-optimal query complexity: it suffices to be able to sort the probabilities in non-increasing order. The algorithm is parametrised by a constant k, which gives the ratio of the geometric progression. We optimise k below; however, changing k only affects the query complexity by a constant factor.

Algorithm 1: Geometric quantum search

```
Input: Advice distribution $\mu = (p_x)$ in non-increasing order; function $f : [n] \to \{0,1\}$ such
       that $f$ takes the value 1 on precisely one input $x$; real $k > 1$
Output: The marked element $x$
$start \leftarrow 1$;
$end \leftarrow 1$;
$step \leftarrow 0$;
while $start \leq n$ do
    perform exact Grover search for one or zero marked elements on subset $\{start, \dots, end\}$;
    if marked element found then
        return marked element;
    end
    $step \leftarrow step + 1$;
    $start \leftarrow end + 1$;
    $end \leftarrow \min(start + \lfloor k^{step} \rfloor - 1, n)$;
end
return error;
```

Proposition 2.1. *The average number of queries used by Algorithm 1, choosing $k = e \approx 2.718$, on an advice distribution $\mu = (p_x)$ is upper bounded by*

$$\pi e \sum_{x=1}^{n} p_x \sqrt{x}.$$

Proof. In the m'th iteration of the loop, the (at most) $\lfloor k^m \rfloor$ elements contained in the range

$$R_m = \{1 + \sum_{i=0}^{m-1} \lfloor k^i \rfloor, \dots, \min(\lfloor k^m \rfloor - \sum_{i=0}^{m-1} \lfloor k^i \rfloor, n)\} \tag{2}$$

will be searched. By Theorem 1.1, the Grover search step in this iteration uses $\left\lceil \frac{\pi}{4}\sqrt{\lfloor k^m \rfloor} \right\rceil + 1$ queries. Then, for any marked element $x \in R_m$, a total of at most

$$\sum_{s=0}^{m} \left(\left\lceil \frac{\pi}{4}\sqrt{\lfloor k^s \rfloor} \right\rceil + 1 \right) \leq 2(m+1) + \frac{\pi}{4} \sum_{s=0}^{m} k^{s/2}$$

queries will be used by Algorithm 1 to find x. It is clear from (2) that, for any $x \in R_m$, $m \leq \log_k x + 1$. The average-case query complexity is therefore upper bounded by

$$\sum_{x=1}^{n} p_x \left(2\log_k x + 4 + \frac{\pi}{4} \sum_{s=0}^{\lfloor \log_k x + 1 \rfloor} k^{s/2} \right),$$

and, estimating the inner sum by an integral, we obtain an upper bound of

$$\sum_{x=1}^{n} p_x \left(2\log_k x + 4 + \frac{\pi}{4} \int_0^{\log_k x + 2} k^{s/2}\, ds \right) = 2\sum_{x=1}^{n} p_x \left(\log_k x + 2 + \frac{\pi k}{4 \ln k}\sqrt{x} \right).$$

Picking $k = e$, and noting that $\ln x + 2 \leq \frac{\pi e}{4}\sqrt{x}$ for all $x > 0$, completes the proof. $\square$

2.2 Optimality of the geometric search algorithm

We now show that Algorithm 1 is in fact optimal, up to a constant factor. This result will rely on the following known exact bound on the query complexity of quantum search.

Theorem 2.2 (Grover [6], Zalka [12]). *Let $f : [n] \rightarrow \{0,1\}$ be a function that takes the value 1 on precisely one input x, and let $\mathcal{A}$ be a quantum search algorithm that uses T queries to f and outputs x with probability at least p, for all x. Then*

$$T \geq \left\lceil \frac{\arcsin\sqrt{p}}{2\arcsin(1/\sqrt{n})} - \frac{1}{2} \right\rceil,$$

and this number of queries is achieved by Grover's algorithm.

As stated, this bound involves worst-case query complexity (that is, the largest possible number of queries used by $\mathcal{A}$, on the worst possible input). In our setting, we will need to lower bound the *expected* number of queries used by $\mathcal{A}$ on the worst possible input. This can be done with the following proposition.

Proposition 2.3. *Let $\mathcal{A}$ be a valid quantum search algorithm such that $T_{\mathcal{A}}(x) \leq T$ for all x, for some T_0. Then*

$$T \geq \frac{0.206}{\arcsin 1/\sqrt{n}} - 0.316 \geq 0.206\sqrt{n} - 1.$$

Proof. Let $t_{\mathcal{A}}(x)$ be the random variable giving the number of queries used by $\mathcal{A}$ on input x. Thus $T_{\mathcal{A}}(x) = \mathbb{E}\, t_{\mathcal{A}}(x)$, where the expectation is taken over $\mathcal{A}$'s internal randomness. By Markov's inequality, for all x and all $0 < p < 1$,

$$\Pr[t_{\mathcal{A}}(x) \geq T_{\mathcal{A}}(x)/(1-p)] \leq (1-p).$$

Thus a quantum search algorithm $\mathcal{A}$ that uses an expected number of at most T queries on all x gives a bounded-error quantum search algorithm that uses at most $T/(1-p)$ queries on all x and succeeds with probability at least p: just run $\mathcal{A}$ until it has used $T/(1-p)$ queries, and if it has not output x, output a random integer between 1 and n. By Markov's inequality, this will succeed with probability at least p.

So, by Theorem 2.2, we have that for any $0 < p < 1$

$$T \geq (1-p)\left(\frac{\arcsin\sqrt{p}}{2\arcsin(1/\sqrt{n})} - \frac{1}{2}\right).$$

Performing numerical maximisation of the right-hand side over p, one finds that for large n the maximum is achieved at $p \approx 0.369$, which proves the proposition. □

Note that it is known that one can indeed achieve an expected query complexity that is somewhat less than the usual worst-case query complexity guaranteed by Grover's algorithm [3, 12]. By stopping and restarting Grover search, it is possible to find the marked element x using approximately $0.690\sqrt{n}$ expected queries on all x, whereas straightforward use of Grover's algorithm guarantees approximately $0.785\sqrt{n}$ queries.

We are now ready to prove that Algorithm 1 is asymptotically optimal.

Proposition 2.4. *Let $\mu = (p_x)$, $x \in [n]$ be an arbitrary probability distribution. Then*

$$Q(\mu) \geq 0.206 \sum_{x=1}^{n} p_x \sqrt{x} - 1.$$

Proof. Let $\mathcal{A}$ be a valid quantum search algorithm and assume that μ is non-increasing. We aim to lower bound $T_{\mathcal{A}}(\mu) = \sum_{x=1}^{n} p_x T_{\mathcal{A}}(x)$. By Proposition 2.3, there must exist a y such that $T_{\mathcal{A}}(y) \geq 0.206\sqrt{n} - 1$. Similarly, there must exist $y' \neq y$ such that $T_{\mathcal{A}}(y') \geq 0.206\sqrt{n-1} - 1$ (or $\mathcal{A}$ would be able to find a marked element in the set of all elements not equal to y, using a number of queries that violates Proposition 2.3). Iterating this argument, we see that for each k such that $1 \leq k \leq n$ there exists an x such that $T_{\mathcal{A}}(x) \geq 0.206\sqrt{k} - 1$. By a rearrangement inequality, this implies that

$$T_{\mathcal{A}}(\mu) \geq \sum_{x=1}^{n} p_x (0.206\sqrt{x} - 1)$$

and proves the proposition. □

2.3 Power law distributions

We now apply Algorithm 1 to a natural class of probability distributions: power law distributions. We will see that significant speed-ups can be obtained over any possible classical algorithm.

Proposition 2.5. *Let $\mu = (p_x)$, $x \in [n]$ be a probability distribution where $p_x \propto x^k$ for some constant $k < 0$. Then*

$$D(\mu) = \begin{cases} \Theta(n) & [-1 < k < 0] \\ \Theta(n/\log n) & [k = -1] \\ \Theta(n^{k+2}) & [-2 < k < -1] \\ \Theta(\log n) & [k = -2] \\ \Theta(1) & [k < -2] \end{cases}, \text{ and } Q(\mu) = \begin{cases} \Theta(\sqrt{n}) & [-1 < k < 0] \\ \Theta(\sqrt{n}/\log n) & [k = -1] \\ \Theta(n^{k+3/2}) & [-3/2 < k < -1] \\ \Theta(\log n) & [k = -3/2] \\ \Theta(1) & [k < -3/2] \end{cases}$$

Proof. Deferred to Appendix. □

Corollary 2.6. *There exists a probability distribution μ such that $D(\mu) = \Omega(n^{1/2-\epsilon})$ for arbitrary $\epsilon > 0$, but $Q(\mu) = O(1)$.*

Proof. Take $k = -3/2 - \epsilon$ in Proposition 2.5. (Indeed, any $k \in (-2, -3/2)$ gives a super-exponential separation between $D(\mu)$ and $Q(\mu)$.) $\square$

3 Unknown probability distribution

In this section we switch to a different model, where the algorithm does not know the advice distribution μ in advance, but must use an oracle to obtain information about this distribution. We begin by noting the somewhat counterintuitive fact that a classical algorithm that merely queries f according to samples from the distribution μ performs no better than an exhaustive search algorithm[1].

Indeed, consider a classical algorithm that consists of repeatedly obtaining a sample y from μ, then querying $f(y)$. If x is the marked element, the expected number of samples from μ required until x is found is exactly $1/p_x$ (assuming that $p_x > 0$). Thus the expected number of samples used is

$$\sum_{x=1}^{n} p_x \left(\frac{1}{p_x}\right) = n;$$

the algorithm might as well have just carried out an exhaustive search to find x. Being given access to a *quantum* oracle O_μ producing a coherent superposition corresponding to the distribution μ, however, will turn out to be very useful.

3.1 Quantum algorithm

Our quantum algorithm will be based on the *amplitude amplification* primitive of Brassard et al [4]. For completeness, an explicit definition of amplitude amplification is given below, as Algorithm 2.

Algorithm 2: Amplitude amplification [4]

```
Input: Function $f : [n] \to \{0,1\}$ such that $f$ takes the value 1 on precisely one input $x$;
       oracle operator $O_\mu : |0\rangle \mapsto |\mu\rangle$; inverse $O_\mu^{-1}$; positive integer $k$ (number of iterations)
Output: The marked element $x$, or fail
create initial state $|\mu\rangle = O_\mu|0\rangle$;
apply operator $-O_\mu I_{|0\rangle} O_\mu^{-1} I_{|x\rangle}$ $k$ times to $|\mu\rangle$;
measure in computational basis, obtaining outcome $y$;
if $f(y)=1$ then
    return $y$;
else
    return fail;
end
```

The notation $I_{|\psi\rangle}$ denotes reflection about the state $|\psi\rangle$; it is well-known that the operator $I_{|x\rangle}$, where $I_{|x\rangle}|y\rangle = (-1)^{f(y)}|y\rangle$, can be implemented using one query to f. The following result was shown by Brassard et al in [4] (with somewhat different terminology).

[1]This phenomenon was recently discussed in the somewhat different context of screening for terrorists [11].

Lemma 3.1. *Applying Algorithm 2 with k iterations returns the location of the marked element with probability* $\sin^2((2k+1)\arcsin\sqrt{p_x})$*, using $k+1$ queries to O_μ, k queries to O_μ^{-1}, and $k+1$ queries to f.*

We now use Algorithm 2 as a subroutine in an algorithm which finds the marked element with certainty, and takes advantage of O_μ to reduce the expected number of queries used. The algorithm is a modified version of previous "exponential searching" algorithms of Brassard et al [4], and Boyer et al [3]. The main difference is that the algorithm gives up after a certain number of iterations and reverts to the exact variant of standard Grover search [8, 4]. This change can make a significant difference to the overall query complexity. The algorithm is stated as Algorithm 3 below.

Algorithm 3: Quantum search with unknown probability distribution

```
Input: Function $f : [n] \to \{0,1\}$ such that $f$ takes the value 1 on precisely one input $x$;
       oracle operator $O_\mu : |0\rangle \mapsto |\mu\rangle$; inverse $O_\mu^{-1}$; real $k > 1$
Output: The marked element $x$
for $j = 0$ to $\lfloor \log_k \sqrt{n} \rfloor$ do
    sample from distribution $\mu$;
    if marked element found then
        return marked element;
    end
    pick $i$ uniformly at random from integers $\{0, \dots, \lfloor k^j \rfloor - 1\}$;
    perform $i$ iterations of amplitude amplification;
    if marked element found then
        return marked element;
    end
end
perform exact Grover search for one marked element on $[n]$;
return marked element;
```

It will turn out to be possible to give a close analysis of the expected query complexity of Algorithm 3, including constants (which we will round to integers; these could be optimised further). The analysis follows the approach taken by Boyer et al [3] to bound the performance of their quantum search algorithm for an unknown number of marked elements.

Proposition 3.2. *On input x, when called with $k \approx 1.162$, Algorithm 3 uses an expected number of at most* $\min\{83/\sqrt{p_x} + 4/3, 53\sqrt{n}\}$ *queries to each of f, O_μ, O_μ^{-1}.*

Proof. Deferred to Appendix. □

Corollary 3.3. *Let $\mathcal{A}$ denote Algorithm 3. Then there are constants K, L, M such that*

$$T^*_{\mathcal{A}}(\mu) \leq K \left(\sum_{x, p_x > 1/n} \sqrt{p_x} \right) + L\sqrt{n} \left(\sum_{x, p_x \leq 1/n} p_x \right) + M.$$

Proof. Apply Algorithm 3, and use Proposition 3.2 to calculate the average number of queries used with respect to the distribution μ. □

3.2 Power law distributions

As with the case of a known probability distribution, power law distributions provide a natural class of examples for search with an unknown probability distribution. For some of these distributions, Algorithm 3 can be used to obtain significant speed-ups over any classical algorithm, even one with complete knowledge of the distribution.

Proposition 3.4. *Let $\mu = (p_x)$ be a probability distribution where $p_x \propto x^k$ for some constant $k < 0$, and let $\mathcal{A}$ denote Algorithm 3. Then*

$$T^*_{\mathcal{A}}(\mu) = \begin{cases} O(\sqrt{n}) & [-1 \le k < 0] \\ O(n^{-(1/2+1/k)}) & [-2 < k < -1] \\ O(\log n) & [k = -2] \\ O(1) & [k < -2] \end{cases}$$

Proof. Deferred to Appendix. □

Acknowledgements

This work was supported by the EC-FP6-STREP network QICS. I would like to thank Aram Harrow for pointing out reference [11].

Appendix

In this appendix we collect some proofs from throughout the paper.

A.1 Proofs from Section 2

Proposition 2.5. *Let $\mu = (p_x)$, $x \in [n]$ be a probability distribution where $p_x \propto x^k$ for some constant $k < 0$. Then*

$$D(\mu) = \begin{cases} \Theta(n) & [-1 < k < 0] \\ \Theta(n/\log n) & [k = -1] \\ \Theta(n^{k+2}) & [-2 < k < -1] \\ \Theta(\log n) & [k = -2] \\ \Theta(1) & [k < -2] \end{cases}, \text{ and } Q(\mu) = \begin{cases} \Theta(\sqrt{n}) & [-1 < k < 0] \\ \Theta(\sqrt{n}/\log n) & [k = -1] \\ \Theta(n^{k+3/2}) & [-3/2 < k < -1] \\ \Theta(\log n) & [k = -3/2] \\ \Theta(1) & [k < -3/2] \end{cases}$$

Proof. From the statement of the proposition, $p_x = \alpha x^k$ for some α. We first estimate the normalising constant α. The constraint that $\sum_{x=1}^n p_x = 1$ implies that $1/\alpha = \sum_{x=1}^n x^k$. Estimating this sum by an integral, we have that

$$\int_1^n x^k \, dx \le 1/\alpha \le 1 + \int_1^n x^k \, dx,$$

implying that, for $k \neq -1$,

$$\frac{n^{k+1} - 1}{k+1} \le 1/\alpha \le \frac{n^{k+1} - 1}{k+1} + 1.$$

By (1), if it also holds that $k \neq -2$,

$$D(\mu) = \alpha \sum_{x=1}^n x^{k+1} \ge \alpha \int_1^n x^{k+1} \, dx \ge \frac{(n^{k+2} - 1)(k+1)}{(n^{k+1} + k)(k+2)}.$$

The upper bound on $D(\mu)$ is very similar. It is easy to see that this proves the deterministic half of the proposition, except for the cases $k = -1$ and $k = -2$, which can be verified directly. In the quantum case, by Proposition 2.1, for $k \neq -1, -3/2$,

$$Q(\mu) \leq \alpha\pi e \sum_{x=1}^{n} x^{k+1/2} \leq \alpha\pi e \left(1 + \int_1^{n+1} x^{k+1/2}\, dx\right) = \pi e \left(\frac{((n+1)^{k+3/2} + k + 1/2)(k+1)}{(n^{k+1} - 1)(k+3/2)}\right).$$

Again, the lower bound is similar and the special cases $k = -1$, $k = -3/2$ can be verified directly. □

A.2 Proofs from Section 3

In this section, we give the proof of Proposition 3.2, which bounds the performance of Algorithm 3. We will need a lemma of Boyer et al [3], which we translate into our terminology.

Lemma A.1 (Boyer et al [3]). *If an integer r is picked from the range $\{0, \dots, m-1\}$ uniformly at random, and r iterations of amplitude amplification are performed, the probability of finding the marked element is exactly*

$$P_m = \frac{1}{2} - \frac{\sin(4m \arcsin \sqrt{p_x})}{8m\sqrt{p_x(1-p_x)}}.$$

In particular, $P_m \geq 1/4$ whenever

$$m \geq \frac{1}{2\sqrt{p_x(1-p_x)}}.$$

We are now ready to prove Proposition 3.2. The proof is similar to a result of Boyer et al [3], but with somewhat more detail.

Proposition 3.2. *On input x, when called with $k \approx 1.162$, Algorithm 3 uses an expected number of at most* $\min\{83/\sqrt{p_x} + 4/3, 53\sqrt{n}\}$ *queries to each of f, O_μ, O_μ^{-1}.*

Proof. We upper bound the expected number of queries to f used by Algorithm 3, which implies the same bound on the number of queries to O_μ and O_μ^{-1}. The bound will be in terms of k, and eventually minimised over k such that $1 < k < 4/3$. However, changing k will only change the number of queries used by a constant factor. Let T_j denote the expected number of queries to f used if the marked element is found in the j'th iteration of the loop. Then

$$T_j = \sum_{i=0}^{j-1} \frac{\lfloor k^i \rfloor + 3}{2} \leq \frac{k^j}{k-1},$$

an inequality which holds for $1 < k < 4/3$ and can be proven by induction on j.

We first find an upper bound by considering the *worst-case* number of queries used. If it has not been found previously, the marked element is guaranteed to be found in the last, exact Grover search step. Thus the number of queries to f used is at most

$$\sum_{j=1}^{\lfloor \log_k \sqrt{n} \rfloor + 1} T_j + \left\lceil \frac{\pi}{4}\sqrt{n} \right\rceil \leq \frac{1}{k-1} \sum_{j=1}^{\lfloor \log_k \sqrt{n} \rfloor + 1} k^j + \left\lceil \frac{\pi}{4}\sqrt{n} \right\rceil \leq \left(\frac{k^2}{(k-1)^2} + \frac{\pi}{4}\right)\sqrt{n}.$$

This deals with one half of the statement of the proposition. For the remainder of the proof, we restrict to the case $p_x \geq 1/n$ (as the case $p_x \leq 1/n$ will be covered by the above bound), and also assume that $p_x \leq 3/4$; this assumption will be removed at the end.

We now assume that the "sample from distribution μ" step always fails (this can only increase the number of queries used). The expected number of queries to f used by Algorithm 3 is then upper bounded by

$$\sum_{j=0}^{\lfloor \log_k \sqrt{n} \rfloor} \left(\prod_{i=0}^{j-1} (1 - P_{\lfloor k^i \rfloor}) \right) P_{\lfloor k^j \rfloor} T_{j+1} + \left(\prod_{i=0}^{\lfloor \log_k \sqrt{n} \rfloor} (1 - P_{\lfloor k^i \rfloor}) \right) \left(T_{\lfloor \log_k \sqrt{n} \rfloor + 1} + \left\lceil \frac{\pi}{4} \sqrt{n} \right\rceil \right).$$

To bound this expression, we split the first sum into two parts. First, we have

$$\sum_{j=0}^{\lfloor \log_k 1/\sqrt{p_x} \rfloor} \left(\prod_{i=0}^{j-1} (1 - P_{\lfloor k^i \rfloor}) \right) P_{\lfloor k^j \rfloor} T_{j+1} \le T_{\lfloor \log_k 1/\sqrt{p_x} \rfloor + 1} \le \frac{k}{k-1} \frac{1}{\sqrt{p_x}}.$$

Using Lemma A.1 and the fact that $p_x \le 3/4$, it holds for $j \ge \lfloor \log_k 1/\sqrt{p_x} \rfloor + 1$ that

$$\prod_{i=0}^{j} (1 - P_{\lfloor k^i \rfloor}) \le \left(\frac{3}{4} \right)^{j - \lfloor \log_k 1/\sqrt{p_x} \rfloor}.$$

Thus

$$\begin{aligned}
&\sum_{j=\lfloor \log_k 1/\sqrt{p_x} \rfloor + 1}^{\lfloor \log_k \sqrt{n} \rfloor} \left(\prod_{i=0}^{j-1} (1 - P_{\lfloor k^i \rfloor}) \right) P_{\lfloor k^j \rfloor} T_{j+1} \\
&\qquad \le \frac{k}{k-1} \sum_{j=\lfloor \log_k 1/\sqrt{p_x} \rfloor + 1}^{\lfloor \log_k \sqrt{n} \rfloor} \left(\frac{3}{4} \right)^{j - \lfloor \log_k 1/\sqrt{p_x} \rfloor - 1} k^j \\
&\qquad \le \frac{k}{k-1} \left(\frac{4}{3} \right)^{\lfloor \log_k 1/\sqrt{p_x} \rfloor + 1} \sum_{j=0}^{\infty} \left(\frac{3k}{4} \right)^{j + \lfloor \log_k 1/\sqrt{p_x} \rfloor + 1} \\
&\qquad \le \frac{4k^2}{(k-1)(4-3k)} \frac{1}{\sqrt{p_x}},
\end{aligned}$$

and also

$$\begin{aligned}
&\left(\prod_{i=0}^{\lfloor \log_k \sqrt{n} \rfloor} (1 - P_{\lfloor k^i \rfloor}) \right) \left(T_{\lfloor \log_k \sqrt{n} \rfloor + 1} + \left\lceil \frac{\pi}{4} \sqrt{n} \right\rceil \right) \\
&\qquad \le \left(\frac{3}{4} \right)^{\lfloor \log_k \sqrt{n} \rfloor - \lfloor \log_k 1/\sqrt{p_x} \rfloor} \left(T_{\lfloor \log_k \sqrt{n} \rfloor + 1} + \left\lceil \frac{\pi}{4} \sqrt{n} \right\rceil \right) \\
&\qquad \le \left(\frac{3}{4} \right)^{\log_k \sqrt{n} - \log_k 1/\sqrt{p_x} - 1} \left(\left(\frac{k}{k-1} + \frac{\pi}{4} \right) \sqrt{n} + 1 \right) \\
&\qquad \le \left(\frac{4k}{3(k-1)} + \frac{\pi}{3} \right) \frac{1}{\sqrt{p_x}} + \frac{4}{3},
\end{aligned}$$

where we use again the restriction that $p_x \ge 1/n$. Combining these bounds gives the following overall upper bound on the expected number of queries used:

$$\left(\frac{k}{k-1} + \frac{4k^2}{(k-1)(4-3k)} + \frac{4k}{3(k-1)} + \frac{\pi}{3} \right) \frac{1}{\sqrt{p_x}} + \frac{4}{3}.$$

Minimising the bracketed expression over k using simple calculus gives that the minimum is found at $k \approx 1.162$; for this value of k, we obtain a bound on the expected number of queries used that is approximately

$$\frac{82.646}{\sqrt{p_x}} + \frac{4}{3}.$$

Finally, consider the case that $p_x \geq 3/4$. In this case, one can find a bound by assuming that only the sampling step in each iteration of the loop can succeed, and ignoring the amplitude amplification step. Using this assumption, the number of queries to f used is upper bounded by

$$\frac{3}{4} \sum_{j=0}^{\lfloor \log_k \sqrt{n} \rfloor} \left(\frac{1}{4}\right)^j (1 + T_j) + \left(\frac{1}{4}\right)^{\lfloor \log_k \sqrt{n} \rfloor + 1} \left(T_{\lfloor \log_k \sqrt{n} \rfloor + 1} + \left\lceil \frac{\pi}{4} \sqrt{n} \right\rceil \right),$$

which is readily seen to be upper bounded by

$$\frac{3}{(k-1)(4-k)} + \frac{k}{k-1} + 2 + \frac{\pi}{4}.$$

Inserting the previously found value of k, $k \approx 1.162$, gives an upper bound of an expected ≈ 16.500 queries used in this case, and completes the proof. □

Proposition 3.4. *Let $\mu = (p_x)$ be a probability distribution where $p_x \propto x^k$ for some constant $k < 0$, and let $\mathcal{A}$ denote Algorithm 3. Then*

$$T^*_{\mathcal{A}}(\mu) = \begin{cases} O(\sqrt{n}) & [-1 \leq k < 0] \\ O(n^{-(1/2+1/k)}) & [-2 < k < -1] \\ O(\log n) & [k = -2] \\ O(1) & [k < -2] \end{cases}$$

Proof. As in the proof of Proposition 2.5, $p_x = \alpha x^k$ for some α, and for $k \neq -1$,

$$\alpha \leq \frac{k+1}{n^{k+1} - 1}.$$

Define $x_0 = \max\{x \in [n] : p_x \geq 1/n\} = \lfloor (\alpha n)^{-1/k} \rfloor$. Then, by Corollary 3.3,

$$Q^*(\mu) \leq K\sqrt{\alpha} \sum_{x=1}^{x_0} x^{k/2} + L\,\alpha\sqrt{n} \sum_{x=x_0+1}^{n} x^k + M,$$

for some constants K, L, M, implying that for $k \neq -1$, $k \neq -2$,

$$\begin{aligned} Q^*(\mu) &\leq K\sqrt{\alpha}\left(1 + \int_1^{x_0} x^{k/2}\,dx\right) + L\,\alpha\sqrt{n} \int_{x_0}^{n} x^k\,dx + M \\ &\leq K\sqrt{\alpha}\left(1 - \frac{2}{k+2} + \frac{2x_0^{k/2+1}}{k+2}\right) + L\frac{\alpha\sqrt{n}}{k+1}(n^{k+1} - x_0^{k+1}) + M \\ &\leq K\sqrt{\frac{k+1}{n^{k+1}-1}}\left(1 - \frac{2}{k+2} + \frac{2x_0^{k/2+1}}{k+2}\right) + L\frac{\sqrt{n}(n^{k+1} - x_0^{k+1})}{n^{k+1}-1} + M. \end{aligned}$$

Now note that

$$x_0 \approx \left(\frac{n(k+1)}{n^{k+1}-1}\right)^{-1/k}$$

is $\Theta(n)$ for $k > -1$, and $\Theta(n^{-1/k})$ for $k < -1$. Inserting this into the previous expression we obtain the claimed results for the cases $k \neq -1$, $k \neq -2$. These remaining special cases can be verified directly. □

References

[1] A. Ambainis and R. de Wolf. Average-case quantum query complexity. *J. Phys. A: Math. Gen.*, 34:6741–6754, 2001. `quant-ph/9904079`.

[2] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. `quant-ph/9802049`.

[3] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschr. Phys.*, 46(4–5):493–505, 1998. `quant-ph/9605034`.

[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information: A Millennium Volume*, pages 53–74, 2002. `quant-ph/0005055`.

[5] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002.

[6] L. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325–328, 1997. `quant-ph/9706033`.

[7] L. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. `quant-ph/0208112`.

[8] P. Høyer. Arbitrary phases in quantum amplitude amplification. *Phys. Rev. A.*, 62:052304, 2000. `quant-ph/0006031`.

[9] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[10] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, 2000.

[11] W. H. Press. Strong profiling is not mathematically optimal for discovering rare malfeasors. *Proceedings of the National Academy of Sciences*, 106(6):1716–1719, 2009.

[12] C. Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A.*, 60(4):2746–2751, 1999. `quant-ph/9711070`.