Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Reinforcement learning with simulated annealing and genetic algorithm for quantum nonlocal games

Bachelor Thesis

2021
Ján Pastorek

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Reinforcement learning with simulated annealing and genetic algorithm for quantum nonlocal games

Bachelor Thesis

Study Programme:	Applied Informatics
Field of Study:	Informatics
Department:	Department of Applied Informatics
Supervisor:	Mgr. Daniel Nagaj, PhD.

Bratislava, 2021
Ján Pastorek

85457298

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	
Študijný program:	informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov:

Anotácia:

Vedúci:	
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.
Dátum zadania:	11.12.2017
Dátum schválenia:	12.12.2017

prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

....
študent

....
vedúci práce

85457298

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title:

Annotation:

Supervisor:	
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	11.12.2017
Approved:	12.12.2017

prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

....
Student

....
Supervisor

Acknowledgments: We are thankful to Daniel Nagaj for supervising this bachelor thesis, introducing us to the quantum computing and for interesting discussions. We are also thankful to Pavel Petrovic for useful discussion about the reinforcement learning.

Abstrakt

Nelokálne hry sú kľúčovým pojmom v odbore kvantovej informácie, využívané od teórie zložitosti po certifikáciu kvantových zariadení. Zahŕňajú dvoch alebo viacerých hráčov, ktorí sa rozhodnú pre stratégiu, a potom musia prestať medzi sebou komunikovať. Hru vyhrajú, ak poskytnú správne korelované odpovede na otázky. Typickým príkladom je hra CHSH súvisiaca s Bellovými nerovnosťami pre klasické korelácie, ktoré môžu byť porušené kvantovou mechanikou. V tejto hre majú kvantoví hráči vyššiu pravdepodobnosť výhry ako klasickí hráči. Stanovenie optimálnej pravdepodobnosti výhry je v skutočnosti zložitým problémom. V tejto práci skúmame rôzne nelokálne hry a hľadáme optimálne kvantové stratégie pomocou strojového učenia (učenia s posilňovaním), simulovaného žíhania a genetického algoritmu. Prehľadali sme všetky hry pre dvoch hráčov s dvoma otázkami, kde hráči zdieľajú jeden previazaný pár, a našli sme viac hier bez jednoduchej interpretácie, ale s kvantovou výhodou. Zistili sme, že genetický algoritmus je efektívnym prístupom k vyšetrovaniu hier, keď majú hráči k dispozícii iba jeden qubit. Sľubné výsledky vykazuje aj učenia s posilňovaním pomocou simulovaného žíhacieho prístupu.

Kľúčové slová: Reinforcement learning, nonlocal games, simulated annealing, genetic algorithm, quantum computing

Abstract

Nonlocal games are a key concept in quantum information, utilized from complexity theory to certification of quantum devices. They involve two or more players that decide on a strategy, and then must stop communicating among themselves. They win the game if they provide properly correlated answers to questions. The typical example is the CHSH game, related to Bell inequalities for classical correlations that can be violated by quantum mechanics. In this game, quantum players have a higher winning probability than classical players. Actually determining the optimal winning probability is a difficult problem in general. In this paper, we investigate a variety of nonlocal games and search for optimal quantum strategies with the help of machine learning (reinforcement learning), simulated annealing and genetic algorithm. We searched all 2-player, 2-question games where players share one entangled pair and found multiple games with no simple interpretation but showing a quantum advantage. We find that genetic algorithm is an effective approach for investigating games when players have only one qubit at their disposal. Reinforcement learning with simulated annealing approach shows promising results as well.

Keywords: Reinforcement learning, nonlocal games, simulated annealing, genetic algorithm, quantum computing

Chapter 1

Introduction

Every piece of information is always stored on some physical medium (paper, magnetic tape, brain ...). Thus, to study information, we need to know how the physical medium behaves, so that we can fully utilize its potential knowledge. In the last decades, we have found that we can store information in quantum states of quantum mechanical systems, such as the spin of a particle, potentially giving us new ways of harnessing this physical medium. Since then, we have already started utilizing knowledge from quantum phenomena in various quantum algorithms. Superposition is one such phenomenon that we have started mining. When a particle is in a superposition, it means it is not in a simple basis state, nor just a random mixture of two basis states. It is best described as a vector of amplitudes of different basis states. For a two-dimensional system (with two basis states), we get a qubit – a quantum system that can be 0 and 1 at the same time. We need a vector of amplitudes of the basis states instead of a simple bit to describe such a state, e.g.

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \text{ but also } \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \text{ and } \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}.$$

However, it gets much more complicated when we describe many particles.

Another important phenomenon is entanglement. Particles that interacted and since this interaction, they can not be reduced to states of individual particles (the whole is more than just the sum of its parts) are called entangled.

Cleverly using superpositions of such states, finding ways to add desirable amplitudes for computational paths, while subtracting undesirable ones (entanglement) lets us get surprising algorithmic advances.

Two of the famous ones with speedups over classical computation are the algorithms of Grover and Shor. Grover's algorithm is a provably optimal algorithm for searching an unstructured database (an unordered list) that can find a marked element in $O(\sqrt{N})$ queries.

Shor's quantum factoring algorithm for integer factoring is able to factor a number stored with N bits in N^3 steps, while the best known classical one is exponential in N.

To understand this in detail, let us start by explaining how physicists got there.

1.1 A Short History of Quantum Mechanics

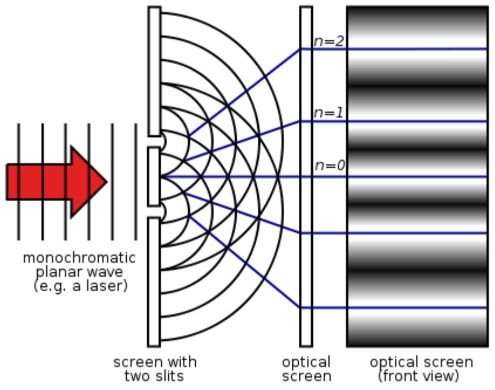
It all started with classical mechanics not being able to match our observations on both very large and very small scales. So, it could neither be used to predict exact trajectories of planets, [15] nor properties of atomic matter [9]. One of the main problems was the ultraviolet catastrophe. Classical models predicted that an ideal black body at thermal equilibrium will emit radiation in all frequency ranges, emitting more energy as the frequency increases. However, this would lead all matter to radiate all of its energy until it would reach absolute zero temperature, which was obviously an error at the heart of the theory.

Physicists were of course looking for ways to fix this, but some took a wholly different path. It was just at this moment that Max Planck proposed that energy is absorbed and emitted in small packets called quanta. Hence the name 'quantum physics'. Calculations based on this hypothesis have matched data from observations of black-body radiation and could eradicate the ultraviolet catastrophe at the expense of abandoning a classical description. Planck's discovery was coined as the birth of quantum mechanics. There were other observations that supported Planck's hypothesis such as the photoelectric effect. Einstein's Nobel prize came for explaining this effect by assuming that light is absorbed and reemitted in quanta as well.

Quantum mechanics also helped shed light on the long standing problem of the paradoxical duality of light. Sometimes, it behaves like a particle (with observable momentum, a view championed by Newton), sometimes as a wave (with interference effects, as described by Huygens). At the beginning of the 19th century, Young performed one of the foundational experiments that demonstrate the wave-like nature of light.

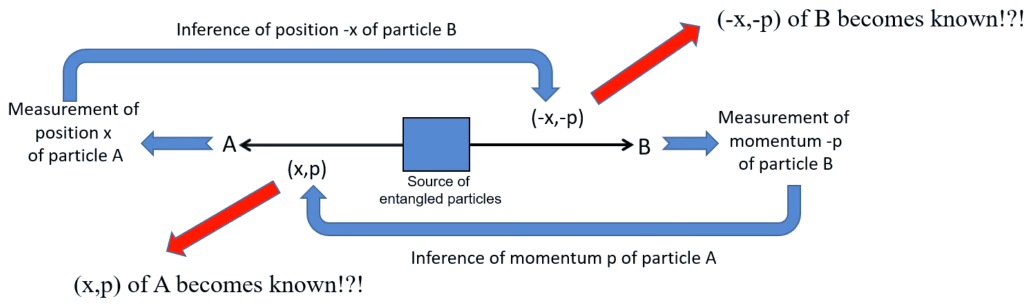
Let us look at this experiment in detail. The setup is as in Fig. 1.1.1. Prepare an optical screen and place another screen with two slits parallel to the optical screen. Then you begin the experiment by sending a light wave (nowadays a laser) in the direction of screens. While crossing the slits, the light will diffract and then interfere with the light passing through another slit causing a light/dark pattern similar to the one shown on the figure. At some places, the light will interfere constructively, producing "white" spots on the optical screen. At other places, the light will interfere destructively and produce "black" spots. However, if the light was made from particles, it should not produce interference patterns like waves do. Therefore, Young concluded that light has a wavy nature.

However, already since Newton, it was also useful to view light also as a stream of particles.



n=2
n=1
n=0
monochromatic planar wave (e.g. a laser)
screen with two slits
optical screen
optical screen (front view)

Figure 1.1.1: The double slit experiment [25].



Inference of position -x of particle B
(-x,-p) of B becomes known!?!
Measurement of position x of particle A
A
(x,p)
Source of entangled particles
(-x,-p)
B
Measurement of momentum -p of particle B
Inference of momentum p of particle A
(x,p) of A becomes known!?!

Figure 1.1.2: The original EPR paradox [16].

Basically two camps arose: One that was saying that light is a wave, the other saying that light is a particle. Surprisingly, experiments supported both views. The way light behaved depended on context.

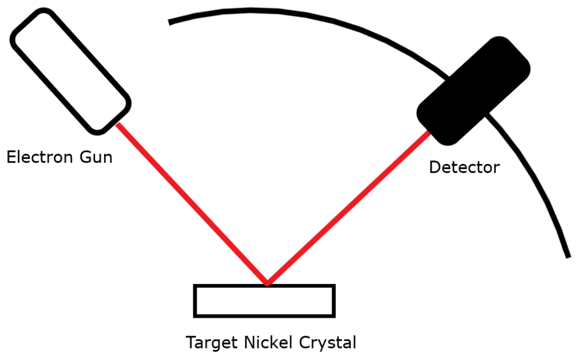


Figure 1.1.3: Davisson-Germer experiment.

Roughly a century later, Davidson and Germer performed the same experiment [8], but now with particles – electrons. A schematical setup can be seen in 1.1.3: an electron gun, a target Ni crystal and a movable detector. Electrons were fired on the target Ni crystal.

After the electrons were scattered by the surface of crystal, they displayed diffraction (wavy) patterns observed with a detector moving along an arc.

The surprising result was the wave-like nature of all the matter, as witnessed by interference effects. This could be explained only by quantum mechanics, and showed how indeterminate and probabilistic our reality actually is.

Questions like: 'What is the true nature of light?', 'How to interpret our observations?', 'Is everything observable?', 'Do our observations change reality?' bothered physicists for further decades. Many interpretations of quantum reality arose. Some have set limits to our knowledge, others assumed that reality is just an illusion.

Even more strange behaviour in the micro world was predicted and later observed. In his thought experiments Heisenberg showed that we can't determine the position and the momentum of a particle at the same time with arbitrary precision. Physicists today argue this is because the momentum and position observables/quantities do not commute. Another aspect of this phenomenon manifests as the fundamental wave-particle duality of light. Once we measure a photon's position with high precision (say with a small pinhole), its momentum will become uncertain, and the photon will necessarily diffract as a wave would. Such wave-particle duality is a fundamental and inescapable law of nature.

Throughout this time, Einstein has believed that quantum mechanics is an incomplete theory and was gathering ideas to demonstrate it. In one of these, known as the EPR paradox, Einstein, Podolsky and Rosen argued that it could be in principle possible to determine the value of two non-commuting observables with certainty and without disturbing the system. They set up the following thought experiment (see Fig. 1.1.2):

1. Make an entangled state with two particles, whose momenta (p) and positions (x) are tied to each other – (e.g. by having a larger particle decay into two identical ones, moving in opposite directions).

2. Let the two particles fly very far from each other.

3. Measure one quantity on the first particle and the other quantity on the second particle.

4. Knowing the position of the first particle lets us infer the position of the other. Meanwhile, knowing the momentum of second particle would let us infer the momentum of the first one. This way, we could infer missing information about the non-commuting observables – momentum and position.

   Thus we get *complete* information about the quantum system and contradiction with Heisenberg's findings.

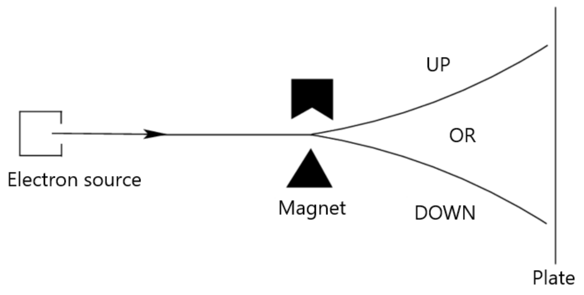


Figure 1.1.4: The Stern-Gerlach experiment [1].

To illustrate the EPR paradox in a more comprehensible way, we will move from continuous variables to discrete values and use 2-dimensional quantum systems – qubits.

A physical example of such a system is the spin of an electron. For now, think of the spin of particle as the rotation of particle around its own axis, just as the Earth rotates around its own axis. A particle can rotate around a particular axis in two directions, and we call these states spin up and spin down. This feature of fermionic quanutm particles (such as the electron) was discovered by Stern and Gerlach [11] in their experiment, depicted in Fig. 1.1.4.

They sent particles through a region of inhomogeneous magnetic field. They found that the particles will go either to upper part (spin up) or lower part (spin down) of the plate after they are influenced by the magnets, and not continuously between the two extreme values. This quantization (two discrete values of spin) was surprising, as for a classical magnet, the magnetic moment in a particular direction can take continuous values, depending on the orientation of the magnet.

In quantum computation, we denote spin up by the state vector of a qubit as

$$|\uparrow\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

and spin down as

$$|\downarrow\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Let us now recast the EPR paradox for the spins of two qubits/particles, instead of position and momentum, which are continuous, and thus more complicated observables. We depict it in Fig. 1.1.5.

1. Make an entangled (singlet) state with two particles, whose spins around two axes (z and x) are tied to each other (see (**??**)). It has the following interesting property: its form is the same in the z and the x basis. The values of the spin of the two particles in the z direction are opposite, and the same is true for the x direction.

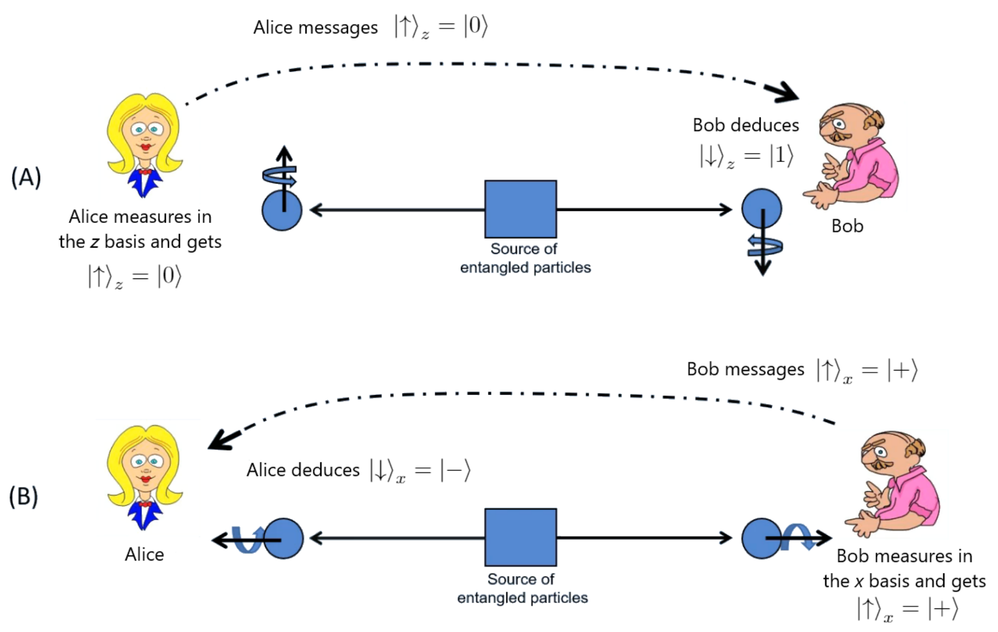

Alice messages $|\uparrow\rangle_z = |0\rangle$
Bob deduces $|\downarrow\rangle_z = |1\rangle$
(A)
Alice measures in the *z* basis and gets $|\uparrow\rangle_z = |0\rangle$
Source of entangled particles
Bob
Bob messages $|\uparrow\rangle_x = |+\rangle$
Alice deduces $|\downarrow\rangle_x = |-\rangle$
(B)
Alice
Source of entangled particles
Bob measures in the *x* basis and gets $|\uparrow\rangle_x = |+\rangle$

Figure 1.1.5: The EPR paradox with spins [16]

2. Let the two particles fly very far from each other.

3. Measures spin in the z direction on the first particle and spin in the x direction on the second particle.

4. Knowing the spin in the z direction of the first particle lets us infer the spin in the z direction of the second one. Meanwhile, knowing the spin in the x direction of the other particle lets us infer the spin in the x direction of the first one.

   Thus we get *complete* information about the quantum system.

Obviously, this thought experiment was somehow violating Heisenberg's uncertainty principle, aiming to determine the precise value of non-commuting observables. As a consequence of the EPR paradox, it must follow that either QM is not a complete theory (the wavefunction is not a complete description of reality, and some hidden variables are at play) or these observables cannot have simultaneous reality – or even worse, faster than light communication (FTL) is at work. FTL was in contradiction with Einstein's special relativity, where the velocity of light is limited to circa 300 000 m/s. Observables not having simultaneous reality was unthinkable for a deterministic universe. Hence, Einstein concluded that QM is not a complete theory. [10]

However, we believe Einstein was not correct. Based on observations and backed by theory, today we still claim that two observables/quantities cannot have simultaneous reality, and that the measurement on one observable makes the values of another observable indeterminate, if the observables are non-commuting. So, the way Einstein inferred knowledge about the second particle by observing the first was flawed.

Could not FTL be at work? Quantum mechanics describes everything through wave functions. These means that everything is probabilistic. Composite systems are also described by a wavefunction. If we now move parts of the system far from each other, the wavefunction description remains valid. If we now measure one part, the whole wave function collapses instantly, according to the the standard interpretation of quantum mechanics. This seems like a nonlocal effect – possibly enabling FTL communication.

It looked like some information was missing. However, no direct FTL was at work because the particles just did not send any other particles to communicate between them. They just inherently "knew" what state to be in. Further investigation into this strange reality was done by John Stewart Bell, who in 1964 came up with a series of inequalities. [3] He calculated that if we assume that our universe is locally realistic (two physical systems can influence each other only when they are close enough), correlations between measurement values on two separated particles are limited. However, these inequalities can be broken when using quantum mechanical calculations. His conclusions were later experimentally demonstrated by Alain Aspect [2], who measured the violation of Bell's inequality for two entangled particles, and thus showed that the quantum meachnical description of nature is necessary.

What is then the true nature of our reality? In order to not to come to contradictions, we had to abandon either realism or locality. If we abandon realism, it means that we abandon the idea that there exist objective, independent physical systems. But what are we living in, an illusion? Does not an illusion presuppose reality? On the other hand, if we abandon locality, in principle it means (only) that physical systems can "interact" without sending any piece of information at any distance. It can be shown that quantum mechanics does not allow instantaneous communication; this property is called *no signaling.*

It might be useful for a while to let this discussion be, and turn to the "shut up and calculate" approach. We will see, how fruitful it will be, analyzing nonlocal scenarios involving communication: games. We will look back at what we did and return to interpretations later in Section 9

1.2 Nonlocal games

In this Section, we will describe a class of communication games that showcases the power of quantum correlations. Indeed, the violation of Bell inequalities by quantum mechanics can be demonstrated using *nonlocal games*, where quantum players can win with a higher probability than classical ones.

Nonlocal games are a key concept in quantum information, utilized from complexity theory to certification of quantum devices. They involve two or more players that win if they provide properly correlated answers to questions. The typical example is the CHSH game, illustrated in Fig.1.2.1.

The CHSH game

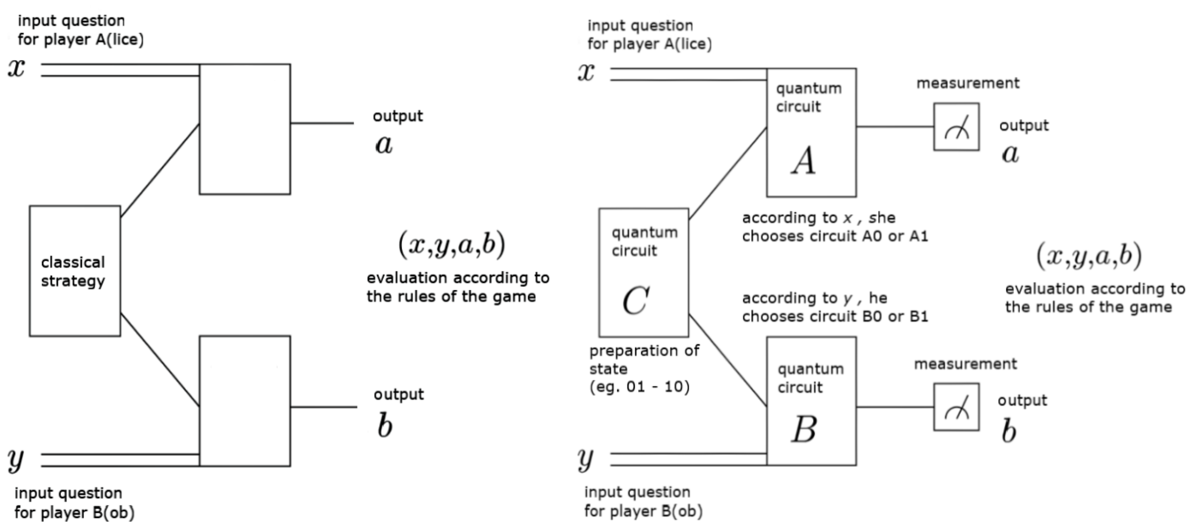

input question for player A(lice)
x
output
a
classical strategy
(x,y,a,b)
evaluation according to the rules of the game
output
b
y
input question for player B(ob)
input question for player A(lice)
x
quantum circuit
A
measurement
output
a
quantum circuit
C
according to x , she chooses circuit A0 or A1
according to y , he chooses circuit B0 or B1
(x,y,a,b)
evaluation according to the rules of the game
preparation of state (eg. 01 - 10)
quantum circuit
B
measurement
output
b
y
input question for player B(ob)

Figure 1.2.1: The CHSH game for a) classical players, and b) quantum players.

1. Alice and Bob are respectively given one random bit (questions x and y), and they each respond with one bit (answers a and b).

2. If either of them gets a question 0 ($x = 0$ or $y = 0$), they win if they answers agree ($a = b$).

3. If they both get 1 they win by giving opposite answers ($a \neq b$).

After agreeing on a joint strategy, Alice and Bob take a brief space voyage before getting their questions, and giving their answers. They will not be able to coordinate their answers after getting their input because they need to send their response back to Earth within a short time period. It's not just against the rules for them to communicate, it's physically impossible due to the finite speed of light.

What would make a good strategy for Alice and Bob? One possible strategy is to always output 1, regardless of input. Then they will win whenever either one of them receives a 0. If the inputs are given at random, they will win 75% of the time. Another strategy would be for Bob to always output 1, and for Alice to output 1 if her input is 1, and 0 otherwise. This also has a 75% chance of winning. It turns out that when the input is random, 75% is the best you can do classically. This can be easily calculated by going through all deterministic strategies, and realizing that any probabilistic strategy is a mixture of deterministic strategies, so no probabilistic strategy can be better than the best deterministic one. Since Alice and Bob have each 2 different possible inputs and each of them can provide on the basis of input 2 possible outputs, each has 4 possibilities. There are two players, Alice and Bob, so there are just 16 different possible deterministic strategies.

However, in a quantum world (see Fig. 1.2.1 - b), Alice and Bob can prepare and share an entangled quantum state which they will bring with them on their voyage. They still cannot communicate, as the finite speed of light forbids communication. They can, however, coordinate their answers better and win with more than 75% probability. When Alice and Bob learn their input questions, they each apply a unitary gate to their half of their shared maximally entangled state. Then they measure their qubits, and answer 0 if their qubit was up and 1 if their qubit was down. We claim that they will do better than is physically possible classically. We will do the calculation in detail in Section 3.2.

Actually, determining the optimal winning probability is a difficult problem in general, when the questions, answers, and the evaluation of the game involve a large number of bits n, instead of just a single bit in the CHSH game. [24]

Brute force is in these cases literally impossible. Usually, approximation algorithms can be quite successful in dealing with similar tasks. However, if we want to say something about the quantum value (the winning probability of the optimal strategy) for the game, we need to do optimization in the quantum domain. There, even the evaluation of a quantum strategy is hard to simulate classically (although it could be done easily on a quantum computer). We know that to simulate even small quantum circuit, we must calculate 2^n x 2^n matrices. And looking for optimal quantum circuits is even harder. That is why we are looking for some shortcut that could get atleast suboptimal results.

A new approach for this optimalization task is needed. We are going to look at this optimization task with the help of an universal approximator, a neural network – with reinforcement learning. In short, we want to let our reinforcement learning agent learn paths that increase the winning probability.

The purpose of this work is thus to explore ways of approximating the optimal winning probability using reinforcement learning and genetic algorithm. We will use reinforcement learning and genetic algorithm to learn strategies that maximize the *quantum* value of a nonlocal game.

Chapter 2

Quantum Mechanics

"If you are not completely confused by quantum mechanics, you do not understand it" – John Wheeler

"Quantum mechanics is simply this: it is a set of four postulates that provide a mathematical framework for describing the universe and (possibly) everything in it." [17]

These postulates state how to describe a physical system, how it evolves, how to measure it and how to combine physical systems.

Postulate 1: State space

How do we describe a physical system?

Every physical system has an associated complex vector space (a Hilbert space). This vector space is to be called the state space *of a system.*

A system is isolated *when it is not interacting with any other system. If a physical system is isolated, then the system is completely described by unit vector in this system's state space. This vector is to be called* state vector.

We have already seen one such state in the introduction, the state vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ of a qubit, representing spin up. However, a state vector can be any vector from the state space with length equal to 1. This length can be calculated as sum of the norms squared of the amplitudes of the state vector. This is known as the *normalization constraint.* The amplitudes are the complex coefficients of the state vector.

$$\sum_i |x_i|^2 = 1 \tag{2.0.1}$$

This just the same as dot product, which can be written in ket notation as $\langle\psi|\psi\rangle = 1$

So for instance, $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ has two amplitudes x_1 and x_2. The length of this state vector is $|x_1|^2 + |x_2|^2$ and this sum must be equal to 1 in order for a vector to be properly normalized.

We have mentioned in the introduction that a qubit can be both 0 and 1 at the same time. This is known as *superposition.* Superposition is just another term for a linear combination. In our case of a qubit, it is a linear combination of the two basis states $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. To illustrate superposition, we can have a state vector

$$0.6 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}$$

It is a state vector because $0.6^2 + 0.8^2 = 1$ holds. The interpretation of this superposition state is that when we measure it in the z basis, we can get the "up" state $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ with probability 0.6^2, and the "down" state $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ with probability 0.8^2.

This showcases that the quantum universe is probabilistic and why we can handwavingly say that our qubit is 0 and 1 at the same time.

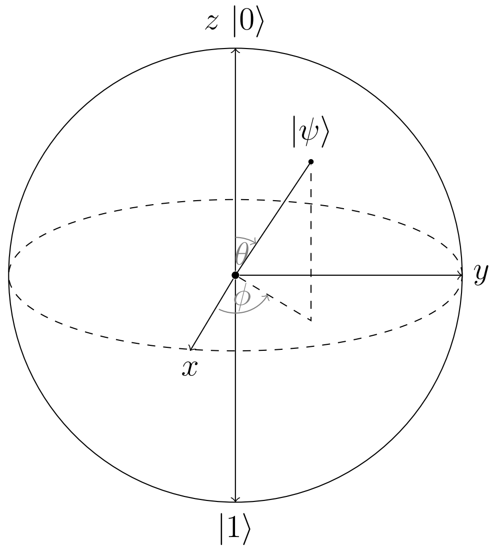
We can also write state vectors in a different notation – the *bra-ket notation*, which is used much more in the community. We label

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

A superposition can be written like this:

$$0.6 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = 0.6\,|0\rangle + 0.8\,|1\rangle$$

Because of the normalization constraint, we can depict the state of one qubit on the surface of a (Bloch) sphere, while if the coefficients are real, on a two-dimensional plane using just a simple unit circle. (see Fig. 2.0.1).



$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle \tag{2.0.2}$$

Figure 2.0.1: The state vector of a qubit on a Bloch sphere described by (2.0.2)

Postulate 2: Dynamics of a system

How do we describe the dynamics of an isolated quantum system?

If we have an isolated physical(quantum) system, we can describe its evolution by a unitary *matrix acting on the system's state space.*

This postulate says that the state of the system $|\psi_1\rangle$ at time t_1 is related to the state of the system $|\psi_2\rangle$ at another time t_2 by some unitary matrix U. This can be written $|\psi'\rangle = U\,|\psi\rangle$. While U is not dependent on any of those states, it could depend on times t_1 and t_2.

For a matrix to be unitary, it needs to satisfy

$$U^\dagger U = I \tag{2.0.3}$$

Where,

$$U^\dagger = (U^T)^*,$$

where † is the *dagger operation.* It first transposes the matrix and then takes a complex conjugate of its terms.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\dagger} = \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{T} \right)^{*} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}^{*} = \begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix}$$

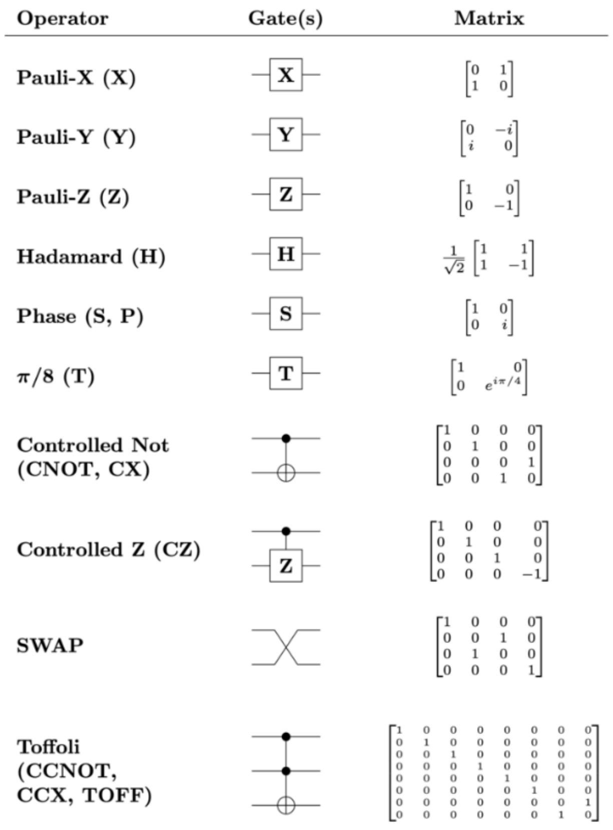
Why do we need to use just unitaries? Unitary matrices have a special attribute, they preserve the length of vectors after multiplication. Because they preserve the length of vectors, the sum of probabilities will remain equal to 1.

To illustrate this, take the state $|\psi_1\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and evolution described by the identity matrix $U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The evolved state will be $|\psi_2\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |\psi_1\rangle$. That is because we used the identity matrix as an example of a unitary matrix – it simply preserves the state.

On the other hand, if $U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, then $|\psi_2\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$, and the state is flipped from the "up" to the "down" basis state.

Although there are many unitaries we could look at, the set of Pauli matrices plays a prominent role in quantum mechanics and quantum computation. They were first developed as operators describing the spin of an electron and they are certainly worth remembering, just as the rest of the set of unitary matrices/gates in Fig. 2.0.2.

A quantum *gate* is a name for a local unitary used as a computational step in quantum circuits. We will be doing circuit simulation by calculating matrices acting on the state vectors. The process is similar to classical circuits where classical gates, such as NAND, "act" on the circuit.



Operator	Gate(s)	Matrix
Pauli-X (X)	$\boxed{X}$	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
Pauli-Y (Y)	$\boxed{Y}$	$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$
Pauli-Z (Z)	$\boxed{Z}$	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$
Hadamard (H)	$\boxed{H}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$
Phase (S, P)	$\boxed{S}$	$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$
$\pi/8$ (T)	$\boxed{T}$	$\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled Z (CZ)	$\boxed{Z}$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
SWAP		$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

Figure 2.0.2: Commonly used unitary matrices/quantum computation gates [22].

Postulate 3: The measurement of a system

How do we describe measurements on quantum systems?

Quantum measurements are described by a collection M_m of measurement operators*. Each M_m is a matrix acting on the state space of the system being measured. The index* m *takes values corresponding to the measurement outcomes that may occur in the experiment.*

If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2.0.4}$$

The *posterior state* is the state of the system after the measurement. A superposition of several eigenstates collapses/projects to a single eigenstate due to interaction with the measurement apparatus. We call this an observation. The result depends on the amplitudes that the system had before the measurement. We can get the posterior state by calculating

$$|\psi_m\rangle = \frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}} \tag{2.0.5}$$

The probability of measurement outcomes must be equal to 1, thus, measurement operators must satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I \tag{2.0.6}$$

Then,

$$I = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2.0.7}$$

Measurement operators are matrices made from the outer product one state with itself. *Measurement of qubit in the computational basis* is one type of measurement. Since this is a single qubit measurement, it can have two outcomes, and so two measurement operators.

$$M_0 = |0\rangle\langle 0|, \quad M_1 = |1\rangle\langle 1|$$

(These are projections on the $|0\rangle$ and $|1\rangle$ vectors, or collectively, a projective measurement in the z basis.)

Suppose we have the state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$$

Let us measure it in the computational basis and calculate the probability of outcome 0. We will construct the corresponding operator M_m, where $m = 0$, by the outer product of state $|m\rangle = |0\rangle$

$$M_0 = |0\rangle\langle 0| = \begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}$$

and get the corresponding probability of outcome 0 by

$$p(0) = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} |\psi\rangle =$$

$$\langle\psi| \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} |\psi\rangle = \langle\psi| M_0 |\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = |\alpha|^2$$

the posterior state, the state after measurement, is according to (2.0.5)

$$|\psi_0\rangle = \frac{M_0 |\psi\rangle}{\sqrt{\langle\psi| M_0^\dagger M_0 |\psi\rangle}}$$

We have already calculated the denominator, which is $|\alpha|^2$, therefore

$$\begin{aligned} |\psi_0\rangle &= \frac{M_0 |\psi\rangle}{\sqrt{\langle\psi| M_0^\dagger M_0 |\psi\rangle}} = \frac{M_0 |\psi\rangle}{\sqrt{|\alpha|^2}} \\ &= \frac{1}{\sqrt{|\alpha|^2}} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{|\alpha|} \begin{bmatrix} \alpha \\ 0 \end{bmatrix} = \frac{\alpha}{|\alpha|} |0\rangle . \end{aligned}$$

Likewise, we would calculate probability of outcome 1 to be $|\beta|^2$, while the state after this measurement would collapse to $|1\rangle$ with probability.

Note that (2.0.6) is also satisfied, since

$$\begin{aligned} M_0 &= |0\rangle \langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \\ M_1 &= |1\rangle \langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \end{aligned}$$

and $M_0 + M_1 = I$.

Postulate 4: Combining systems

How do we describe the state space of composite quantum systems?

The state space of a composite physical system is described by the tensor product of the state spaces of the component physical systems.

To illustrate this, take two physical systems, e.g. two qubits, in the states $|\psi_1\rangle = |0\rangle$ and $|\psi_2\rangle = |1\rangle$. If we want to describe them as a part of a composite physical system, we will use this postulate. Therefore,

$$
\begin{aligned}
|\psi_1\rangle \otimes |\psi_2\rangle = |0\rangle \otimes |1\rangle &= \begin{bmatrix} x_{\psi_1} \\ y_{\psi_1} \end{bmatrix} \otimes \begin{bmatrix} x_{\psi_2} \\ y_{\psi_2} \end{bmatrix} \\
&= \begin{bmatrix} x_{\psi_1}.x_{\psi_2} \\ x_{\psi_1}.y_{\psi_2} \\ y_{\psi_1}.x_{\psi_2} \\ y_{\psi_1}.y_{\psi_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}
$$

However, the situation is harder to interpret when we now take linear combinations of such states, as is allowed by quantum mechanics. For example, the already mentioned singlet state

$$
\frac{1}{\sqrt{2}} \left(|0\rangle |1\rangle - |1\rangle |0\rangle \right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}. \tag{2.0.8}
$$

Surprisingly, it can not be written as a tensor product of individual basis states, i.e. it doesn't have the form $|\psi_1\rangle \otimes |\psi_2\rangle$. We call such states *entangled*, and they will be key to providing quantum correlations that help us beat the Bell inequalities, and play nonlocal games better than classical players ever could.

Now that we have the preliminaries and foundations, we can delve into our problem of interest: multiplayer games that require correlated strategies, and how quantum mechanics can help us play them better.

Chapter 3

Nonlocal games

"Entanglement is at the heart of quantum mechanics. The nonlocal correlations that can be obtained from space-time separated measurements on an entangled state are a central feature which provably distinguish it from local theories."[24]

We have briefly mentioned the importance of entanglement in Chapter 1. Another argument for its importance comes from the study of nonlocal games, where it can be used as a resource. The CHSH game from Chapter 3.2 is one of those simple demonstrations of how entanglement can give us an advantage over classical theories.

In general, multiplayer nonlocal games/interactive proofs (MIP) are interactive games where a polynomial-time referee/verifier interacts with the players/provers to verify their answers/proofs. To verify their proofs, the verifier can ask only a limited number of questions.

Another useful analogy is a teacher-student relationship. The verifier is a teacher questioning two or more students individually about some claim, using questions, chosen randomly from some set, with each question composed of two or more "subquestions". Each student gets one subquestion and is asked to answer. Their answers are deemed convincing, iff they collectively and consistently meet the teacher's rules of evaluation.

The provers are not allowed to communicate once their interaction with the verifier has started. In order to "win", they will always try to convince the verifier about their claim. Therefore, the game should have a soundness property:

For all false claims, the interactions should be such that no provers' answers could convince the verifier.

The simple CHSH game from Chapter 3.2 is only a small example/instance of a game in MIP, a huge class of games. In general, a language (game) L is in $\mathsf{MIP}(k, m, c, s) \iff$ there exists an m-turn interactive protocol for a polynomial-time verifier V and k provers $P_1, \ldots, P_k$ such that, for every input x:

- (Completeness) if $x \in L$, the interaction protocol of the verifier with the provers results in acceptance with probability at least c,
- (Soundness) if $x \notin L$, the probability that the interaction protocol ends with V accepting is at most s.

Nonlocal/entangled games (MIP^*) are an extension of multiplayer games, where we allow the provers to share quantum states (especially entangled pairs). For example, each player takes their respective part of an entangled pair with himself and can perform arbitrarily many rotations on his qubit. Then, the *quantum value* is the maximum of winning probabilities over all strategies using entangled pairs. If the quantum value of a nonlocal game is larger than its classical value, then the game exhibits *quantum advantage.*

To develop a good strategy, the players should involve the qubits at their disposal as well. They can agree on rotating qubits if they get such and such questions, entangling them with other qubits, and measuring them to get desired correlations that help them answer the verifier's questions more consistently. Note though, that they can not communicate once the game starts, so they must agree on the strategy before interacting with the referee. This allows us to study entanglement in a wholly new, computational context.

Finally, here's the formal definition of MIP^*. Formally, a language L is in $\mathsf{MIP}^*(k, m, c, s) \iff$ there exists an m-turn interactive protocol for a polynomial-time classical verifier V and k provers $P_1, \ldots, P_k$ sharing some quantum (entangled) state $|\psi\rangle$, such that, for every input x:

- (Completeness) if $x \in L$, the interaction protocol of the verifier with the provers results in acceptance with probability at least c,
- (Soundness) if $x \in L$, the probability that the protocol results in the verifier V accepting is at most s.

We will discuss the possible complexity of these games in Section 3.3. Before that, let us discuss the verifier's questions and the provers' strategies.

3.1 (Classical) entangled game

Let us formalize a 1-round game. Let G be a nonlocal game and

- A = set of all possible answers,
- Q = set of all questions,
- W = set of all strategies.

Then, $G = (V, \pi)$, where:

- V is a function $V : A^k \text{ x } Q^k \to \{0, 1\}$
- π is a probability distribution $\pi : Q^k \to [0, 1]$ for some $k \in \mathbb{N}$.

The verifier randomly samples questions $(q_1, \ldots, q_k)$ according to π and sends q_i to prover i from whom he gets an answer a_i. They win iff the verifier accepts their answers $a_1, \ldots, a_k$, if $V(a_1, \ldots, a_k|q_1, \ldots, q_k) = 1$.

A game G is *symmetric* when the outcome does not change after any permutation of the players (i. e. $V(\zeta(a_1, \ldots)|\zeta(q_1, \ldots)) = V(a_1, \ldots|q_1, \ldots)$ for any permutation ζ). It will be useful to use this symmetry to reduce the the number of games that we will look at in our numerical investigations.

The winning probability of the game is

$$p_{win} = \sum_{q_1,\ldots,q_k} \pi(q_1, \ldots, q_k) = \tag{3.1.1}$$
$$\sum_{a_1,\ldots,a_k} V(a_1, \ldots, a_k|q_1, \ldots, q_k)p(a_1, \ldots, a_k|q_1, \ldots, q_k).$$

The provers' strategies are $W_i : Q \text{ x } R \to A$ for some domain R (where R means shared randomness – e.g. a string of random numbers which they agree on before the game starts, and they are allowed to use it during the game). The *value* of the game is

$$\omega(G) = \max \sum_{W_i,\ldots,W_k} p_{win} \tag{3.1.2}$$

So $\omega(G)$ is the maximum of the winning probabilities over all strategies of the players.

In comparison to a classical nonlocal game, a classical entangled nonlocal game is just allowing provers to share any quantum state $|\Psi\rangle$. The power is not in the sheer number of quantum states to choose from. The computational power of the provers was unbounded in the classical case already. However, these preshared quantum states bring qualitatively different possibilities for the provers' strategies, relying on quantum correlations. We will see an example of this in the next Section.

3.2 CHSH

We described the CHSH game briefly in 1. Let us now elaborate on it. Consider two distant players. Each of the players holds one half of an entangled pair of qubits (e.g. (2.0.8)). Each of the players receives a single input bit (question) $x, y \in \{0, 1\}$, chosen randomly, from a referee. They are able to perform any operation on their half of the entangled pair – for example, measure it in a basis of their choice – but they are forbidden to communicate. Their goal is to maximize their winning probability by producing outputs a, b that satisfy the CHSH condition

$$a \oplus b = x \wedge y \tag{3.2.1}$$

as often as possible.

We can put this CHSH condition into a table (3.2.2), where each row corresponds to a pair of questions from $\{00, 01, 10, 11\}$ and each column corresponds to the player's combined actions. For example, when they get the combined question 11 (both Alice and Bob get 1 on input, line 4 in the table), then they win iff they respond 01 or 10.

A_0B_0	A_0B_1	A_1B_0	A_1B_1
1	0	0	1
1	0	0	1
1	0	0	1
0	1	1	0

(3.2.2)

Thanks to the uniform distribution of the questions for the players, we can calculate the overall winning strategy of the players by summing the winning probability for each question, expressed with the help of (3.2.1).

$$\begin{aligned} P_{\mathcal{S}} \equiv {} & \frac{1}{4}A_0 \oplus B_0 + \frac{1}{4}A_0 \oplus B_1 \\ & + \frac{1}{4}A_1 \oplus B_0 + \frac{1}{4}(1 - A_1 \oplus B_1) \end{aligned} \tag{3.2.3}$$

There are 16 deterministic classic CHSH strategies. For a given question, a player always answers the same way. For example, if $A_0 = 1$, player A always answers "1" to question "0".

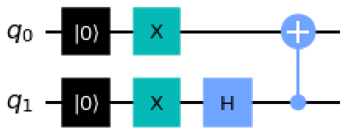


Figure 3.2.1: A quantum circuit to make an entangled pair.

We show them along with the calculated $P_{\mathcal{S}}$ from (3.2.3).

A_0	A_1	B_0	B_1	$4P_{\mathcal{S}}$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	3
0	0	1	1	3
0	1	0	0	1
0	1	0	1	3
0	1	1	0	1
0	1	1	1	3
1	0	0	0	3
1	0	0	1	1
1	0	1	0	3
1	0	1	1	1
1	1	0	0	3
1	1	0	1	3
1	1	1	0	1
1	1	1	1	1

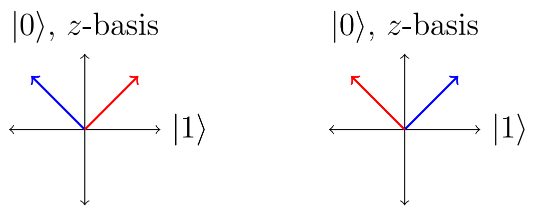
(3.2.4)

The winning probability can be also calculated by

$$p_{win} = \sum_{x,y} \pi(x,y) \sum_{a,b} V(a,b|x,y) p(a,b|x,y). \tag{3.2.5}$$

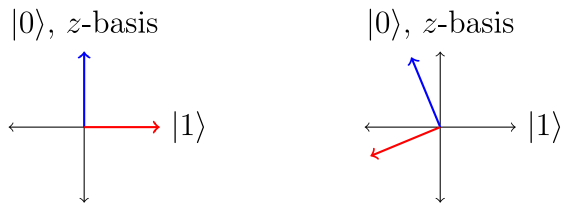
If we go through the table (3.2.4), we quickly find that the maximal winning probability for deterministic classical strategies is 0.75. Probabilistic strategies cannot do better because they are just combinations of deterministic strategies.

However, things are different in the quantum realm. Let Alice and Bob share an EPR entangled pair (2.0.8) made by the circuit in Figure 3.2.1. Alice and Bob then take each one qubit from the EPR pair and make them move far away from each other. Now, have them play the CHSH game (3.2.2). When given a question, Alice is going to measure her qubit in a



(a) Alice's qubit **(b) Bob's qubit**

Figure 3.2.2: Alice and Bob begin with



(a) Alice's qubit **(b) Bob's qubit**

Figure 3.2.3: For $x = 0$, $y = 0$

basis of her choice, corresponding to her strategy. Bob is also going to measure in one of two bases. We will now show, that the probability that they answer correctly can then be written as $\cos^2 \alpha$ for some angle α such that the overall probability exceeds the classical limit of $\frac{3}{4}$.

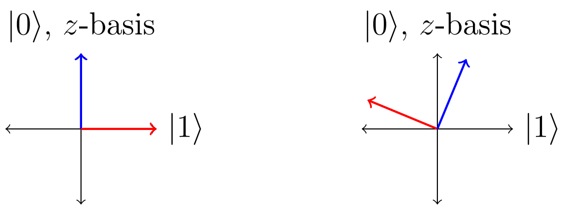
According to rules of the game (3.2.1) their answers should be:

$$a = b \text{ for } x = 0,\, y = 0,$$
$$a = b \text{ for } x = 0,\, y = 1,$$
$$a = b \text{ for } x = 1,\, y = 0,$$
$$a \neq b \text{ for } x = 1,\, y = 1,$$

or in short, such that $a \oplus b = x \wedge y$

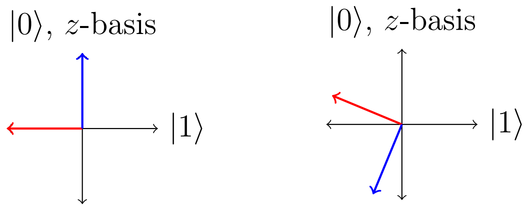
There is a wonderful property of the EPR pair. If Alice and Bob measured their respective halves in the same basis, their results would always be perfectly anticorrellated. As depicted in Figure BLA – sem daj este jeden obrazok, tiez s modrymi a cervenymi, ze co sa deje, ked su obaja v z baze bez rotacii.

The trick they will use now is that they will rotate their qubits according to the questions, in such a way, that the correlations between their measurement results will closely resemble those desired for the CHSH game. They won't be perfect, but good enough to beat the classical limit, with success probability $\cos^2 \pi/8$ in each case.



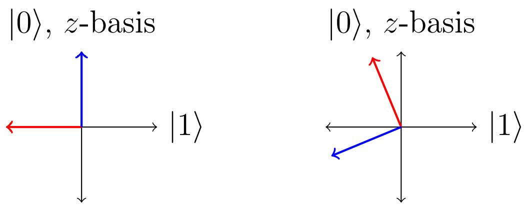
(a) Alice's qubit **(b) Bob's qubit**

Figure 3.2.4: For $x = 0$, $y = 1$



(a) Alice's qubit **(b) Bob's qubit**

Figure 3.2.5: For $x = 1$, $y = 0$



(a) Alice's qubit **(b) Bob's qubit**

Figure 3.2.6: For $x = 1$, $y = 1$

Here's Alice's strategy. For $x = 0$, alice measures. For $x = 1$, she first rotates the qubit clockwise by $\pi/2$. As for Bob, in the $y = 0$ case he will first rotate his qubit clockwise by $\frac{5\pi}{8}$ and measure. In the $y = 1$ case, he will instead rotate his qubit clockwise by $\frac{3\pi}{8}$ and measure afterwards.

Let us see how well this strategy works in each case. Let the players share the entangled state (2.0.8) (see Fig. 3.2.2). We know that if one measures, the others' qubit will collapse to opposite value. They will always measure in the z-basis, because this is easily implementable in our code – simply reading off the amplitude in the computational basis encoding of the state vector. In all Figures we show Alice's states after measurement but Bob's states before measurement.

For $x = 0$ and $y = 0$ (see Figure 3.2.3), let Alice do nothing, if she gets 0 as input question. If she measures, she will get $|0\rangle$ with probability $1/2$ (blue). If she gets 0, he gets 1. Whatever Alice gets, Bob rotates his qubit (which for him unknowingly collapsed to $|1\rangle$) by $\pi/8$ anticlockwise + $\pi/2$ in the same direction. So if Alice got $|0\rangle$, Bob gets the blue vector in Figure 3.2.3. The probability for him to afterwards measure $|0\rangle$ is $\cos^2 \pi/8$. If Alice got $|1\rangle$ (red), Bob will do the same thing but now he began with $|0\rangle$, rotated by $\pi/8$ anticlockwise + $\pi/2$ in the same direction. He will get $|1\rangle$ with $\cos^2 \pi/8$ probability (red). In total they win, $1/4(1/2 \cos^2 \pi/8 + 1/2 \cos^2 \pi/8) = 1/4 \cos^2 \pi/8$. (1/4 because it is one of four combination of questions)

For the $x = 0, y = 1$ case, Alice again does nothing with her qubit. If she measures, she will get $|0\rangle$ with probability $1/2$. If she gets 0, he gets 1. Whatever Alice gets, Bob rotates his qubit from collapsed state $|1\rangle$ by $\pi/8$ clockwise and $\pi/2$ in the opposite direction. So if Alice got $|0\rangle$ (blue), Bob gets the blue vector in Figure 3.2.4. The probability for him to afterwards measure $|0\rangle$ is $\cos^2 \pi/8$. If Alice gets $|1\rangle$ (red), Bob will do the same thing but now he began with $|0\rangle$, rotated by $\pi/8$ clockwise and $\pi/2$ in the opposite direction (red). He will get $|1\rangle$ with $\cos^2 \pi/8$ probability. In total they win, $1/4(1/2 \cos^2 \pi/8 + 1/2 \cos^2 \pi/8) = 1/4 \cos^2 \pi/8$.

The case $x = 1$, $y = 0$. For a simpler interpretation, we can rewrite (2.0.8) to $\frac{1}{\sqrt{2}} (|-\rangle |+\rangle - |+\rangle |-\rangle)$. [JP: toto som opravil, predtym tu bolo $\frac{1}{\sqrt{2}} (|+\rangle |-\rangle - |-\rangle |+\rangle)$, ale ked som si to prepocital, nesedelo to] Alice rotates her qubit by $\pi/4$ anticlockwise. It transforms the global state to $\frac{1}{\sqrt{2}} (|1\rangle |+\rangle - |0\rangle |-\rangle)$.[JP: nvm ci toto rozpisat, preco sa to da tak prepisat, myslim si vsak ze by to rusilo chod textu] If she now measures in the z-basis, it is as if she did not do her $\pi/4$ rotation and measured in the x-basis. Thus, the probability to measure $|0\rangle$ and $|1\rangle$ are now both $\frac{1}{2}$ for her, while Bob's qubit collapses into $|-\rangle$ or $|+\rangle$, depending on Alice's outcome.

Now, Alice measures $|0\rangle$ with $1/2$ probability (see 3.2.4). If she gets 0 (blue), he gets $|-\rangle$. Bob rotates his qubit from collapsed state $|-\rangle$ by $\pi/8$ anticlockwise + $\pi/2$ in the same direction. He measures $|0\rangle$ with $\cos^2 \pi/8$ (blue). If Alice gets $|1\rangle$ after measurement (red), Bob does again the same thing, rotates his qubit from collapsed state $|+\rangle$ by $\pi/8$ anticlockwise $+\ \pi/2$ in the same direction, but now he gets $|1\rangle$ with $\cos^2 \pi/8$ probability (red). In total they win, $1/4(1 \cos^2 \pi/8) = 1/4 \cos^2 \pi/8$.

The case $x = 1, y = 1$ (see 3.2.6). We again begin with $\frac{1}{\sqrt{2}}(|-\rangle |+\rangle - |+\rangle |-\rangle)$. Alice rotates her qubit by $\pi/4$ anticlockwise. Now, Alice measures $|0\rangle$ with probability equals to $1/2$. If she gets 0 (blue), he gets $|-\rangle$. Bob rotates his qubit from collapsed state $|-\rangle$ by $\pi/8$ clockwise and $\pi/2$ in the opposite direction. So if Alice gets $|0\rangle$, Bob gets $|1\rangle$ with $\cos^2 \pi/8$ probability (blue). If Alice gets $|1\rangle$ (red), Bob will do the same thing but now he began with $|+\rangle$, rotated by $\pi/8$ clockwise and $\pi/2$ in the opposite direction. He will get $|0\rangle$ with $\cos^2 \pi/8$ probability (red). In total they win, $1/4(1/2 \cos^2 \pi/8 + 1/2 \cos^2 \pi/8) = 1/4 \cos^2 \pi/8$.

Now we add all the wins, and we get $\cos^2 \pi/8 = 1/2 + 1/2\sqrt{2} \approx 0.8536$ winnning probability for the whole game.

3.3 Complexity

How hard could it be to determine the quantum value of a nonlocal game? Let us explore the complexity of this problem. Recall (3.1) ($V : A^k$ x $Q^k \Rightarrow \{0, 1\}$) and (3.2.2). If we would add any new type of question, then our table would have to be rescaled by a factor 2. The same would happen if we would add a new player. (This is just a simple combinatorics). So, types of nonlocal games scale exponentially with k.

Now recall (3.2.4) where we displayed a table of all strategies for the CHSH game, where $k = 4$. On each question, players can answer either 0, or 1. That makes number of all deterministic strategies grow with exponential factor 2^k making the number of all possible deterministic strategies be equal to 2^{2^k}.

However, in the quantum realm the number of strategies is way beyond this double exponential, because they could share any entangled state, and then perform any local quantum circuit on their respective qubits.

For example, if they share m qubits and if we allow them to use universal set of gates - CNOT, T, S, H. For this set of gates on m qubits for each it is $m(m-1) + 3m$ options in each round of the game. Therefore, for k rounds together there are $(m(m-1) + 3m)^{pk}$ options, where p is the number of players. But we must simulate each circuit in each round, thus, we need the classical calculation of order 2^{pm}. So, the cost of checking all discrete strategies is $2^{pm} (m(m-1) + 3m)^{pk}$.

However, the most difficult is to work with a vector the players share. The size of vector is 2^m, where m is the number of qubits. Further, we know that each number in that vector is complex, this fact also doubles the amount of computation. Moreover, each manipulation with this vector costs at least 2^{m^2} because of the cost of matrix multiplication. The game just as simple as it looks scales enormously.

In fact, one can encode the proofs of NP and even harder problems with these games because we are proving both random things (MA), and quantum things (QMA). (and way beyond this is MIP*). [24]

To sum up, there are lots of strategies and non-local games out there and we can't hope for an exhaustive search and comparison of the classical and quantum approaches. Instead, we want to find at least some interesting games, exhibiting quantum advantage. We will also investigate the possible limits of this advantage. To search for these games and their optimal classical/quantum strategies, we are going to use machine learning, simulated annealing and genetic algorithms.

Chapter 4

Reinforcement learning

We will use reinforcement learning approach to search for the best quantum strategies in nonlocal games. Before we talk about reinforcement learning let us define Markov decission processes as a preliminary for reinforcement learning.

4.1 Markov Decision processes

Since Markov decision processes are built upon Markov chains, we will start with a simpler concept: Markov chains.[23]

A random process is the sequence of random variables S_t. We call these variables states of the system in time t. If the set of all states is finite or countable, then $\{S_t\}_{t=1}^{\infty}$ is called a chain. Then, the chain S_t has the Markov property if

$$P(S_{n+1}|S_n, S_{n-1}, ..., S_1) = P(S_{n+1}|S_n).$$

It means the next step S_{n+1} in the random process depends only on the immediate (previous) state S_n, and not on the whole history of the system. We can say that the process is memoryless. Such a sequence $\{S_t\}_{t=1}^{\infty}$ is called a Markov chain.

We can describe the mathematics behind RL through Markov decision processes (MDP). MDP is powerful to describe decision making where a subset of outcomes are random. Therefore, an agent does not have full control over the remaining outcomes. MDP extends the Markov chain by the addition of actions, therefore allowing choice, and rewards (thus giving motivation). A MDP is a four-tuple (S, A, R_a, P_a), where

- S is the set of all possible states (S_t is the state at epoch/time t)
- A is the set of all possible actions (A_t is the action at epoch/time t)

- $R_a(s, s')$ is the expected immediate reward when state transition occurs from state s to state s' after action a.
- $P_a(s, s') = P(s_{t+1} = s'|s_t = s, a_t = a)$ is the transition protatbility from state s after taking action a to state s'

Now we are ready to talk about reinforcement learning.

Reinforcement learning (RL) is a type of machine learning in which agents learn to perform actions by interacting with an environment and getting a reward from it based on their actions. "The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment." [23]

The environment can either punish the agent for undesirable actions (bad behaviour – negative reward) or reward the agent for desirable actions (positive reward).

For the successful training of agent, there needs to be the right balance between agent's exploration of unknown paths and agent's exploitation of already acquired knowledge from the previous experience.

The training of RL agents consists of agent-environment interactions. The agent gets the information about the state of the environment and performs an action. Agent's actions change the environment and the agent receives a reward or punishment (see Fig. 4.1.1).

Total reward can be defined as

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_{t+n} \tag{4.1.1}$$

The goal of an agent is to maximize the sum of its future rewards G_t. We use a discounted sum of future rewards by hyperparameter γ in reinforcement learning because of unreliability of agent's future predictions. Ultimately, we can not maximize the return because both the policy and environment transitions can be random, making the return also random. Because of this, we maximize only the expected return.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T \check{} t \check{} 1} R_T \tag{4.1.2}$$

To describe how "good" any state s following decision policy π is we use a state-value function of MDP and we call it value function (4.1.3).

$$V^{\pi}(s) = E_{\pi}[G_t|S_t = s] \tag{4.1.3}$$

The expected return given that the agent is in state S_t and performs action A_t at time t with the policy π is given by the action-value function specified in (4.1.4).

$$Q^{\pi}(s,a) = E_{\pi}[G_t|S_t = s, A_t = a] \tag{4.1.4}$$

Then, the relation between Q^{π} and $V^{\pi}(s)$ is simply

$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s,a) \tag{4.1.5}$$

Optimal value function is the function that has the highest value for all states. It is described as (4.1.6).

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{4.1.6}$$

Then, we can also define also optimal action-value function as (4.1.7)

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{4.1.7}$$

Then, the relationship between optimal value function and action-value function is

$$V^{*}(s) = \max_{a} Q^{\pi^{*}(s,a)} \tag{4.1.8}$$

The essence of reinforcement learning is the *Bellman equation* 4.1.9, which says that the maximum future reward is the reward received from s by taking action a + maximum future reward for the next state s' by taking the best action a' discounted by factor γ.

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \tag{4.1.9}$$

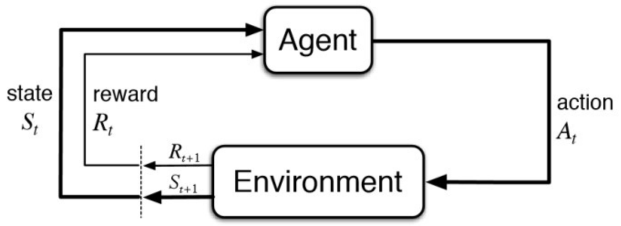


Figure 4.1.1: The basic scheme for reinforcement learning [23].

There are different RL algorithms. We will use Q-learning approach because of the simplicity and effectivity of the method.

4.2 Q-learning

The idea behind *Q-learning* is that we can approximate Q^* using the Bellman equation. [23]

Q-learning equation (4.2.1) iteratively approximates the optimal action-value function, Q^* , by the learned action-value function, Q, independent of the policy being followed (see that it has no π in superscript). This leads to better convergence and that is why Q-learning is off-policy method.

$$\underbrace{Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underset{\text{Learning rate}}{\alpha} \left[\underbrace{R(s,a)}_{\text{Reward}} + \underset{\text{Discount rate}}{\gamma} \overbrace{\max_{a'} Q'(s',a')}^{\substack{\text{Maximum predicted reward, given} \\ \text{new state and all possible actions}}} - Q(s,a)\right] \tag{4.2.1}$$

The pseudocode of algorithm computing Q(s,a) is shown below.

Algorithm 4.2.1 Q-Learning

```
Initialize $Q(s,a)$, for all $s$ and $a$ arbitrarily, except that $Q(\text{terminal},\cdot) = 0$
For each episode:
  For each $s \in S$:
    Select $a$ using $\pi$ derived from $Q$
    Take $a$, observe $r$, $s'$
    Update Q-table according to (4.2.1)
    $s = s'$
```

Chapter 5

Genetic algorithm

Further, we will use Genetic algorithm (GA) to optimalize circuit's gates' parameters. [13] GA is an optimization algorithm inspired by natural selection. Algorithm starts with generated initial population and lets the initial population evolve while utilizing the survival of the fittest. New populations are produced iteratively by selection, crossover and mutation.

Key features of GA are:

- chromosome representation (X) - represents individual qualities (e.g. N qualities $\longrightarrow$ N dimensional vector where each entry corresponds to the quality)
- selection strategy (S) - selects individuals into the next generation (usually the best-fit individuals)
- replacement strategy (R) - replaces individuals with the new individuals (usually the least-fit)
- crossover (C) - combines multiple individuals into new individuals
- mutation (M) - changes individual qualities with some probability (e.g. numbers corresponding to qualities in vector)
- fitness function (F) - evaluates individual according to some criteria

Algorithm can be writen as follows:

Algorithm 5.0.1 Genetic algorithm

```
Initialize population ($P$)
Evaluate population according to ($F$)
while ($episode < MAX$)
  Select individuals for reproduction given ($S$)
  breed new individuals through crossover, ($C$)
  mutate new individuals according to ($M$)
  evaluate fitness of new individuals given ($F$)
  replace individuals with new ones given ($R$)
```

GA iteratively converges to optimal/suboptimal solution. Because of iteratively evolving the whole population it has the ability to output multiple solutions.

Chapter 6

Simulated Annealing

Simulated annealing is a probabilistic algorithm used to approximate the global optimum of given functions. [4]

Algorithm can be writen as follows:

Algorithm 6.0.1 Simulated annealing

```
initialize one individual state to some variable $x$
set starting temperature $t_{start}$
set ending temperature $t_{end}$
while ($episode < MAX$)
  choose random neighbouring state $n$ of $x$
  calculate the $\Delta$ in fitness/energy of $x$ and $n$
  if $n$ has bigger energy
    $x = n$
  else with small probability (e.g. $< e^{\triangle/t}$)
    $x = n$
  calculate new temperature as $t = t_{start}$ .
  $(t_{end}/t_{start})^{i/steps}$
return $x$
```

We will use simulated annealing to search for the optimal last (chosen by RL) action's parameter.

Chapter 7

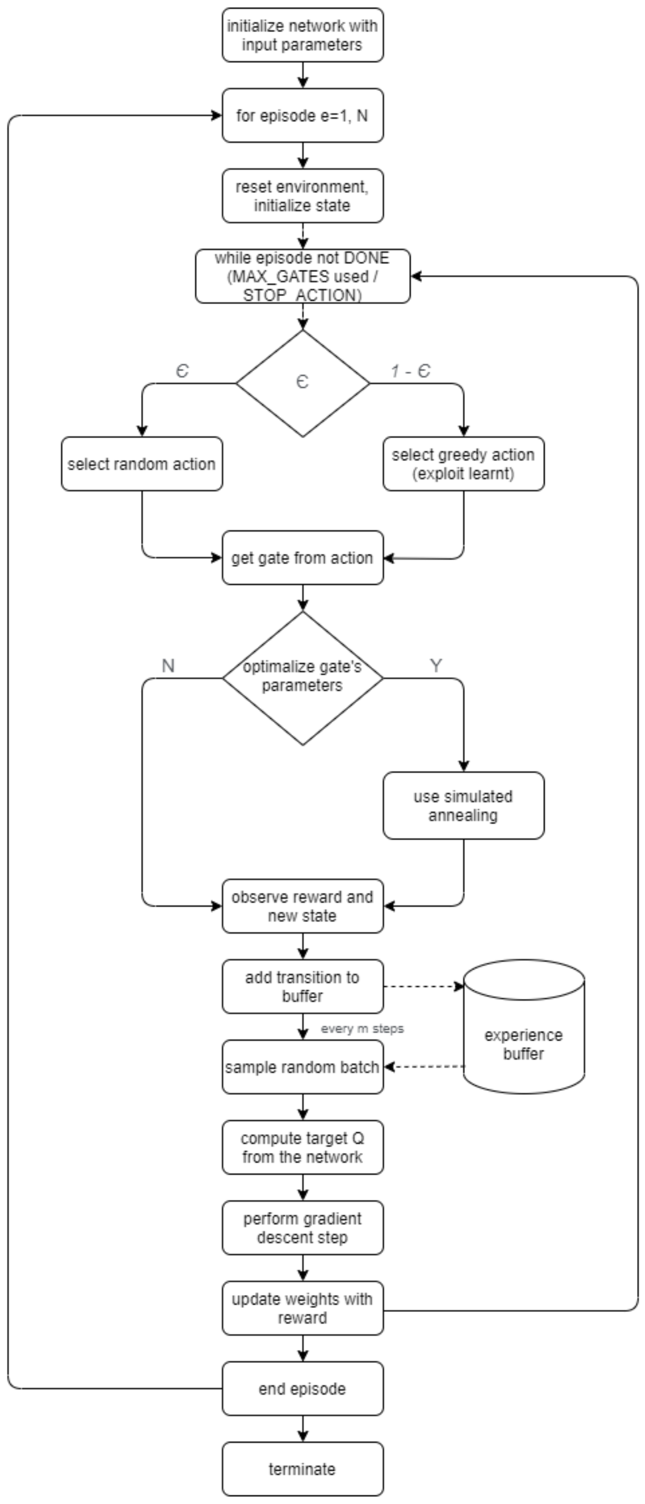
Design and Implementation

We have covered all the theory. Now, let's dive into the design and implementation of the problem (see [20] for code and documentation) using the algorithms we described in previous sections. We use the term winning probability interchangeably with winning accuracy and win rate.

We tried to solve this problem of finding strategies that maximize quantum value using three approaches: 1) by combining reinforcement learning (Deep Q-learning) and simulated annealing, where simulated annealing is used as local search for last step gate's parameters that maximize quantum value (similarly as it is used in [18]). for each player). 2) using reinforcement learning (Deep Q-learning) where we do not use simulated annealing, but we descretize actions. 3) with genetic algorithm to optimalize already chosen gates' parameters (this approach worked very well for games with just one qubit disposal

The first approach is depicted schematically on flowchart in Figure 7.0.1. We start by initializing network along with agent with hyperparameters shown in Figure 7.2.1. Then we let agent learn in episodes. At the start of each episode, we reset environment and initialize state. Then we enter another loop where we iterate until either we have already used maximal number of gates, or agent chose to stop. In each such loop, agent chooses action either randomly, $a_t = random.choice(Actions)$, or by exploiting what it has already learnt by taking argmax over Q, $a_t = argmax_a Q$. Then we decode the action and we get a gate and a "place" (e.g. a0 - means you should place the gate when Alice gets 0 as a question) where should we place the gate. Now we can use simulated annealing to optimalize the chosen gate's parameters to maximize winning probability. Agent observes reward r_t, calculated by (7.0.2), and new state s_{t+1}, calculated as described in Section 7.0.1. Then, we add transition calculations, s_t, a_t, r_t, s_{t+1}, to experience buffer (which serves for learning from past memory). Every m steps we sample random batch (s_x, a_x, r_x, s_{x+1}) and "replay/learn from memory". Further, we calculate target Q from the network, perform gradient descent step and update weights of the neural network with reward r.

Moreover, while building this approach, we had a problem with numerical instabilities which we solved by rounding the states before inputing them to DQN.



initialize network with input parameters
for episode e=1, N
reset environment, initialize state
while episode not DONE (MAX_GATES used / STOP_ACTION)
ϵ
ϵ
1 - ϵ
select random action
select greedy action (exploit learnt)
get gate from action
optimalize gate's parameters
N
Y
use simulated annealing
observe reward and new state
add transition to buffer
experience buffer
every m steps
sample random batch
compute target Q from the network
perform gradient descent step
update weights with reward
end episode
terminate

Figure 7.0.1: Implementation scheme of learning

The second approach follows the same calcuation flow as the first one but does not optimalize gate's parameters. It proved useful for testing phase and for cases, where we have some intuition how could we choose to discretize rotation gates.

In the third approach, we used genetic algorithm on the parameters of already pre-chosen gates. This approach worked very well on the games with just one qubit per player because the players could not do anything special with their one qubit anyways. They would just use rotation gates to maximize quantum advantage. Therefore, we decided to put all the parameters of rotation gates to be optimized. This proved to be much more effective and more precise than the first approach. (see results in Section 8)

Further, we used memoization technique in all approaches to reduce the amount of computation. After each step we check whether we have already used the same ordered array of actions before. If yes, we retrieve from dictionary what we have already calculated. The technique speeds up the learning process drastically in later episodes when the agent is already exploiting what it has learnt.

7.0.1 Nonlocal Environment

We decided to combine nonlocal game with the concept of reinforcement learning (RL) environment. (see 7.0.2)

Thus, we have classical reinforcement learning methods such as reset and step along with the methods needed for nonlocal game such as calculating winning probability from the state implemented in the same class.

There are multiple attributes that we need to fully describe any nonlocal game along with RL environment. We store:

- the number of players
- the number of unique questions (e.g. 0 and 1)
- initial quantum state
- state vector for the episode (e.g. 2.0.8)
- state for RL agent (see Figure 7.0.5)
- state size
- action size
- all possible actions

- dictionary for storing and accessing already calculated (memoization)
- array of previous actions used from the same episode
- winning probability for current step
- other variables for printing and saving

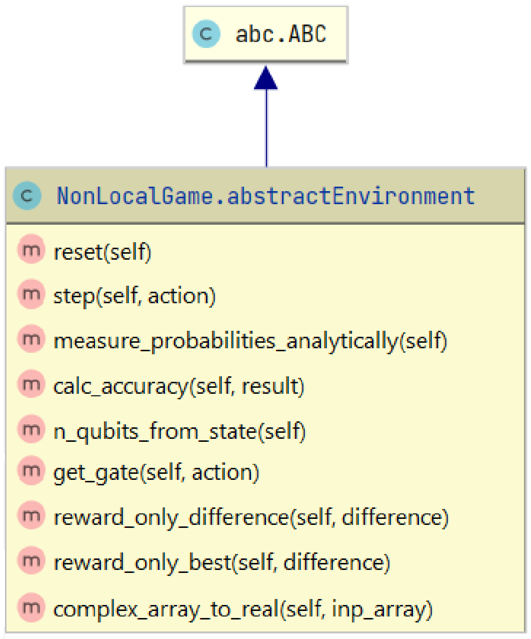
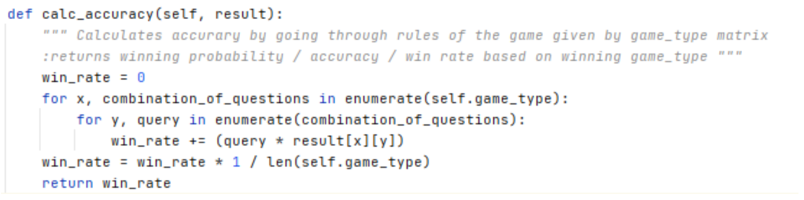

abc.ABC
NonLocalGame.abstractEnvironment
reset(self)
step(self, action)
measure_probabilities_analytically(self)
calc_accuracy(self, result)
n_qubits_from_state(self)
get_gate(self, action)
reward_only_difference(self, difference)
reward_only_best(self, difference)
complex_array_to_real(self, inp_array)

Figure 7.0.2: Implementation scheme of nonlocal game

State representation and step calculation

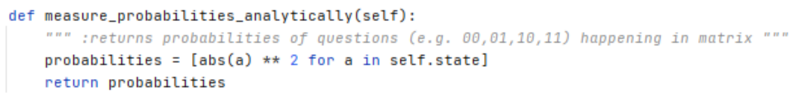
We distinguish between state that players use and the state for the DQN training. State that the players use can be, for instance, (2.0.8). However, the state for the DQN training captures potential states for each combination of questions (see Figure 7.0.5).

Winning probability is calculated as follows:



```
def calc_accuracy(self, result):
    """ Calculates accurary by going through rules of the game given by game_type matrix
    :returns winning probability / accuracy / win rate based on winning game_type """
    win_rate = 0
    for x, combination_of_questions in enumerate(self.game_type):
        for y, query in enumerate(combination_of_questions):
            win_rate += (query * result[x][y])
    win_rate = win_rate * 1 / len(self.game_type)
    return win_rate
```

Figure 7.0.3: Calculating winning probability



```
def measure_probabilities_analytically(self):
    """ :returns probabilities of questions (e.g. 00,01,10,11) happening in matrix """
    probabilities = [abs(a) ** 2 for a in self.state]
    return probabilities
```

Figure 7.0.4: Probabilities of answers

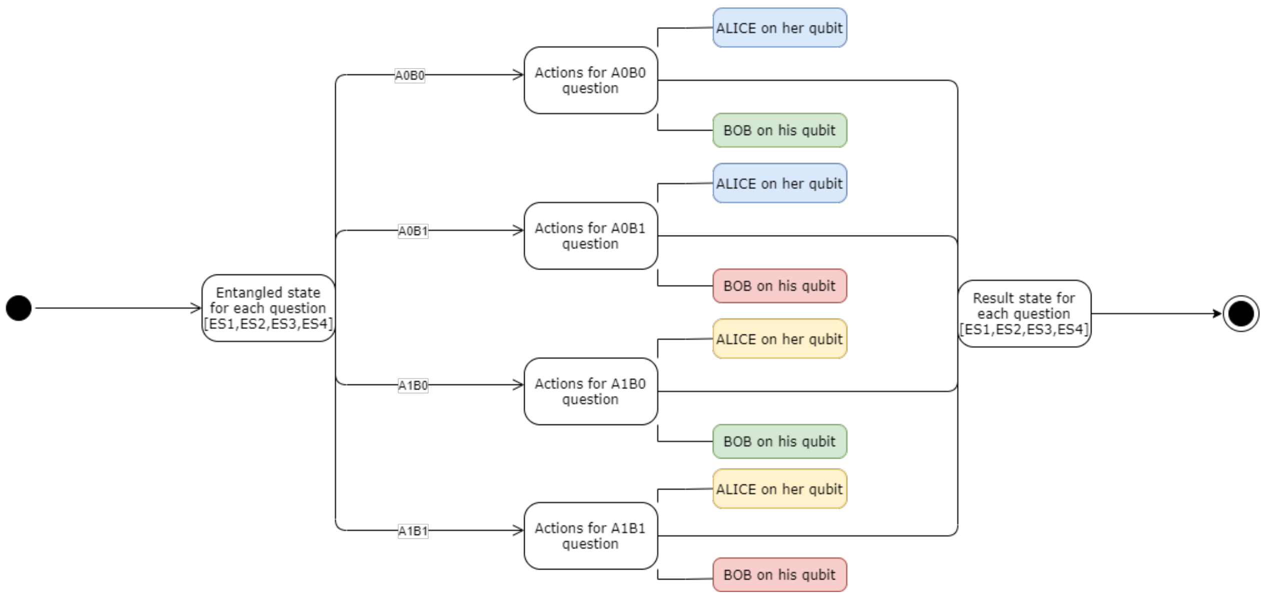

Entangled state for each question [ES1,ES2,ES3,ES4]
A0B0
A0B1
A1B0
A1B1
Actions for A0B0 question
Actions for A0B1 question
Actions for A1B0 question
Actions for A1B1 question
ALICE on her qubit
BOB on his qubit
ALICE on her qubit
BOB on his qubit
ALICE on her qubit
BOB on his qubit
ALICE on her qubit
BOB on his qubit
Result state for each question [ES1,ES2,ES3,ES4]

Figure 7.0.5: Implementation scheme of state for reinforcement learning

Game type is a matrix that captures the rules of the game (e.g. in 3.2.2) and result is the vector of vectors where each internal vector corresponds to one question and each entry in that vector corresponds to Alice's and Bob's answers' probabilities. These probabilities are calculated as shown in Figure 7.0.4.

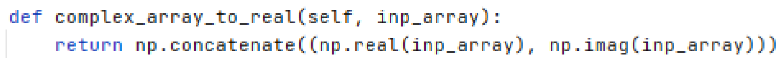
For example, we can calculate probabilities of answers from state (2.0.8) as

$$\begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \longrightarrow \begin{matrix} \text{players answer 00 with } 0^2 \text{ probability} \\ \text{players answer 01 with } (\frac{1}{\sqrt{2}})^2 \text{ probability} \\ \text{players answer 10 with } (-\frac{1}{\sqrt{2}})^2 \text{ probability} \\ \text{players answer 11 with } 0^2 \text{ probability} \end{matrix} \tag{7.0.1}$$

As we can see, each query in vector corresponds to the probability of different combination of players' answers.

Complex states

Machine learning libraries such as Keras, Tensorflow and PyTorch do not support complex arrays as input to neural networks, but we need those to represent quantum states. Therefore, we reduce state - complex array to array of real numbers before feeding it into neural networks by concatenating the real part of an array with the imaginary part. (see 7.0.6)



```
def complex_array_to_real(self, inp_array):
    return np.concatenate((np.real(inp_array), np.imag(inp_array)))
```

Figure 7.0.6: Transforming complex array to array of real numbers

Reward function engineering

Reward function describes how we want our agent to behave. Agent is motivated to gain reward. So, we want to reward the agent for "good" actions.

In our case we want to reward the agent for increasing winning probability. See (7.0.2), where p'_{win} refers to the winning probability on the previous step and p_{win} to current winning probability.

$$reward = p'_{win} - p_{win} \tag{7.0.2}$$

We also tried another functions such as rewarding our agent only if he manages to overcome the best already found winning probability, but they showed big instabilities and undesired effects during learning. Moreover, we also wanted to reward our agent for finding shorter strategies with the same quantum value but this also had undesired effects on the learning performance.

7.1 Reinforcement learning agent

Agents in reinforcement learning are usually trying to achieve some terminal state. However, there is no finish line in these games as we do not know what is the optimal quantum strategy before trying it. Therefore, we limited the number of gates Bob and Alice can use in the circuit to 10/15 due to performance.

Agent in reinforcement learning can learn to take actions through a multiple of paradigms and algorithms. In this paper, we implement deep Q-network (DQN) agent.

Machine learning

We already described algorithm that we use in Section 4.

However, we do not know $Q(s, a)$ and major disadvantage of calculating Q-table is efficiency. In big state spaces and big action spaces, $|S|$ and $|A|$, it gets computationally efficient to store Q-table and calculate expected value because we would have to sum over all transitions $p(s', r|s, a)$.

Despite computational efficiency, Q-function can be approximated by neural networks. We can realize this from the fact that the values in the Q-table only have relative importance. They have importance with respect to the other values. Thus, in come neural networks, universal function approximators. [12]

We do not know what are the correct answers beforehand. What we have is reward in each step and this only gives us ability to train. [23]

To utilize the reward we use means squared error cost function. It is the appropriate choice of calculating difference between expectation and result. It basically measures the average of the squares of the errors and that means that it measures the average squared difference between the estimated value and actual value that we got from the network. Since means squared error is usually used in supervised learning, we can restate it with the help of reward for RL in this way:

$$J = (r + \gamma \max Q'(s', a') - Q(s, a))^2 \tag{7.1.1}$$

Further, gradient descent is complicated in this case, so we would just use automatic differentiation built into most of the machine learning libraries and not mention it here.

The difference between Deep-Q learning and Q-learning can be seen in Figure 7.1.1. Instead of some data structure storing all the Q(s,a), any state now serves as an input to deep neural network (DNN) and we want DNN learn to take the best (output) actions for incoming states.

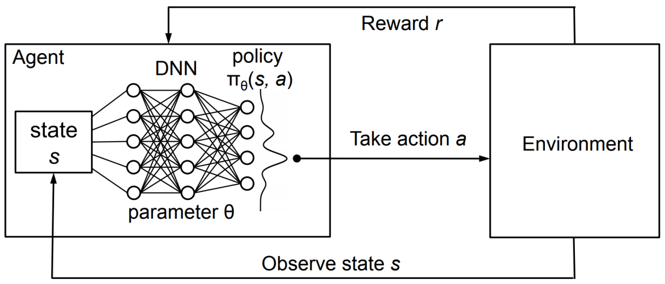


Figure 7.1.1: Deep-Q learning scheme [21]

There are multiple parameters that need to be set before we run such a Deep Q-Learning. These parameters are called *hyperparameters.*

- Gamma, γ - is the discount factor, it determines the importance of future rewards. (see 4)

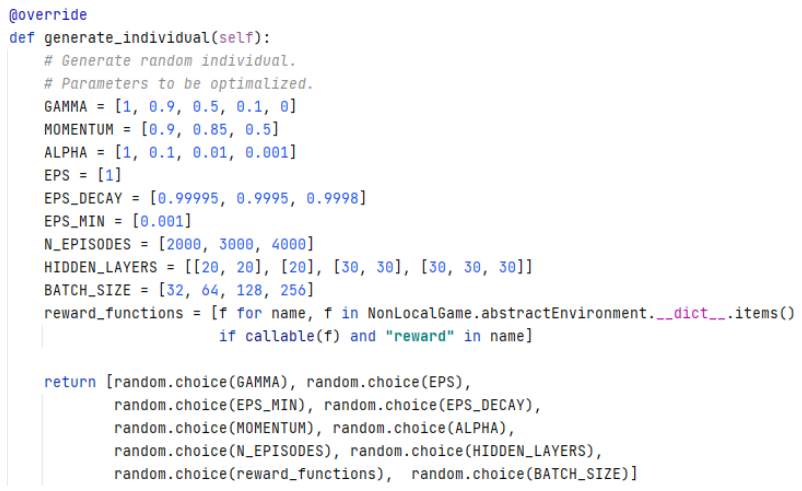
- Momentum - parameter that helps to prevent learning oscillations
- Learning rate, α - defines how much a network updates weights at each step (see (4.1.9))
- Epsilon, ϵ - is a parameter related to epsilon-greedy action selection. If set to 0, the agent would never explore another action, but it would always just expoit what it has already learnt.
- Epsilon decay - is a parameter that sets how quickly should epsilon decrease at each step.
- Number of epochs - is the number of times the whole training repeats (see Fig. 7.0.1)
- Batch size - is the number of samples given to the network before updating NN
- Activation function - is a function that describes when should a neuron in network be activated and is used to bring nonlinearity to NN, which allows to learn nonlinear predictions. We used ReLU activation function in the inner layers (which outputs input for positive input and 0 for negative input) and linear activation function at the output layer.
- Reward function - see (7.0.2)
- Number of Hidden Layers and neurons on each layer - Hidden layers are the layers between input layer and output layer while neurons are units of which layers are composed of.

We use Adam optimalization algorithm for stochastic gradient descent for training deep learning models because of its effectivity. (see [14] for more information about this algorithm)

7.2 Hyperparameters optimalization

We optimized not only the parameters of gates used on the entangled state, but also the hyperparameters of the reinforcement learning itself.

In our case, we were optimalizing the parameters shown in Figure 7.2.1, while we allowed only parameters, GAMMA, MOMENTUM, ALPHA , EPS, EPS decay, EPS minimal value, to mutate. This hyperparameter optimalization with genetic algorithm has improved the learning curve.



```
@override
def generate_individual(self):
    # Generate random individual.
    # Parameters to be optimalized.
    GAMMA = [1, 0.9, 0.5, 0.1, 0]
    MOMENTUM = [0.9, 0.85, 0.5]
    ALPHA = [1, 0.1, 0.01, 0.001]
    EPS = [1]
    EPS_DECAY = [0.99995, 0.9995, 0.9998]
    EPS_MIN = [0.001]
    N_EPISODES = [2000, 3000, 4000]
    HIDDEN_LAYERS = [[20, 20], [20], [30, 30], [30, 30, 30]]
    BATCH_SIZE = [32, 64, 128, 256]
    reward_functions = [f for name, f in NonLocalGame.abstractEnvironment.__dict__.items()
                        if callable(f) and "reward" in name]

    return [random.choice(GAMMA), random.choice(EPS),
            random.choice(EPS_MIN), random.choice(EPS_DECAY),
            random.choice(MOMENTUM), random.choice(ALPHA),
            random.choice(N_EPISODES), random.choice(HIDDEN_LAYERS),
            random.choice(reward_functions),  random.choice(BATCH_SIZE)]
```

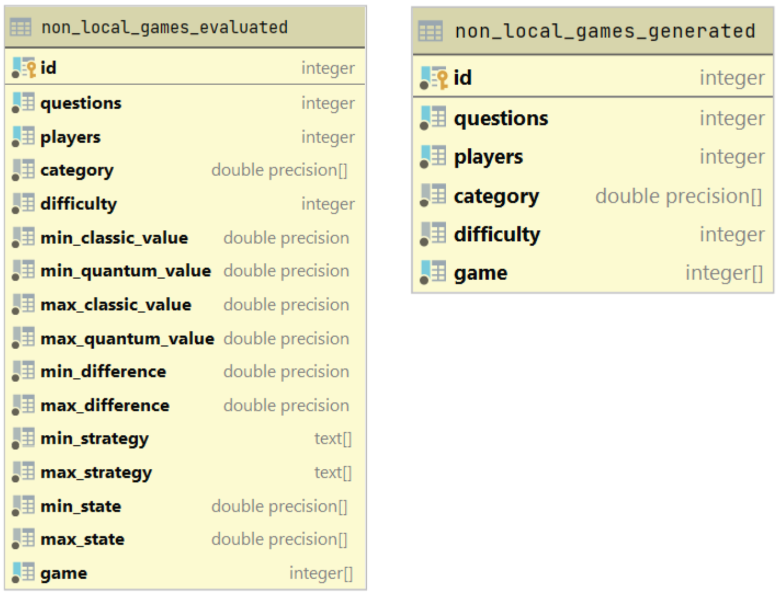
Figure 7.2.1: Individual for hyperparameters optimalization

7.3 Database for storing best results

We implemented local database in PostgreSQL using psycopg2 python library where we store already generated nonlocal games. But before putting them into the database we check whether they are not symmetrical copy of some already stored game. We can spot symmetries as described in 3.1.

For each already searched games we store important informations from the search such as the best found quantum and classical value, the best strategy leading to the best found values. See what we store in Figure 7.3.1. Minimum difference is the difference between mininum quantum value and minimum classical value. Maximum difference is the difference between maximum quantum value and maximum classical value. The difficulty of game is calculated as a sum of 1s it has in matrix corresponding to game (e.g. see 3.2.2 - has difficulty 8). The category of the mentioned game is tuple of (classical maximum value, classical minimum value).

Local database allows us to effectively compare classical value with quantum value and sort games that show the biggest quantum advantage. We also do not have generate all possible games each time we run our program.



non_local_games_evaluated	
id	integer
questions	integer
players	integer
category	double precision[]
difficulty	integer
min_classic_value	double precision
min_quantum_value	double precision
max_classic_value	double precision
max_quantum_value	double precision
min_difference	double precision
max_difference	double precision
min_strategy	text[]
max_strategy	text[]
min_state	double precision[]
max_state	double precision[]
game	integer[]

non_local_games_generated	
id	integer
questions	integer
players	integer
category	double precision[]
difficulty	integer
game	integer[]

Figure 7.3.1: Database for storing nonlocal games

Chapter 8

Results

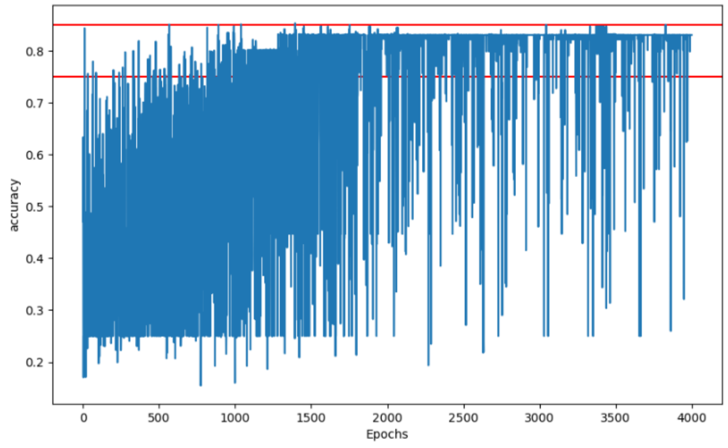


Figure 8.0.1: DQN with simulated annealing - winning probability

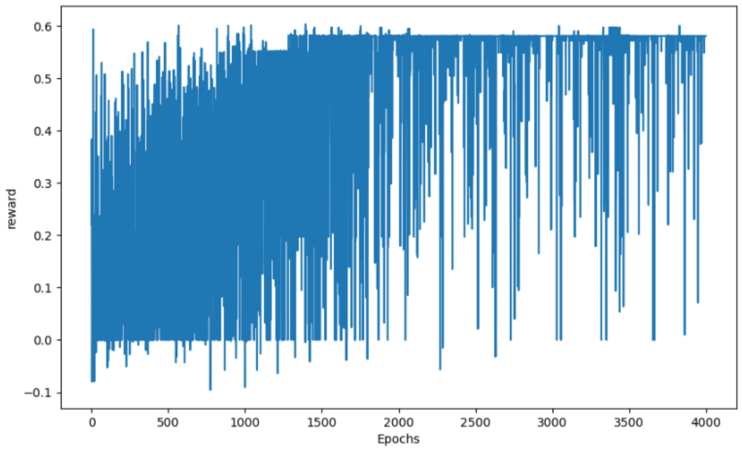


Figure 8.0.2: DQN with simulated annealing - reward

With the first approach, DQN with simulated annealing, the agent was able to converge to suboptimal winning probability. In Fig 8.0.1 is the evolution of winning probability through the 6000 epochs and evolution of agent's learning with its reward in Fig 8.0.2.

In the second approach we used the DQN without simmulated annealing. In Figure 8.0.4 is the evolution of reward our agent gets through learning in 6000 epochs on CHSH game (3.2). We used discretized rotation gates into k - $\pi/16$ angles. Agent was able to converge to suboptimal value as shown in 8.0.3 by continually maximizing the expected return (see 8.0.4).

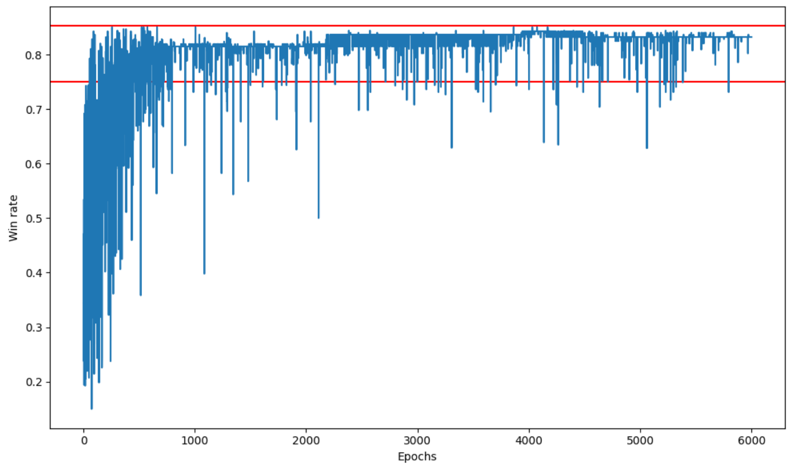


Figure 8.0.3: DQN - winning probability

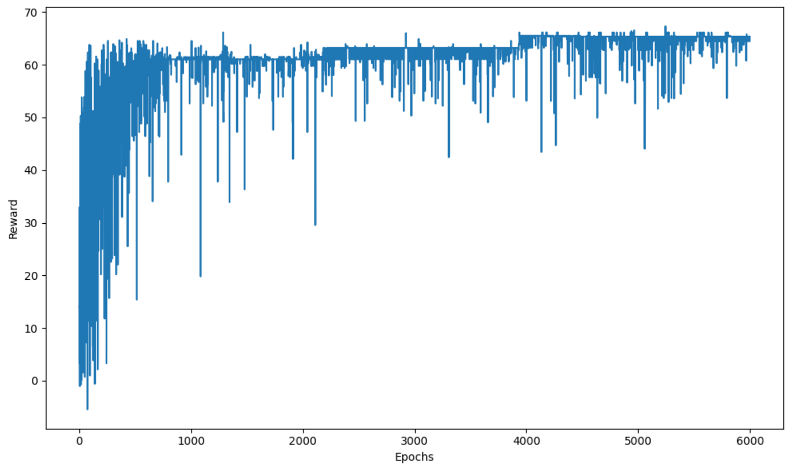


Figure 8.0.4: DQN - reward

In the third approach we searched with genetic algorithm through all (except symmetrical copies) 2-player, 2-question nonlocal games, where the players were allowed to share only one

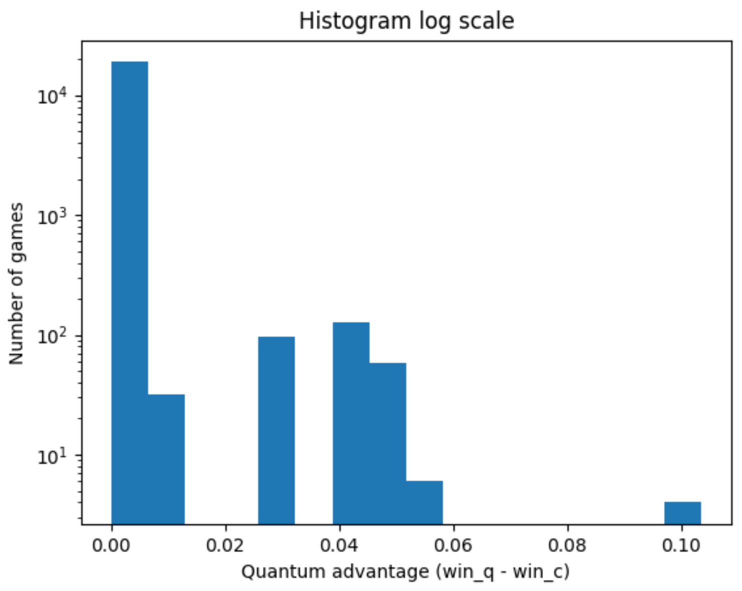


Figure 8.0.5: Searched 2-player, 2-question games with one entangled pair

EPR pair. We did not find any game with bigger quantum advantage than the one in CHSH game (≈ 0.1035) but we find many games where quantum advantage is half of the quantum advantage of CHSH game (≈ 0.05177).

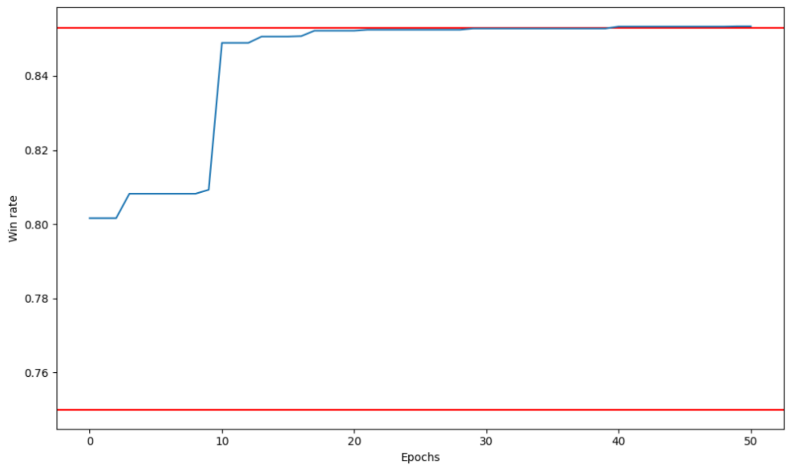


Figure 8.0.6: Genetic algorithm - winning probability

For example, game with these rules:

$$\begin{array}{cccc} A_0B_0 & A_0B_1 & A_1B_0 & A_1B_1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{array} \tag{8.0.1}$$

has a quantum advantage ≈ 0.05177. In classical world, this can be won 0.75 times, while in quantum ≈ 0.80177. The players start with (2.0.8). Alice rotates ≈ 73.5 degrees around y axis, when she receives question 0. If she receives 1, she rotates by ≈ 16.39 degrees. Bob rotates by ≈ -28.42 degrees, when he receives 0. If he receives 1, he rotates by ≈ -118.52 degrees. See [20] for all these 2-player, 2-question games. For any 2-player, 2-question games, the algorithm (see Figure 8.0.6) was able to converge in 20 epochs to maximal quantum value by optimalzing pre-chosen gates for all players. (universal rotation gates)

The results in Fig. 8.0.5 show us the frequency and distribution of 2-player, 2-question games where players share one entangled pair. Quantum advantage is calculated as winning probability of quantum players subtracted by winning probability of classical players. Games that show 0.10 quantum advantage are all CHSH-like. They have just swapped rows in game matrix.

Chapter 9

Discussion and Conclusion

The purpose of this work was to explore ways of approximating the optimal winning probability of nonlocal games using reinforcement learning and genetic algorithm. We set our goals to be able to approximate winning probability of all sorts of nonlocal games. Therefore, we designed and implemented a system for playing nonlocal games with different approaches. We implemented a way to effectively generate nonlocal games with their classical strategies. While we set goals high, we were unable to effectively search for quantum strategies for bigger games because of the computational complexity and huge state vector representation scaling.

However, we searched all 2-player, 2-question games where players share one entangled pair and found multiple games with no simple interpretation but showing a quantum advantage of ≈ 0.05177 (see 8) against classical strategies. We have experimentally proved that the CHSH inequality proposed by [6] is the highest possible violation for 2-player, 2-question games where players share one entanglement pair.

If we want a bigger advantage, we need bigger games. But surprisingly, when we do a combined two or more paralel CHSH games, classically the success is better than $3/4^2$, but quantum it is still q^2 with $q \approx 0.8535$ as proposed in [26].

Further, we have tried another approach for playing, generating and representing nonlocal games. We found that representing and manipulating with the whole state vector while also using it as a part of the state in RL is computationally efficient.

We find that genetic algorithm is an effective approach for investigating games when players have only one qubit at their disposal, but in the currently implemented form it would not be able to search properly for bigger games, because it was implemented only to optimalize gates' parameters, not to choose gates' themselves. DQN with simulated annealing approach takes a while to converge but shows promising results. It can be used for bigger games because it has the ability to choose gates' while also optimalizing gates' parameters, when combined with simulated annealing.

For further elaboration, we would restate the problem in a more algortihmical way in order for it to be more implementantion-friendly.

Moreover, we would look for more effective state-of-art (e.g. DDQN, A3C, PPO) algorithms to search for the best strategies and compare their effectiveness in these kinds of problems. (See how PPO could be used for these games in [5]) Another approach would be to use genetic algorithm to not only optimalize parameters, but also to construct circuit itself (see [19]).

[JP: na popisanie zlozitosti Q-learningu a Deep-Q learningu som ani poriadne nevedel najst literaturu, preto to tu pisem tak jednoducho, ze do akej triedy patria, ostatne by tak ci tak boli mensie funckie ako simulacia kvantoveho obvodu, preto to mozem dat do]

Regarding complexity of used methods, in all algorithms we needed to simulate a quantum circuit on a classical computer. There is no known way to efficiently simulate a quantum computational model with a classical computer (This belief is formalized as $\mathsf{BPP} \subsetneq \mathsf{BQP}$). This means that a classical computer cannot simulate a quantum computational model in polynomial time. However, a quantum circuit of $S(n)$ qubits with $T(n)$ quantum gates can be simulated by a classical circuit with $O(2^{S(n)}T(n)^3)$ classical gates. [7] This is because we need to simulate a state vector describing the whole system (tensor product - see Section 2) consisting of individual qubits. The result of tensor product of $S(n)$ qubits is a complex vector of $2^{S(n)}$ entries where each entry corresponds to the amplitude. To obtain an upper bound for the number of classical gates required to simulate a quantum circuit we need a sufficient upper bound for the amount data used to specify the information about each of the $2^{S(n)}$ amplitudes. To do this $O(T(n)$ bits of precision are sufficient for encoding each amplitude. So it takes $O(2^{S(n)}T(n))$ classical bits to account for the state vector of the $S(n)$ qubit system. Next the application of the $T(n)$ quantum gates on $2^{S(n)}$ amplitudes must be accounted for. Quantum gates acting on $S(n)$ qubits' system can be represented as $2^{S(n)}$ x $2^{S(n)}$ matrices. Every time the state vector is multiplied by a quantum gate, $O(2^{S(n)})$ basic arithmetic operations must be performed (calculating how a matrix affects each entry). Thus, there are $O(2^{S(n)}T(n)^2)$ bit operations for every quantum gate applied to the state vector and there are $T(n)$ quantum gates we want to simulate. Therefore, we need $O(2^{S(n)}T(n)^3)$ classical gates to simulate a single quantum circuit.

Furthermore, the genetic algorithm time complexity belongs to $O(O(Fitness)*(O(mutation)+O(crossover)))$. In our case, for calculating the fitness function we needed to simulate the quantum circuit with gates' parameters in order to find the winning probability. But we needed to repeat this calculation of of fitness on each generated individual and for several generations. Moreover, it scales similarly in Deep Q-learning, where we also need to simulate multiple quantum circuits at each episode because we are always trying to optimalize the last action using simulated annealing.

Further, we would consider different state representation since storing and manipulating with the quantum state in state vector representation gets computationally efficient when dealing with bigger states and we have to store one such vector for all combinations of questions.

We would do a different abstraction over the problem because mixing nonlocal game with reinforcement learning environment proved counterproductive. We would also represent quantum states and operations on these gates as a lone-standing classes.

To summarize, we contributed to this topic by showing that

- there are no other 2-player, 2-question games with probability ≈ 0.8535 or higher.
- there are many 2-player, 2-question games that show ≈ 0.05177 quantum advantage
- genetic algorithm is an effective approach for investigating these games when players have only one qubit at their disposal
- reinforcement learning (DQN) can also be used to search strategies for these games
- we need to find another approach how to operate with the state vector because it scales drastically and slows learning process dramatically

We also made an implementation abstraction over nonlocal games in Python that can be built upon with another approaches in future expansions.

[JP: nevadi, ze niektore obrazky mam odkaz na wikipediu?]

Bibliography

[1] S. Adler. Where is quantum theory headed? *Journal of Physics: Conference Series*, 504, 01 2014.

[2] A. Aspect. Proposed experiment to test the nonseparability of quantum mechanics. *Physical review D*, 14(8):1944, 1976.

[3] J. S. Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[4] D. Bertsimas, J. Tsitsiklis, et al. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[5] K. Bharti, T. Haug, V. Vedral, and L.-C. Kwek. How to teach ai to play bell non-local games: Reinforcement learning. *arXiv preprint arXiv:1912.10783*, 2019.

[6] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt. Proposed experiment to test local hidden-variable theories. *Physical review letters*, 23(15):880, 1969.

[7] R. Cleve. An introduction to quantum complexity theory. *Collected Papers on Quantum Computation and Quantum Information Theory*, pages 103–127, 2000.

[8] C. Davisson and L. H. Germer. The scattering of electrons by a single crystal of nickel. *Nature*, 119(2998):558–560, 1927.

[9] A. Einstein. The photoelectric effect. *Ann. Phys*, 17(132):4, 1905.

[10] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

[11] W. Gerlach and O. Stern. Der experimentelle nachweis der richtungsquantelung im magnetfeld. *Zeit. fur Physik*, 9:349–352, 1922.

[12] M. H. Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

[13] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, pages 1–36, 2020.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[15] U. J. Le Verrier. Theorie du mouvement de mercure. In *Annales de l'Observatoire de Paris*, volume 5, 1859.

[16] M. Masi. *Quantum physics: an overview of a weird world: a primer on the conceptual foundations of quantum physics*. Independently published, 2019.

[17] A. Matuschak and M. A. Nielsen. Quantum country. `https://quantum.country/qm`, 2020.

[18] A. A. Melnikov, P. Sekatski, and N. Sangouard. Setting up experimental bell tests with reinforcement learning. *Phys. Rev. Lett.*, 125:160401, Oct 2020.

[19] R. Nichols, L. Mineh, J. Rubio, J. C. Matthews, and P. A. Knott. Designing quantum experiments with a genetic algorithm. *Quantum Science and Technology*, 4(4):045012, 2019.

[20] J. Pastorek. Machine learning, simulated annealing and genetic algorithms for nonlocal games. `https://github.com/janpastorek/Bachelor-Thesis`, 2021.

[21] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth. Deep reinforcement learning enabled self-learning control for energy efficient driving. *Transportation Research Part C: Emerging Technologies*, 99:67–81, 2019.

[22] Rxtreme. English: common quantum logic gates by name, circuit form(s) and matrices. `https://commons.wikimedia.org/wiki/File:Quantum_Logic_Gates.png`, Dec. 2019.

[23] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[24] T. Vidick. *The complexity of entangled games*. PhD thesis, UC Berkeley, 2011.

[25] Wikipedia, the free encyclopedia. Two-slit experiment light. `https://commons.wikimedia.org/wiki/File:Two-Slit_Experiment_Light.svg`, 2021. [Online; accessed January 1, 2021].

[26] H. Yuen. A parallel repetition theorem for all entangled games. *arXiv preprint arXiv:1604.04340*, 2016.