Contents

1 Introduction **3**

2 Multi-armed bandits **5**
2.1 Formal definition 5
2.2 Strategies 9
2.3 Simulations? 13

3 Quantum computing **15**
3.1 Quantum states 15
3.2 Quantum operations 19
3.3 Quantum algorithms 24

4 Quantum bandits **29**
4.1 Best arm identification 29
4.2 One at a time and regrets 30

References **31**

Chapter 1

Introduction

Chapter 2

Multi-armed bandits

The multi-armed bandit (MAB) problem is a classic problem in reinforcement learning and probability theory. It poses a simple yet challenging problem, where the agent must sequentially choose between a number of arms (distributions) of which the mean reward is unknown, trying to maximise the cumulative reward. This poses a constant struggle between exploration and exploitation, where exploration is the process of trying out new arms, and exploitation is the process of sampling from the distribution with the highest average thus far.

Bandit problems were first considered in 1933 [1], but the term was not coined before 1952 [2]. Some of the many real-world setting where the multi-armed bandit problem is applicable are listed in table 2.1. Despite being a simple problem, its countless variants and applications make it not only a useful tool for real-world problems. Netflix uses MAB theory to recommend movies [3], Amazon for its website layout [4], Facebook for video compression [5] and Doordash to identify responsive deliverymen [6]. The problem and its variations are still being studied with several results in what follows being recent.

2.1 Formal definition

In the multi-armed bandit problem, a set of k distributions are given, each with an unknown mean μ_i. Denote the set of distributions by $\mathcal{A}$ and the distributions by P_i, $i \in \{1, 2, \ldots, k\}$. These distributions have unknown means $\{\mu_1, \mu_2, \ldots, \mu_k\}$. The agent is given a number of turns T, known as the (time-) horizon, and at each turn t, the agent selects an arm i_t whence it samples independently of previous samples, receiving a reward $r_{i_t} \sim P_{i_t}$. The goal is to maximise the expected sum of rewards.

Table 2.1: Some applications of the multi-armed bandit problem.

Application	Arms	Reward
Medical trials	Drugs	Patient health
Online advertising	Ad placements	Number of clicks
Website design	Layouts/fonts &c.	Number of clicks
Recommendation systems	Items	Number of clicks
Dynamic pricing	Prices	Profit
Networking	Routes, settings	Ping
Lossy compression	Compression settings	Quality preserved
Tasking employees	Which employee	Productivity
Finance	Investment options	Profit

To summarise:

Known: Number of arms k, number of rounds T

Unknown: Mean rewards $\{\mu_1, \mu_2, \ldots, \mu_k\}$, distributions $\{P_1, P_2, \ldots, P_k\}$

Game: For $t = 1, \ldots, T$: select arm i_t and receive reward $r_{i_t} \sim P_{i_t}$ (independently of previous samples)

Goal: Maximise $E(\sum_{t=1}^{T} r_{i_t})$

While the mean rewards always are unknown, some assumptions will be made about the distributions, of which some common ones are listed in table 2.2. Naturally, stronger assumptions lead to better algorithms.

2.1.1 Performance

For the analysis of algorithm performance, the regret is used. At round T, the regret is defined as

$$R(T) = \sum_{t=1}^{T} \mu^* - \mu_{i_t}, \tag{2.1}$$

where μ^* is the highest, optimal mean, and μ_{i_t} is the mean of the arm selected at time t. Denoting by $N_i(T)$ the number of times arm i is pulled

Table 2.2: Common assumptions made about MAB distributions.

Assumption	Description
Bernoulli	$r \sim \mathcal{B}(\mu)$
Gaussian, unit variance	$r \sim \mathcal{N}(\mu, 1)$
Gaussian, unknown variance	$r \sim \mathcal{N}(\mu, \sigma^2)$
Sub-Gaussian	$\exists C > 0 : P(\lvert r \rvert \geq \epsilon) \leq 2\exp(-\epsilon^2/C^2)$
Bounded value	$r \in [0, 1]$

and Δ_i the difference between the mean of arm i and the optimal mean, the regret can be rewritten as

$$R(T) = \sum_{i=1}^{k} N_i(T)\Delta_i. \tag{2.2}$$

Algorithms are often probabilistic, so often the expected regret is of more interest, given by

$$E(R(T)) = \sum_{i=1}^{k} E(N_i(T))\Delta_i. \tag{2.3}$$

Regret will be used interchangeably with expected regret, with the meaning being clear from the context. When discussing algorithms in general, the expected regret is of concern, while considering particular simulations, the regret is of interest. The number of turns T is often referred to as the (time-) horizon and will be assumed to be greater than k, such that all arms may be pulled. While T is assumed given, many algorithms are designed to work independently of it, being known as anytime algorithms.

2.1.2 Optimality

In the realm of multi-armed banditry, expressing optimality is fraught with difficulties. Any precise formulation is contingent upon not only the assumptions made, but also the particular instance, namely actual means and any other parameters.

In order to meaningfully define a lower bound, it is imperative to assume a reasonable algorithm. Otherwise, trivial policies, such as always pulling the first arm, would achieve zero regret, hindering any meaningful comparison. One may therefore impose asymptotic consistency, which by definition means that for all inferior arms and all $\alpha \in (0, 1)$,

$$E(R(T)) = o(T^\alpha), \tag{2.4}$$

where $N_i(T)$ is the number of times arm i have been pulled in the first T turns. For algorithms obeying this property, the Lai-Robbins bound holds [7], namely that

$$\lim_{T\to\infty} \inf \frac{E(N_i(T))}{\ln T} \geq \frac{1}{D(P_i||P^*)}, \tag{2.5}$$

where P_i is the reward distribution of arm i, P^* that of the optimal distribution and $D(\cdot||\cdot)$ is the Kullback-Leibler divergence. Effectively then, any asymptotically consistent algorithm will suffer a regret of at least $\Omega(\ln T)$.

The Lai-Robbins bound is instance-dependent through its dependence on the Kullback-Leibler divergences. An alternative bound is proved in [8], states that for all algorithms, given a fixed horizon T and number of arms K, there is at least one problem instance such that

$$E(R(T)) \geq \Omega(\sqrt{KT}). \tag{2.6}$$

This is truly a worst-case bound; good algorithms will typically perform much better.

2.1.3 Best-arm identification

An alternative problem is to find the best arm with as few turns as possible. In this version, a δ is given, and the goal is to find the best arm with probability at least $1 - \delta$. The metric here is how the turns needed grows with δ. Unlike regret minimisation, exploitation is less of a concern, but much theory can be transferred from the regret minimisation problem. Though less is gained from repeatedly pulling the assumed best arm, there is still incentive to investigate the better arms than the worse.

2.1.4 Bandit generalisations

The multi-armed bandit problem has numerous generalisations, including the non-stationary multi-armed bandit where the underlying reward distributions change over time, presenting a challenging environment for traditional algorithms developed for the standard, stationary multi-armed bandit problem. In this variant, agents must continuously explore and adapt to the changing environment. Other cases give the agent more info, such as letting it know what the rewards for all arms, were they pulled instead, would have been. Another area of study is the contextual multi-armed bandit problem, in which contextual information must be incorporated into the decision-making process for arm selection, adding a layer of complexity to the standard multi-armed bandit problem, particularly

Table 2.3: Comparison of strategies.

Strategy	Regret	Tuning	Ease of implementation
Random	Linear	NA	Easy
Greedy	Linear	NA	Easy
Epsilon-greedy	Linear	Difficult	Easy
Epsilon-decay	Logarithmic	Difficult	Easy
UCB	Logarithmic	Barely	Medium
Thompson	Logarithmic	Priors	Harder

useful for recommender systems, where the context is the user and their preferences. Moreover, the adversarial multi-armed bandit problem represents a significant departure from the standard, stochastic multi-armed bandit problem, where rewards are chosen by an adversary instead of following a stationary distribution. The infinite-armed variants, where the arm space is infinite but constrained by for example linearity, also have practical applications. While beyond the scope of this report, these generalisations of the multi-armed bandit problem represent important areas of study and much of the theory developed for the standard, stochastic multi-armed bandit problem can be extended to these problems as well[1].

2.2 Strategies

2.2.1 Explore-only

Pure exploration is obviously a suboptimal strategy, but it is a good baseline against which to compare. It can be implemented by selecting an arm uniformly or in order, but it will perform poorly either way. The arm-selection procedure is described by algorithm 1.

Algorithm 1 Random arm selection

Sample i from $\{1, \ldots, k\}$ uniformly
return i

[1] C.v. [8, 9].

It is easy that the expected regret is

$$R(T) = T\left(\mu^* - \frac{1}{k}\sum_{i=1}^{k}\mu_i\right), \tag{2.7}$$

which is necessarily linear in T. This motivates the search for an algorithm with sublinear regret.

2.2.2 Greedy

Going the other way, a greedy algorithm will always select the arm with the highest empirical mean. Here, all arms are tried N initial times, and the empirical means are used to select the best arm. Afterwards, the arm with the highest empirical mean is selected for all remaining turns. The arm-selection procedure is listed in algorithm 2, where $\hat{\mu}_i$ is the empirical mean of arm i.

Algorithm 2 Greedy arm selection

```
if t ≤ Nk then
    return (t mod k) + 1
else
    return argmax_{i=1,...,k} μ̂_i
```

With greedy selection, the expected regret is clearly still linear in the horizon, as there is a non-zero probability of selecting the wrong arm. Still, there is a chance of achieving zero regret and the constant factor is reduced compared to random selection. To improve hereupon, it is necessary to occasionally explore other arms, which leads into the epsilon-greedy algorithm.

2.2.3 Epsilon-greedy

The problem with the greedy algorithm is that it may be unlucky and not discover the best arm in the initial exploration phase. To mitigate this, the epsilon-greedy algorithm may be used. In this algorithm, the presumed best arm is pulled with probability $1-\epsilon$, while in the other ϵ proportion of the turns, an arm is pulled uniformly at random. This ensures convergence to correct exploitation as the horizon increases, and it will generally reduce the regret.

Still, with a constant ϵ, a constant proportion of the turns will be spent exploring, keeping the regret necessarily linear in the horizon. Choosing

ϵ is a trade-off between exploration and exploitation and can significantly affect the regret.

Algorithm 3 Epsilon-greedy arm selection

```
if t ≤ Nk then
    return (t mod k) + 1
else
    Sample u from [0, 1) uniformly
    if u < ε then
        Sample i from {1, ..., k} uniformly
        return i
    else
        return argmax_{i=1,...,k} μ̂_i
```

Epsilon-decay

To remedy the linear term in the regret, modifications to the epsilon-greedy algorithm have been proposed wherein ϵ is a function of the number of trials, t. Specifically, in order to achieve sublinear regret, it is necessary to decay ϵ towards zero. Decreasing ϵ over time makes intuitive sense; exploration is more crucial in the beginning stages of the algorithm, whereas exploitation is more important when the agent has more reliable estimates of the reward means. For example, one popular strategy is to set $\epsilon \sim 1/t$, which has been shown to achieve logarithmic regret [10]. It is worth noting, however, that the optimal decay rate depends on the specific problem instance, and achieving logarithmic regret can be challenging in practice [11].

2.2.4 UCB

The upper confidence bound (UCB) algorithm is a more sophisticated algorithm based on estimating an upper bound for the mean of each arm. One always chooses the arm whose upper confidence bound is highest, a principle known as 'optimism in the face of uncertainty'. This should make sense, as if the wrong arm appears best, it will be pulled more often and the empirical mean will be corrected, while the true best arm with its larger bound will eventually become highest and so pulled. When exploiting the actual best arm, the agent can trust it to be the best, as the confidence bound will remain above those of all the other arms.

Assuming rewards in $[0, 1]$ and using Hoeffding's inequality, one has

$$p = P\left(\mu_i > \hat{\mu}_i + \mathrm{UCB}_i\right) \leq \exp\left(-2N_i\mathrm{UCB}_i^2\right), \tag{2.8}$$

where UCB_i is the upper confidence bound for arm i and N is the number of times arm i has been pulled. Solving for UCB_i gives

$$\mathrm{UCB}_i = \sqrt{\frac{-\ln p}{2N_i}}, \tag{2.9}$$

and letting $p = t^{-4}$ gives

$$\mathrm{UCB}_i = \sqrt{\frac{2\ln t}{N_i}}, \tag{2.10}$$

which is a common choice for the upper confidence bound, leading to the UCB1-algorithm. In [10], it was shown that this algorithm achieves $O(\ln T)$ regret. Regardless of the assumptions made, the procedure follows as in algorithm 4. Many variants of the algorithm exist; different assumptions about the distributions change the confidence bounds. While the choice of p is arbitrary, it is less of nuisance than the choice of ϵ in the epsilon-greedy algorithm, with specific choices of p, such as the UCB1-algorithm, being well-studied and known to perform well, achieving logarithmic regret.

Algorithm 4 UCB arm selection

```
if t ≤ k then
    return t
else
    return argmax_{i=1,...,k} μ̂_i + UCB_i
```

2.2.5 Bayesian: Thompson sampling

Thompson sampling is a Bayesian approach to the multi-armed bandit problem, being the original approach to the problem [1]. This method is noteworthy for its ability to incorporate Bayesian modelling concepts into the fundamentally frequentist problem of multi-armed banditry. The idea is to sample from the posterior distribution of the means of the arms and pull the arm with the highest sample.

It was first in 2012 that Thompson sampling was proven optimal for Bernoulli rewards [12] with uniform priors. Also for Gaussian rewards, it was proven optimal [13] with uniform priors. Notably, the Jeffreys prior

Algorithm 5 Thompson sampling arm selection

for $i = 1, \ldots, k$ **do**
    Sample θ_i from $P(\mu_i | \{r_i^{(1)}, \ldots, r_i^{(N_i(t))}\})$
return $\text{argmax}_{i=1,\ldots,k} \theta_i$
Update posterior for arm i with reward r_i

was shown to be inadequate in achieving optimal regret, highlighting the importance of the prior selection.

One of the key advantages of Thompson sampling is that it can natively incorporate prior knowledge about the arms, whereas doing so with the above methods would require some sort of ad-hoc manipulation of the recorded rewards and arm pulls. Furthermore, empirical results indicate it achieving lower regrets than the other algorithms [12]. Still, Thompson sampling is not without its drawbacks. The algorithm can be computationally expensive, as it requires sampling from the posterior distribution for each arm at each time step. Even with conjugate priors, the computational costs of sampling will be higher than the simple computations required by UCB and epsilon-greedy algorithms.

2.3 Simulations?

Chapter 3

Quantum computing

The field of quantum computing is split into two main branches: the development of quantum hardware and the study of algorithms that use such hardware. Only the second branch is relevant for this thesis, and even so only a brief explanation is offered here. For more details, see [14] for a rigorous, complete description or [15] for an introduction focused on programming. Any reader should have a basic understanding of linear algebra and classical computing. Knowledge of quantum mechanics is not assumed, albeit certainly helpful.

3.1 Quantum states

3.1.1 The qubit

The quantum bit, the qubit, is the building block of quantum computing. Like the classical binary digit, it can be either 0 or 1. But being quantum, these are quantum states, $|0\rangle$ and $|1\rangle$[1], and the qubit can be in any superposition of these states. This follows from the first postulate of quantum mechanics[2], which states that an isolated system is entirely described by a normalised vector in a Hilbert space. For the qubit, this is the two-dimensional space where the states $|0\rangle$ and $|1\rangle$ are basis vectors, known as the computational basis states. Thus, the state of a qubit can

[1]The $|\cdot\rangle$ notation is known as a ket and is used in quantum mechanics to denote a quantum state. It is effectively a column vector. The inner product may be taken with a bra, $\langle\cdot|$, to give a scalar. These inner products are then denoted by $\langle\cdot|\cdot\rangle$. Similarly, outer products are well-defined and denoted by $|\cdot\rangle\langle\cdot|$.

[2]As they are laid out in [14].

be expressed as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \tag{3.1}$$

where $\alpha, \beta \in \mathbb{C}$. The only requirement is that the state is normalised, $|\alpha|^2 + |\beta|^2 = 1$. Normalisation is required due to the Born rule, as the absolute square of the coefficients is the probability of measuring the qubit in the corresponding basis state.

3.1.2 The Bloch sphere

A useful tool for visualising the state of a qubit is the Bloch sphere. First, it should be noted that states on the form eq. (3.1) are not physically unique, only the relative complex phase matters. There is a global phase which can not be observed, and so it is not physically relevant. Taking that and the normalisation into account, the state of the qubit can be expressed as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{3.2}$$

where $\theta, \phi \in \mathbb{R}$. Interpreting θ as the polar angle and ϕ the azimuthal angle, the state of the qubit can be identified with a point on a sphere. See fig. 3.1. The state $|0\rangle$ is typically thought of as the north pole of this sphere and $|1\rangle$ the south pole.

3.1.3 Mixed states and density operators

It is not only the superpositions of states that are important in quantum computing, but also the mixed states, states that are statistical ensembles of pure states. Pure states are those expressible as a single ket like eq. (3.1), while mixed states arise when the preparation the system is not perfectly known or when the system interacts with the environment. For the description of mixed states, the formalism of density operators is more useful than the state vector formalism. If there are no classical uncertainties, the state is pure, and the density operator can be expressed a single ket-bra, $\rho = |\psi\rangle\langle\psi|$. In a mixed state, however, some classical probabilities p_i are associated with the different pure states $|\psi_i\rangle$, and the state of the system is described by the density operator

$$\rho = \sum_{i=1}^{n} p_i\,|\psi_i\rangle\langle\psi_i| \tag{3.3}$$

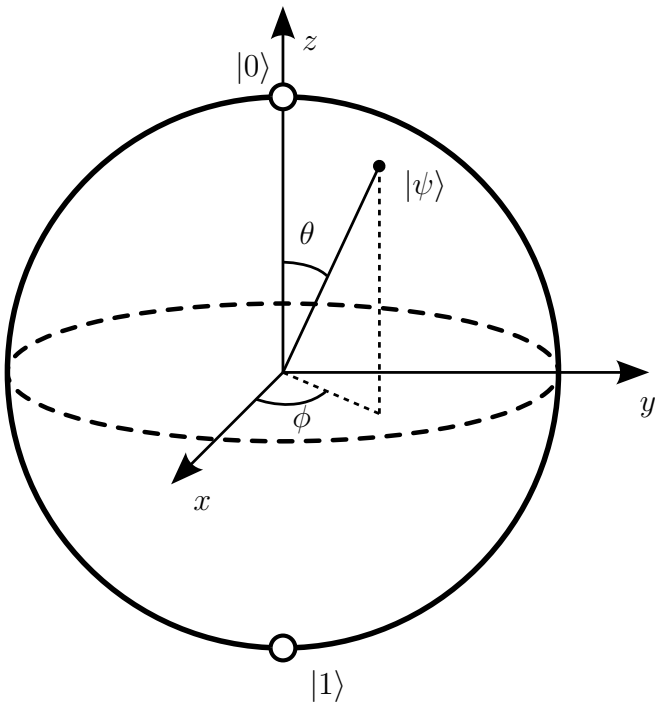


Figure 3.1: The Bloch sphere. On it, the state of a single qubit state is represented by a point. The state $|0\rangle$ is the north pole, and $|1\rangle$ is the south pole. The latitudinal angle θ determines the probability of measuring the qubit in the state $|0\rangle$, while the longitudinal angle ϕ corresponds to the complex phase between the two basis states. From [16].

where $|\psi_i\rangle$ are the states of the system, and $\langle\psi_i|$ are the corresponding dual vectors. Being probabilities, the p_i must be non-negative and sum to one. Given a basis and a finite Hilbert space, the density operator can be expressed as a density matrix[3] where the diagonal elements are the probabilities of measuring the system in the corresponding basis state. Furthermore, it is easily seen that the density operator must be positive semidefinite and Hermitian.

The Pauli matrices,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{3.4}$$

together with the identity matrix serve as a basis for the real vector-space of Hermitian 2×2-matrices. Since the diagonal elements of a density matrix must sum to one, the density matrix for a single qubit can be expressed as

$$\rho = \frac{1}{2}\left(I + x\sigma_x + y\sigma_y + z\sigma_z\right), \tag{3.5}$$

where $x, y, z \in \mathbb{R}$. Being positive semidefinite, the determinant should be non-negative, and thus it can be shown that $x^2 + y^2 + z^2 \leq 1$. This allows density operators to be interpreted as points on the Bloch sphere or indeed within it. Notably, pure states lie on the surface, while mixed states lie within the sphere (or rather, the Bloch ball). A pure quantum superposition of $|0\rangle$ and $|1\rangle$ with equal probabilities would have a complex phase and lie somewhere on the equator, while a statistical mixture with equal classical probabilities of being $|0\rangle$ and $|1\rangle$ would lie in its centre.

3.1.4 Systems of multiple qubits

Although the continuous nature of the qubit is indeed useful, the true power of quantum computers lie in how multiple qubits interact. Having multiple qubits enables entanglement, which is a key feature of quantum computing.

With two qubits, there are four possible states, $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Each of these four states have their own probability amplitude, and thus their own probability of being measured. A two-qubit system will therefore operate with four complex numbers in the four-dimensional Hilbert space $\mathbb{C}^4$.

[3]Density operators and matrices are often used interchangeably in quantum computing. Due to the finite number of qubits, the Hilbert spaces are always finite-dimensional, and with the canonical basis, there is a canonical way of representing density operators as matrices.

Generally, the state of multiple qubits can be expressed using the tensor product as

$$|\psi_1\psi_2\cdots\psi_n\rangle = |\psi_1\rangle\,|\psi_2\rangle\cdots|\psi_n\rangle = |\psi_1\rangle\otimes|\psi_2\rangle\otimes\cdots\otimes|\psi_n\rangle\,. \tag{3.6}$$

What makes this so powerful is that the state of a multi-qubit system has the general form

$$\begin{aligned}|\psi_1\psi_2\cdots\psi_n\rangle &= c_1\,|0\ldots00\rangle + c_2\,|0\ldots01\rangle + \cdots + c_{2^n}\,|1\ldots11\rangle \\ &= (c_1, c_2, \ldots, c_{2^n})^\top \\ &\in \mathbb{C}^{2^n},\end{aligned} \tag{3.7}$$

which means that with n qubits, the system can be in any superposition of the 2^n basis states. Operating on several qubits then, one can do linear algebra in an exponentially large space. This is a key part of the exponential speed-ups possible with quantum computers.

3.2 Quantum operations

3.2.1 Single-qubit gates

To operate on one or more qubits, a unitary operation is applied to the state. This is a computational interpretation of the unitary time evolution resulting from a Hamiltonian acting on the (closed) quantum system, described by the second postulate of quantum mechanics and the Schrödinger equation. As the operations are unitary, a pure state remains pure. These operations are often thought of as gates, paralleling the classical gates in digital logic. Mathematically, with a finite number of qubits, a unitary gate U can be expressed as matrices acting on the state vector, $|\psi\rangle$, as

$$|\psi'\rangle = U\,|\psi\rangle\,, \tag{3.8}$$

where $|\psi'\rangle$ is the resulting state.

The most basic gates are the Pauli gates, which are applications of the Pauli matrices from eq. (3.4) and are as gates simply denoted as X, Y, and Z. These gates can be seen as half turns around the x-, y- and z-axes, respectively, of the Bloch sphere. The X-gate is also known as the NOT gate, as it mirrors the classical NOT gate by mapping $|0\rangle$ to $|1\rangle$ and vice versa. It is however more general, being also applicable to superposition states.

The Hadamard gate,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{3.9}$$

is a rotation around the line between the x- and z-axes by $\pi/2$. It is an important gate in quantum computing, as it is used to create superpositions of the computational basis states. Applied on an initial $|0\rangle$ state, it creates the entangled state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Two consecutive applications thereof returns the state to the initial state, as can be seen from the matrix squaring to the identity.

The R_X-, R_Y- and R_Z-gates are rotations around the x-, y- and z-axes, respectively, by an arbitrary angle θ:

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$
$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$
$$R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}.$$

These parametrised gates will be useful in **??**.

3.2.2 Multi-qubit gates

Multi-qubit gates are gates that act non-trivially on more than one qubit. The most used multi-qubit gate is the controlled X-gate, also known as the CNOT. Being controlled means that it only acts on the second qubit if the first qubit is in the state $|1\rangle$. Of course, the first qubit may be in a superposition, and the CNOT this way allows for the creation of entanglement between the two qubits. If the first qubit has probability amplitude α of being in the state $|1\rangle$, the second qubit will have probability amplitude α of being flipped. The CNOT-gate, acting on the leftmost qubit in the tensored two-qubit system can be expressed in matrix form as

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.10}$$

In theory, any unitary single-qubit operation can be controlled. However, it is often only the CNOT that is used is implemented in the hardware. Another interesting two-qubit gate is the controlled Z-gate, CZ,

expressible as the matrix

$$\mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{3.11}$$

Because it only alters the amplitude of $|11\rangle$, it does not actually matter which qubit is the control and which is the target.

3.2.3 Observables and measurements

For an output to be obtained from a quantum computer, a measurement must be performed. This is typically done at the end of all operations and of all qubits, where each qubit is measured in the computational basis to yield a string of bits.

As described by the third postulate of quantum mechanics, any observable quantity has a corresponding Hermitian operator A, spectrally decomposable as $A = \sum_i \lambda_i P_i$, where λ_i are the (necessarily real) eigenvalues and P_i are the corresponding projectors onto the eigenspaces. When measuring, the probability of obtaining the outcome λ_i is given by

$$p_i = \langle\psi| P_i |\psi\rangle, \tag{3.12}$$

where $|\psi\rangle$ is the state before the measurement. It is one of Nature's great mysteries what exactly a measurement is and even more so how and why it is different from the unitary evolution described by the second postulate. In the quantum computational setting, it can be thought of as taking a random sample with the probabilities as given by the above equation.

Often, the underlying probabilities are what is of interest. Therefore, many measurements will be performed. Usually, these results are averaged to obtain an estimate, but more complicated post-processing methods are also possible. For instance, neural networks have shown useful properties in regard of reducing variance in the estimates, though at the cost of some bias [17].

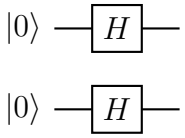
Canonically, the computational Z-basis is used for measurements, and it is usually the only basis for which measurements are physically implemented in a quantum computer. When measuring in the computational basis in which a state is expressed, as eq. (3.7), the probabilities are simply given by the absolute square of the coefficients. To virtually measure another observable, a change of basis is performed. This is achieved by applying a unitary transformation before measurement.

Measurements may be done in the middle of a computation and be used to control gates. If the qubits are entangled, measuring one will affect the measurement probabilities of others. Using such intermediate measurements is a way of introducing non-linearities in the otherwise unitary nature of the unmeasured quantum world.

3.2.4 Quantum circuits

The operations on qubits are often described using quantum circuits, which are a graphical representation of the operations on the qubits, the quantum algorithms. They are read from left to right. It is standard procedure to assume all qubits start in the state $|0\rangle$. Gates are generally written as boxes with the name of the gate inside.

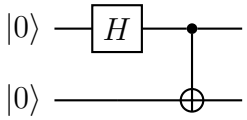
A simple example is the circuit



, (3.13)

which prepares the state $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This is a pure state with no entanglement, and so the measurement probabilities of the two qubits are independent. When measured, all four outcomes are equally likely.

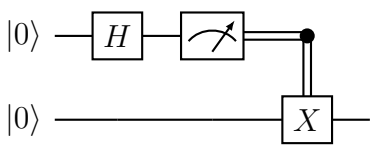
Slightly more interesting is the circuit



(3.14)

in which the first qubit is put into a superposition using the Hadamard gate before a CNOT gate is applied to the second, controlled by the first. This creates the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The measurement probabilities of the two qubits are now correlated; if the first qubit is measured to be $|1\rangle$, the second will always be $|1\rangle$ and vice versa. The probability of measuring the qubits to be different is nil.

To create a mixed state, an intermediate measurement can be used to control a gate. For instance, the circuit



(3.15)

places the second qubit in the mixed state $\frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$. If it were immediately to be measured, it would have a 50% chance of being $|0\rangle$ and a 50% chance of being $|1\rangle$. The uncertainty is only classical, and it could therefore not be used to create entanglement or for any other quantum spookiness.

3.2.5 Quantum supremacy

Exponential speed-ups do not come for free. Although the states spaces are exponentially large, with only a limited set of operations available, states can not be created and manipulated arbitrarily; the problem must have some structure to be exploited for a speed-up to be possible. Quantum computers do only solve certain problems more efficiently than classical computers, and finding the algorithms to do so is not trivial. Shor's algorithm has time complexity $O((\log N)^3)$ while the most efficient known classical algorithm, the general number field sieve, is sub-exponential with a time complexity on the form $\Omega(k^{\frac{1}{3}} \log^{2/3} k)$, where $k = O(2^N)$ [18]. To solve linear system, there is the HHL algorithm with time complexity $O(\log(N)\kappa^2)$, where κ is the condition number. This is an exponential speed-up over the fastest known classical algorithm[4], which has time complexity $O(N\kappa)$. Still, these are non-trivial algorithms, not yet usable in practice and that were not easily found.

Polynomial speed-ups are perhaps more easily found. For example, the Grover algorithm which is used to search for an element in an unsorted list has time complexity $O(\sqrt{N})$ [20]. Classically, this can not be done in less than $O(N)$ time. It can be proven that the Grover algorithm is optimal [21], so for this problem, an exponential speed-up is impossible. This algorithm and the more general amplitude amplification on which it builds solves very general problems and are often used subroutines to achieve quadratic speed-ups in other algorithms. Being only a quadratic speed-up, it is not as impressive as the exponential speed-ups, and achieving quantum supremacy in that manner would require larger quantum computers than if the speed-up were exponential.

It is proven that the class of problems quantum computers can solve in polynomial time (with high probability), BQP, contains the complexity class P [14]. This follows from the fact than quantum computers run do any classical algorithm. Since quantum computers can solve problems like

[4]Given that the condition number does not grow exponentially. There are also difficulties in loading the data into the quantum computer and extracting the solution that could negate any exponential speed-up. C.f. [19].

integer factorisation and discrete logarithms efficiently, it is believed that BQP is strictly greater than P, but as whether P = NP remains unknown, these problems could actually be in P. In a similar vein, NP-complete problems are believed to lie outside BQP.

3.3 Quantum algorithms

3.3.1 Grover's algorithm

The quantum search algorithm of Grover [20] is a quantum algorithm that finds an element in an unstructured list with high probability. While such a problem necessarily requires $O(N)$ time in a classical setting, needing on average $N/2$ steps to find the element and in the worst case N, Grover's algorithm finds the element in $O(\sqrt{N})$ steps. This is a quadratic speed-up.

Grover's algorithm is provably optimal; no quantum algorithm can perform such general searches faster [21]. This should not be surprising. If an exponential speed-up were possible, Grover search could be used to find the solution to NP-hard problems fast.

For Grover's algorithm to work, assume there is a function f that maps the index of an element to 1 if it is the one desired and 0 otherwise. Then, one assumes access to a quantum oracle, $\mathcal{O}_f$ (effectively a black box subroutine) that implements f thus:

$$\mathcal{O}_f \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle . \tag{3.16}$$

A single application of this oracle is not enough to find the desired element, as the square of the amplitude of the desired element remains unchanged. Central to Grover's algorithm is the idea of amplifying the amplitude of the desired element. This is done by applying a sequence of operations that is repeated until the amplitude of the desired element is large enough for it is most likely to be measured, while the amplitudes of the other elements are reduced.

Let the state $\left|w\right\rangle$ which be the winner state, a state with amplitude 1 for the desired element and 0 for all others. Then consider the state $\left|s\right\rangle$, which is a uniform superposition state, a state with equal amplitudes for all elements. Define the state $\left|s'\right\rangle$ by subtracting the projection of $\left|w\right\rangle$ onto $\left|s\right\rangle$ from $\left|s\right\rangle$:

$$\left|s'\right\rangle = \left|s\right\rangle - \left\langle w|s\right\rangle \left|w\right\rangle . \tag{3.17}$$

These two orthogonal states form a basis of a two-dimensional subspace of the greater Hilbert space. This permits a perspicuous visualisation of

the algorithm, as in fig. 3.2. The uniform superposition state $|s\rangle$ serves as a starting point for the algorithm, and is achieved by applying Hadamard gates to all qubits. It is expressible as

$$|s\rangle = \cos(\theta)\,|s'\rangle + \sin(\theta)\,|w\rangle\,, \tag{3.18}$$

where $\theta = \arcsin\langle s|w\rangle = \arcsin(1/\sqrt{N})$.

Applying the oracle on $|s\rangle$ leaves its $|s'\rangle$ component unchanged, but flips the sign of the $|w\rangle$ component. This results in the state $|\psi\rangle = \cos(-\theta)\,|s'\rangle + \sin(-\theta)\,|s'\rangle$, which can be seen as reflection of $|s\rangle$ in the $|s'\rangle$ direction.

Next, the state $|\psi\rangle$ is reflected about the initial $|s\rangle$ state, resulting in the state $|\psi'\rangle = \cos(3\theta)\,|s'\rangle + \sin(3\theta)\,|w\rangle$. Reflection thus is achieved by the diffusion operator

$$D = H^{\otimes n} S_0 (H^{\otimes n})^{-1} = H^{\otimes n} S_0 H^{\otimes n}, \tag{3.19}$$

where $S_0 = 2\,|0\rangle\langle 0| - I$ is the reflection operator about the $|0\rangle$ state, that is an operator that flips the sign of all but the $|0\rangle$ component.

The product of the oracle and the diffusion operator defines the Grover operator, which is simply applied until the amplitude of the $|w\rangle$ is sufficiently amplified. After k iterations, the state is $|\psi_k\rangle = \cos((2k+1)\theta)\,|s'\rangle + \sin((2k+1)\theta)\,|w\rangle$. Measuring the correct state has probability $\sin^2((2k+1)\theta)$. Therefore, $k \approx \pi/4\theta$ iterations should be completed. Assuming large N, for a short list would not warrant the use of Grover's algorithm, $\theta = \arcsin(1/\sqrt{N}) \approx 1/\sqrt{N}$, and so $k \approx \pi\sqrt{N}/4$.

For lists with more than a single desired element, a similar reasoning will lead to the same algorithm, but instead with $k \approx \pi/4\sqrt{N/M}$, where M is the number solutions to $f(x) = 1$ [14].

3.3.2 Amplitude amplification

Amplitude amplification can be considered a generalisation of Grover's algorithm. Instead of a single oracle, let the partitioning of the state space be given by a Hermitian projector P, whose image will be the space of states to amplify. Then, for some given initial state $|\psi\rangle$, it is decomposed into the orthogonal components

$$|\psi\rangle = \sin(\theta)\,|\psi_0\rangle + \cos(\theta)\,|\psi_1\rangle\,, \tag{3.20}$$

where $|\psi_1\rangle = P\,|\psi\rangle$ and $|\psi_0\rangle = |\psi\rangle - |\psi_1\rangle$, effectively the projections onto the image and kernel of P. Clearly, the angle θ is given by $\arcsin(|P\,|\psi\rangle\,|)$.

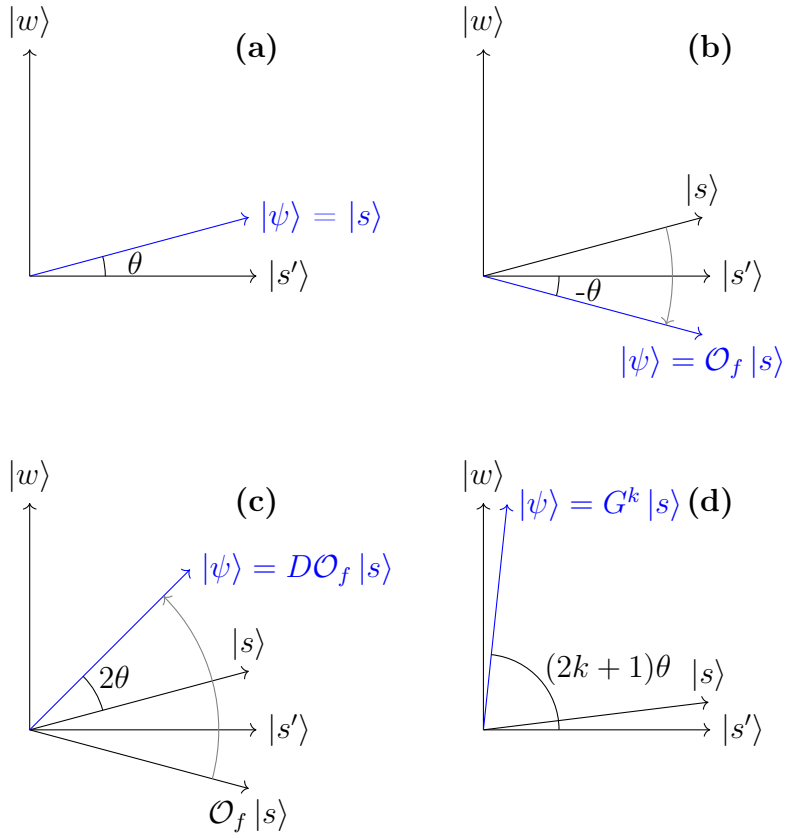


Figure 3.2: Grover's algorithm visualised. (a) The initial uniform superposition state $|s\rangle$ is prepared, which can be seen as a linear combination of $|w\rangle$ and $|s'\rangle$, forming an angle θ to the s'-axis. (b) The oracle $\mathcal{O}_f$ is applied to $|s\rangle$, flipping the sign of its $|w\rangle$ component, inverting the angle θ. (c) The diffusion operator D is applied, reflecting the state about the initial state and towards the goal, resulting in a state with an angle 3θ to the w-axis. (d) After repeating the previous two steps a k times, the angle is $2k + 1\theta$, and if k is chosen wisely, this means that the system is in a state close to the desired state $|w\rangle$, such that measuring the system will likely result in $|w\rangle$.

The Grover operator is now given by $G = -S_\psi S_P$ where

$$S_\psi = I - 2\left|\psi\right\rangle\left\langle\psi\right| \tag{3.21}$$
$$S_P = I - 2P, \tag{3.22}$$

such that S_ψ being analogue to the diffusion operator D and S_P being the oracle operator.

Following the same reasoning as in the previous section, the state after k iterations is given by

$$G^k\left|\psi\right\rangle = \sin((2k+1)\theta)\left|\psi_1\right\rangle + \cos((2k+1)\theta)\left|\psi_0\right\rangle, \tag{3.23}$$

meaning that $k \approx \frac{\pi}{4\theta}$ will result in a state close to $|\psi_1\rangle$.

Amplitude amplification can be used to speed up Grover search by using an informed prior rather than a uniform prior. Furthermore, it is useful as a subroutine in other algorithms, such as finding the number of 'good' states for a Grover search [22], quantum Monte Carlo methods [23] and some bandit algorithms to be discussed in what follows.

3.3.3 Amplitude estimation

As with amplitude amplification, amplitude estimation considers states that are decomposed into a superposition of two states, and, as the name suggests, estimates the amplitude of one of the states. Given a state, or more generally the algorithm, $\mathcal{A}$ with which it is generated,

$$\mathcal{A}\left|0\right\rangle = \sqrt{a}\left|\psi_1\right\rangle + \sqrt{1-a}\left|\psi_0\right\rangle, \tag{3.24}$$

where $|\psi_1\rangle$ is a state of interest and $|psi_0\rangle$ its orthogonal complement, the goal is to estimate its amplitude $a = |\langle\psi_1|\psi_1\rangle|^2$.

With its original formulation in [22], it was proven that the amplitude can be estimated with an additive error of

$$\epsilon \leq 2\pi\frac{\sqrt{a(1-a)}}{t} + \frac{\pi^2}{t^2} \tag{3.25}$$

with probability at least $8/\pi^2 (\approx 81.06\%)$ using t calls to the algorithm $\mathcal{A}$. The probability can be increased to $1-\delta$, requiring $O(\log(1/\delta))$ calls to $\mathcal{A}$ [23]. Later variants have been proposed to reduce the qubits needed and circuit depths [24, 25, 26], making it more feasible for NISQ devices, while still achieving similar asymptotic error bounds.

Chapter 4

Quantum bandits

Several formulations of the multi-armed bandit problem have been made for a quantum computing setting. As the central issue in bandit problems lie in sample efficiency rather than computational difficulties, quant-um computers offer little advantage assuming classical bandits. However, by granting some sort of superposition queries, means can be estimated more efficiently, and so regrets may be reduced, or best arms found more quickly.

Querying in superposition may at first appear to remove any real-world relevance; administering medications to patients can certainly not be done in superposition. However, with the training for reinforcement learning primarily being done in simulation, it is conceivable that the theory of quantum bandits may be applied to the learning of agents that are trained on quantum hardware and subsequently deployed to the real world.

4.1 Best arm identification

In [27], an algorithm based on amplitude amplification is proposed and is shown to find the optimal arm with quadratically fewer queries than the best classical algorithm for classical bandits — albeit with a significant drawback: the probability of the correct arm being suggested can not be set arbitrarily high, but is instead given by the ratio of the best arm's mean to the sum of the means of all arms. This greatly limits the usefulness of the algorithm, but it may still serve as a useful baseline for comparison, with the more complicated algorithms discussed in the following sections seeable as extensions hereof.

They assume access to an oracle O_e that encodes the probabilities of

the arms,

$$O_e : |x\rangle |0\rangle \mapsto |x\rangle \left(\sum_y \sqrt{p(y|x)} |y\rangle \right), \tag{4.1}$$

where x is the arm to be queried, y some internal state and $p(y|x)$ the probability of transitioning into state y given the pulling of arm x. For a given arm x and the resulting state $|y\rangle$, the reward is determined by some function $f(x, y) \to \{0, 1\}$, accessed through the phase oracle O_f,

$$O_f : |x\rangle |y\rangle \mapsto (-1)^{f(x,y)} |x\rangle |y\rangle . \tag{4.2}$$

For such bandits, regret minimisation is no longer a valid objective, as all arms are in a sense pulled simultaneously. Instead, the problem is to find a strategy that maximises the probability of finding the optimal arm with as few applications of $\mathcal{O}_\urcorner$ as possible.

4.1.1 Proper best arm identification

The authors of [28] propose a more sophisticated algorithm, improving the results of [27] by allowing the probability of finding the optimal arm to be set arbitrarily high.

4.2 One at a time and regrets

By instead having one oracle for each arm x,

$$O_x : |0\rangle \mapsto \sum_\omega \sqrt{p_x(\omega)} |\omega\rangle |y_x(\omega)\rangle , \tag{4.3}$$

where ω is the internal state of the arm, sampled according to the measure p_x, and $y_x\omega$ a random variable that is the reward received when pulling arm x and producing state ω. The agent then decides for each step in the bandit problem, which oracle to invoke, trying to minimise the cumulative regret, where the means here are given defined as

$$\mu_x = \sum_\omega p_x(\omega) \mathbb{E}[y_x(\omega)]. \tag{4.4}$$

In [29], an algorithm for bounded-value arms achieving $O(n \log T)$ regret was proposed, n being the number of arms.

References

[1] William R. Thompson. 'On the Likelihood That One Unknown Probability Exceeds Another in View of Evidence of Two Samples'. In: *Biometrika* 25.3-4 (1st Dec. 1933), pp. 285–294. DOI: 10.1093/biomet/25.3-4.285. URL: https://doi.org/10.1093/biomet/25.3-4.285 (visited on 12/11/2022).

[2] Herbert Robbins. 'Some Aspects of the Sequential Design of Experiments'. In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535. DOI: 10.1090/S0002-9904-1952-09620-8. URL: https://www.ams.org/bull/1952-58-05/S0002-9904-1952-09620-8/ (visited on 19/11/2022).

[3] Jaya Kawale and Elliot Chow. 'A Multi-Armed Bandit Framework for Recommendations at Netflix'. Data Council. 2018. URL: https://www.datacouncil.ai/talks/a-multi-armed-bandit-framework-for-recommendations-at-netflix.

[4] Daniel N. Hill et al. 'An Efficient Bandit Algorithm for Realtime Multivariate Optimization'. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17: The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax NS Canada: ACM, 13th Aug. 2017, pp. 1813–1821. ISBN: 978-1-4503-4887-4. DOI: 10.1145/3097983.3098184. URL: https://dl.acm.org/doi/10.1145/3097983.3098184 (visited on 12/02/2023).

[5] Sam Daulton. 'Facebook Talk at the Netflix ML Platform Meetup (Part 3)'. Machine Learning Platform (Los Gatos). Sept. 2019. URL: https://youtu.be/A-JJvYaBPUU.

[6] Arjun Sharma. *Using a Multi-Armed Bandit with Thompson Sampling to Identify Responsive Dashers*. DoorDash Engineering Blog. 15th Mar. 2022. URL: https://doordash.engineering/2022/03/15/using-a-multi-armed-bandit-with-thompson-sampling-to-identify-responsive-dashers/ (visited on 12/02/2023).

[7] T.L Lai and Herbert Robbins. 'Asymptotically Efficient Adaptive Allocation Rules'. In: *Advances in Applied Mathematics* 6.1 (Mar. 1985), pp. 4–22. DOI: 10.1016/0196-8858(85)90002-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0196885885900028 (visited on 13/02/2023).

[8] Aleksandrs Slivkins. 'Introduction to Multi-Armed Bandits'. In: *Foundations and Trends® in Machine Learning* 12.1-2 (2019), pp. 1–286. DOI: 10.1561/2200000068. URL: http://www.nowpublishers.com/article/Details/MAL-068 (visited on 05/11/2022).

[9] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. 1st ed. Cambridge University Press, 31st July 2020. ISBN: 978-1-108-57140-1 978-1-108-48682-8. DOI: 10.1017/9781108571401. URL: https://www.cambridge.org/core/product/identifier/9781108571401/type/book (visited on 19/11/2022).

[10] Peter Auer, Nicolò Cesa-Bianchi and Paul Fischer. 'Finite-Time Analysis of the Multiarmed Bandit Problem'. In: *Machine Learning* 47.2 (1st May 2002), pp. 235–256. DOI: 10.1023/A:1013689704352. URL: https://doi.org/10.1023/A:1013689704352 (visited on 12/11/2022).

[11] Sébastien Bubeck and Nicolò Cesa-Bianchi. 'Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems'. Version 2. In: (2012). DOI: 10.48550/ARXIV.1204.5721. URL: https://arxiv.org/abs/1204.5721 (visited on 05/11/2022).

[12] Emilie Kaufmann, Nathaniel Korda and Rémi Munos. 'Thompson Sampling: An Asymptotically Optimal Finite Time Analysis'. Version 2. In: (2012). DOI: 10.48550/ARXIV.1205.4217. URL: https://arxiv.org/abs/1205.4217 (visited on 13/02/2023).

[13] Junya Honda and Akimichi Takemura. 'Optimality of Thompson Sampling for Gaussian Bandits Depends on Priors'. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Vol. 33. Reykjavik, Iceland}: PMLR, Apr. 2014, pp. 375–383. arXiv: 1311.1894 [math, stat]. URL: http://proceedings.mlr.press/v33/honda14.pdf (visited on 13/02/2023).

[14] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 1st ed. Cambridge University Press, 5th June 2012. ISBN: 978-0-511-97666-7. DOI: 10.1017/CBO9780511976667. URL: https://www.cambridge.org/core/product/identifier/9780511976667/type/book (visited on 13/11/2022).

[15] Amira Abbas et al. *Learn Quantum Computation Using Qiskit*. 2020. URL: https://qiskit.org/textbook/.

[16] Smite Meister. *Bloch sphere*. 30th Jan. 2009. URL: https://upload.wikimedia.org/wikipedia/commons/6/6b/Bloch_sphere.svg (visited on 13/10/2022).

[17] Giacomo Torlai et al. 'Precise Measurement of Quantum Observables with Neural-Network Estimators'. In: *Physical Review Research* 2.2 (17th June 2020), p. 022060. DOI: 10.1103/PhysRevResearch.2.022060. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.022060 (visited on 04/03/2022).

[18] Danial Dervovic et al. 'Quantum Linear Systems Algorithms: A Primer'. 2018. DOI: 10.48550/ARXIV.1802.08227. arXiv: 1802.08227 [quant-ph]. URL: https://arxiv.org/abs/1802.08227 (visited on 30/11/2022).

[19] Scott Aaronson. 'Read the Fine Print'. In: *Nature Physics* 11.4 (Apr. 2015), pp. 291–293. DOI: 10.1038/nphys3272. URL: http://www.nature.com/articles/nphys3272 (visited on 30/11/2022).

[20] Lov K. Grover. 'A Fast Quantum Mechanical Algorithm for Database Search'. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. Philadelphia, Pennsylvania, United States of America: ACM Press, 1996, pp. 212–219. ISBN: 978-0-89791-785-8. DOI: 10.1145/237814.237866. URL: http://portal.acm.org/citation.cfm?doid=237814.237866 (visited on 30/11/2022).

[21] Christof Zalka. 'Grover's Quantum Searching Algorithm Is Optimal'. In: *Physical Review A* 60.4 (1st Oct. 1999), pp. 2746–2751. DOI: 10.1103/PhysRevA.60.2746. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746 (visited on 06/12/2022).

[22] Gilles Brassard et al. 'Quantum Amplitude Amplification and Estimation'. In: *Contemporary Mathematics*. Ed. by Samuel J. Lomonaco and Howard E. Brandt. Vol. 305. Providence, Rhode Island: American Mathematical Society, 2002, pp. 53–74. ISBN: 978-0-8218-2140-4 978-0-8218-7895-8. DOI: 10.1090/conm/305/05215. URL: http://www.ams.org/conm/305/ (visited on 07/02/2023).

[23] Ashley Montanaro. 'Quantum Speedup of Monte Carlo Methods'. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (Sept. 2015), p. 20150301. DOI: 10.1098/rspa.2015.0301. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.2015.0301 (visited on 06/02/2023).

[24] Yohichi Suzuki et al. 'Amplitude Estimation without Phase Estimation'. In: *Quantum Information Processing* 19.2 (Feb. 2020), p. 75. DOI: 10.1007/s11128-019-2565-2. arXiv: 1904.10246 [quant-ph]. URL: http://arxiv.org/abs/1904.10246 (visited on 07/02/2023).

[25] Kouhei Nakaji. 'Faster Amplitude Estimation'. In: *Quantum Information and Computation* 20 (1st Nov. 2020), pp. 1109–1123. DOI: 10.26421/QIC20.13-14-2.

[26] Dmitry Grinko et al. 'Iterative Quantum Amplitude Estimation'. In: *npj Quantum Information* 7.1 (19th Mar. 2021), p. 52. DOI: 10.1038/s41534-021-00379-1. arXiv: 1912.05559 [quant-ph]. URL: http://arxiv.org/abs/1912.05559 (visited on 07/02/2023).

[27] Balthazar Casalé et al. 'Quantum Bandits'. In: *Quantum Machine Intelligence* 2.1 (June 2020), p. 11. DOI: 10.1007/s42484-020-00024-8. URL: https://link.springer.com/10.1007/s42484-020-00024-8 (visited on 19/11/2022).

[28] Daochen Wang et al. 'Quantum Exploration Algorithms for Multi-Armed Bandits'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11 (11 18th May 2021), pp. 10102–10110. DOI: 10.1609/aaai.v35i11.17212. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17212 (visited on 19/11/2022).

[29] Zongqi Wan et al. 'Quantum Multi-Armed Bandits and Stochastic Linear Bandits Enjoy Logarithmic Regrets'. Version 1. In: (2022). DOI: 10.48550/ARXIV.2205.14988. URL: https://arxiv.org/abs/2205.14988 (visited on 05/11/2022).