Universiteit Leiden

Master Computer Science

Quantum Checkers: The development and analysis of a quantum combinatorial game

Name:	Luuk van den Nouweland
Student ID:	s2175355
Date:	July 4, 2024
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. E.P.L. van Nieuwenburg
2nd supervisors:	Dr. M. Preuss
	M.F.T. Müller-Brockhausen MSc

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

This thesis focuses on the development and analysis of a quantum combinatorial game: quantum checkers. Multiple concepts from quantum computing are implemented into the game of checkers. More specifically: superpositions, entanglement and measurement. We implemented the previously mentioned concepts successfully by translating each checker piece to a qubit. From this, it is possible to derive rules as to how these concepts will work in the game. Subsequently, an analysis will be done on checkers with different levels of quantumness to see if new strategies emerge. Next, we developed an AI to analyse the game. This analysis showed that new strategies emerged when quantumness is introduced.

Contents

1 Introduction

Quantum computing is a relatively new research field. Unlike classical computers, which use bits to process information, quantum computers use quantum bits or qubits, which can exist in multiple states simultaneously. These properties allow quantum computers to solve some problems significantly faster than classical computers. However, public knowledge about the properties of quantum computers is lacking. This gap in understanding can largely be attributed to the fact that quantum phenomena are inherently counter-intuitive and are not experienced in the daily lives of the average person.

Games could be used to introduce and explain these quantum concepts in an accessible manner, since they provide the perfect playground to get familiar with the concepts. Much research is done into using games as educational tools [13]. This thesis however, will not focus on the effectiveness of games as educational tools.

Various different quantum games have already been created, like quantum chess [4, 2], quantum minesweeper [9], quantum prisoners dilemma [5], quantum tic-tac-toe (or tiq taq toe) [8, 19] to give a few examples. This thesis focuses on the development of quantum checkers.

The development of quantum checkers will focus on implementing the concepts of quantum computing in a way that is easy to grasp. The game also allows us to analyse how the game of checkers changes when introducing the principles of quantum computing.

This thesis aims to design and implement a functional version of quantum checkers, with a focus on an accessible design. We also implement an AI for quantum checkers, which subsequently is used to analyse the game by measuring and comparing the performance of different types of agents playing it. Using this the differences between classical checkers and quantum checkers will be analysed. Two main question will be answered in this thesis:

- How to quantize the game of checkers?
- What new strategies emerge from the quantum version of the game compared to the classical version?

1.1 Thesis structure

This thesis is organised into several chapters. Section 2 will introduce some fundamentals of quantum computing necessary to understand this thesis. Section 3 discusses background and related work. Section 3.1 discusses the implementation of Monte Carlo Tree Search for the game of quantum checkers. Section 4 discusses the design and implementation of the concepts of quantum computing into the rules of classical checkers. Section 5 discusses the experimental setup and the results of the experiments. Section 6 discusses the results of the experiments. Section 7 concludes the thesis.

2 Fundamentals of quantum computing

To understand concepts introduced in this thesis a short introduction to quantum computing will be given. For more in-depth explanations, see Quantum Computation and Quantum Information by Nielsen and Chuang [14].

2.1 Quantum bits

Similarly to bits in classical computers quantum computers have quantum bits, or in short qubits. A qubit can either be in a state of $|0\rangle$ or $|1\rangle$.

2.2 Superpositions

The difference between classical bits and qubits is that qubits can exist in superposition. Superpositions allow qubits to be in a combination of both 0 and 1 simultaneously, rather than being confined to a single binary state. This is formulated as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

2.3 Measurement

When measuring a qubit in superposition it will collapse back to a classical state. You will observe either state $|0\rangle$ or $|1\rangle$ with probability $|\alpha|^2$ or $|\beta|^2$ where to probability of $|\alpha|^2 + |\beta|^2 = 1$. An example of a equal probability between two qubits would be

$$|\psi\rangle = \tfrac{1}{\sqrt{2}}\,|0\rangle + \tfrac{1}{\sqrt{2}}\,|1\rangle$$

2.4 Entanglement

Entanglement is a superposition of multiple qubits where the state of one qubit directly influences the state of another qubit. When one of the qubits is measured, the state of the other qubit instantly collapses.

3 Related Work

Dorbec and Mhalla propose a standardised framework for introducing quantum-inspired moves in combinatorial games [6]. This framework proposes several different rule-sets for developing quantum games. Burke et al. do further research into quantum combinatorial games and the previously mentioned ruleset, which are called quantum flavors in their research [3]. Their research is primarily focused on quantum flavor D, which is in their opinion the most natural expansion of quantum combinatorial games. Citing Dorbec and Mhalla ruleset D states that "unsuperposed moves are always allowed (seen as the superposition of two identical moves)." [6]. In more accessible terms this means that a player can always choose between executing classical moves or quantum moves. This ruleset will also be used when developing quantum checkers.

Quantum Chess [4] is a version of chess in which players are able to use quantum moves to their advantage. The game includes, according to them, the trifecta of quantum phenomena: superposition, entanglement, and interference. Chess pieces are able to become superposed and entangled and executing quantum moves can lead to destructive and constructive interference, which players are able to use to their advantage. In their research the "no double occupancy" rule is proposed, which will also be used in quantum checkers. This rule will be explained in Section 4.3.

In another implementation of quantum chess [2, 1], all pieces exist in a quantum superposition of two piece-type states. A piece collapses to a classical known state when touched. The initial version of the game only has superpositions, but a second version is proposed where each piece is initially entangled with another piece.

Both quantum minesweeper [9] and quantum tic-tac-toe [8] have been created as educational tools to teach the concepts of quantum mechanics in more accessible manner. In quantum minesweeper there are several classical boards in superposition. The goal is to figure out the layout of all the mines in all the superposed classical boards. In quantum tic tac toe the player has to place two marks on two different squares. These marks exist in superposition. When measurement happens, the marks collapse back to a classical state. Another version of quantum tic tac toe named Quantum TiqTaqToe, created by Evert van Nieuwenburg, can be found at [19]. This version introduces the concepts of superpositions and entanglement. A superposition happens when a player marks two different squares, similarly to the other version of quantum tiq tac toe. Entanglement happens when the opponent occupies a square which is already occupied by the opponent. Measurement happens when all squares have been occupied.

All these games share a common goal of teaching the concepts of quantum mechanics in an engaging and accessible manner. Quantum games also serve as a playground for the development of Artificial Intelligence for quantum systems.

This thesis will not focus on the effectiveness of games as educational tools, but engagement and accessibility will still be important design goals.

3.1 Monte Carlo Tree Search

Monte Carlo Tree search [12] (MCTS) is a multi-armed bandit search algorithm that is used for decision making, which works especially well in board games. It has already been implemented in many combinatorial games like Chess, Poker, Settlers of Catan, Othello [17]. More recently MCTS has been combined with deep reinforcement learning to achieve expert level play in the game of Go [16].

The success of MCTS in these diverse games lies in its ability to construct and navigate a game tree, where each node represents a state of the game. In the case of quantum checkers, this means that each node is a possible board state. Each child node is a board state reachable from the parent node by doing a legal move. Each node contains information about the number of times this node has been visited and the number of games won from the node. MCTS uses this information together with the following four steps called selection, expansions, simulation and propagation to select which move to do. For an schematic overview of the algorithm, see Figure 1.

The big difference between games like Chess and Go in comparison to quantum checkers is the former are deterministic, while the latter is stochastic. The stochastic nature of the game will introduce chance nodes into the game tree, which will impact the performance of MCTS.

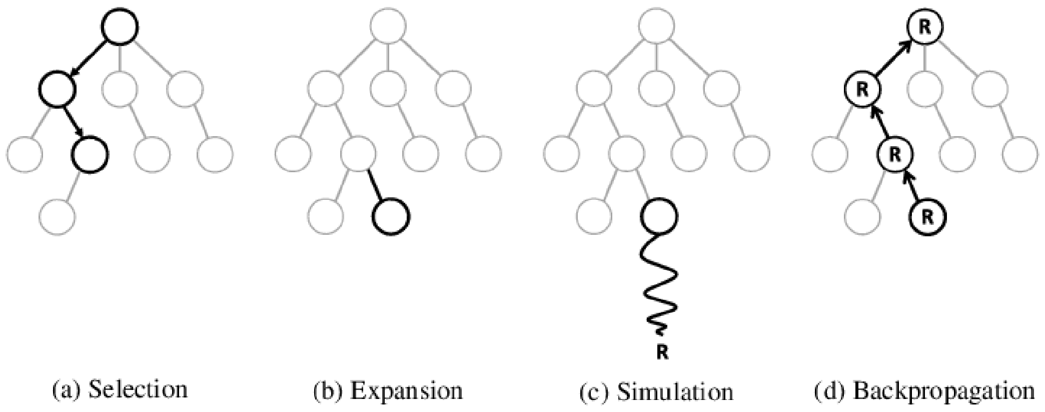

Figure 1: An schematic overview of the four steps of the MCTS algorithm [11]. These steps are repeated to build the MCTS tree.

3.1.1 Selection

First, a node needs to be selected that is going to be expanded. For this the Upper Confidence Bound (UCB) algorithm is used. Starting from the root node, continuously select child nodes using UCB until a leaf node is reached. A leaf node is a node that has not yet been fully expanded, or where the node is in a terminal state; i.e. the game is finished.

$$k_i \cdot (\frac{w_i}{n_i} + c \cdot \sqrt{\frac{\ln N}{n_i}}) \tag{1}$$

- w_i: the number of times a simulation from this node has resulted in a win.
- n_i: the total number of simulations run from this node.
- N: the total number of simulations from the parent node.
- c: constant exploration parameter.
- k_i: weight of the current node.

This formula consists of two parts that balance exploration and exploitation.

The first part of the formula, the exploitation term $\frac{w_i}{n_i}$, increases the more simulations are won from this node.

The second part of the formula, the exploration term $c\sqrt{\frac{\ln N}{n_i}}$, increases when the total number of simulations N grow and decreases when more simulations have been run from this node. c is the exploration parameter. A higher value means more exploration will happen.

Because quantum checkers is stochastic, a weight k_i is added. This weight is an approximation of the probability this game state will be reached from a specific node.

3.1.2 Expansion

Once a node has been selected and it is not in a terminal state, a child node will be added. A random move is chosen from all possible legal moves that can be executed from this node, which have not already been chosen. E.g. moving a piece on a diagonal. All possible states that can result from this move will be added to the node. This means that if this move causes a measurement to happen, multiple nodes will be added.

3.1.3 Simulation

For each node that has been added during the expansion phase, a simulation will be run. A simulation is a random play-out of the game from the current state with a play-out meaning playing the game until it ends by either a player winning or the game drawing. The results of these simulations will be backpropagated through the tree. If the game ends in a loss from the perspective of the MCTS agent, a value of 0 will be returned. If the game ends in a win for the MCTS agent, a value of 1 will be returned. If the game ends in a draw, a value of 0.5 will be returned. This is done to prevent the algorithm losing a game that could also end in a draw.

3.1.4 Backpropagation

The value that is returned in the simulation phase will be backpropagated through the search tree. The visit count and win count for each node will updated.

3.1.5 Summary

These four steps, which can be seen in Figure 1, are repeated until the budget of the algorithm runs out. For quantum checkers, the budget is a fixed parameter that is determined beforehand. The value of this parameter affects the performance of MCTS significantly. The higher the value, the better the performance, but the slower the algorithm. The value for this parameter will be specified for each experiment in Section 5.1.

4 Design & Implementation

In this section the implementation and ruleset of quantum checkers is explained. For the experiments three different implementations will be used. Classic checkers, which used the original ruleset. Checkers with superpositions, which introduces superpositions and measurement to the game. And checkers with entanglement, which introduces superpositions, entanglement and measurement to the game. The source code for the game can be found at [18].

Section 4.1 introduces the rules of normal checkers on which quantum checkers is based. Section 4.2 explains how superpositions and entanglement are introduced. Section 4.3 explains the no double occupancy rule. Section 4.4 summarises all possible moves following these rule-sets. Section 4.5 introduces the Python library used to implement the game.

4.1 Original rule-set: classic checkers

The implementation of quantum checkers is based on the official rule-set for English Draughts. The rules of the International Draughts Federation [7] are used.

4.1.1 Setup

English draught is a game played between 2 players on an 8 by 8 board with alternating black and white tiles. The game is only played on the black tiles, and therefore only 32 tiles are used. The board has to be placed so that each player has a black square in the bottom left from their own perspective.

The game is played with 12 white pieces and 12 black/dark pieces. These pieces are placed on the black tiles on the first three rows from the perspective of each player.

In this implementation the white player always starts, similarly to chess.

4.1.2 Movement

- A piece can either be a man and a king.
- All pieces start as men.

- A man can only move **forwards** (from their perspective) on the diagonals to an empty square. See Figure 2 for an example.
- If a man reaches the end of the board, it is kinged.
- A king is able to move on all diagonals.

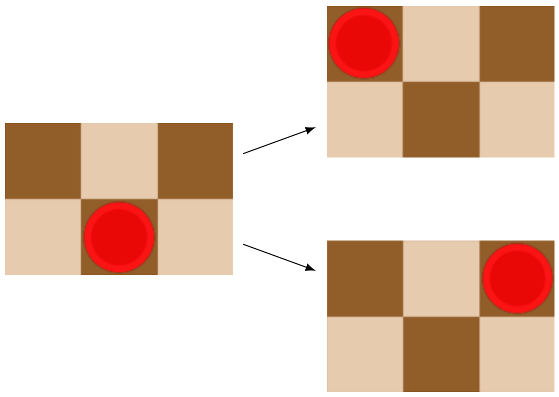
Figure 2: A normal move in classical checkers.

4.1.3 Capturing

- When a piece encounters an opponent's piece, on the diagonal it is able to move, and there is an empty square behind the opponent's piece, the player is obligated capture the opponent's piece by jumping over it. See Figure 3.
- If a piece just captured an opponent's piece, and it encounters another opponent's piece on the diagonal it is able to move, and there is an empty square behind the opponent's piece, the player is again obligated capture the opponent's piece by jumping over it. This rule is repeated for as long as the piece is able to capture the opponent's pieces.

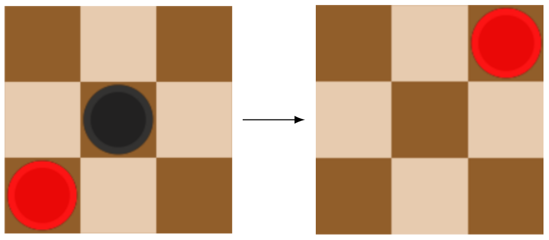

Figure 3: A move where the red player takes a piece of the black player.

4.1.4 Results

A player wins when his opponent:

- Has no pieces left.
- Has no legal moves left (e.g. all pieces are blocked).

The game ends in a draw when:

- There have been more than 40 moves without any player capturing a piece.

For a more in-depth explanation of the official rules, see the rules of the International Draughts Federation [7].

4.2 Quantum rule-set

The game of quantum checkers uses the previously mentioned rules, unless mentioned otherwise. This section is split into two parts. First, checkers with superpositions will be explained in Section 4.2.1. Next, checkers with entanglement will be explained in Section 4.2.2.

4.2.1 Checkers with superpositions

In this version of the game, superpositions are introduced. The most natural way to introduce superpositions into the game of checkers is to see each piece as a qubit. Since a qubit can exist in superposition, a piece will also be able to exist in superposition. Instead of doing a classical move where a piece moves to only one tile, a piece will be able to move to multiple tiles at once and therefore exist in superposition. This reasoning implies that two different pieces can exist in superposition on the same tile. This would make both the game itself and the implementation of the game more complicated. Therefore the no double occupancy rule, which will be described in Section 4.3, is enforced. It states that no two pieces can exist on the same tile.

With the introduction of superpositions, the rules of measurement also need to be defined. Again the most natural way to implement this is to measure when trying to capture a piece. If you attempt to capture a piece in superposition, you first need to verify it is actually there. If the piece collapses to the square that you were attempting to capture, you will be able to take the piece. However, if the piece collapses to a different square on the board than the square you tried to capture, your move "failed". This results in your piece not moving and your turn being wasted. This also means that in checkers with superpositions you are able to reach states that you are not able to in classic checkers, since you are not able to "pass" a turn in classical checkers. This measurement rule also applies in reverse, i.e. when you try to capture a piece with a piece in superposition, that piece first needs to be measured.

This version of quantum checkers is inherently probabilistic. When doing a split move, the probability for each piece is (approximately) half of that of the original piece. Therefore, all possible outcomes for a measurement can be calculated relatively easy.

In section 4.2.1.1 the movement of checkers with superpositions will be formally defined. Section 4.2.1.2 defines the capturing mechanics. Finally, Section 4.2.1.3 defines when measurement happens for checkers with superpositions.

4.2.1.1 Movement

- When a man is able to move in both diagonals, meaning both squares in front of the piece are empty, instead of moving to the left or right diagonal, it is also possible to move to both diagonals at the same time. This causes the piece to be in an equal superposition. Both squares will be occupied by the piece. No other pieces can occupy these squares.
- A king can move to a maximum of four diagonals. If possible, it can create a superposition for any combination of pairs of these four diagonals. It is not possible to move to more than two squares at a time.

- All rules apply to a piece in superposition, which means that it can again move in a superposition.
- See Figure 4 for an example of how a split move looks.

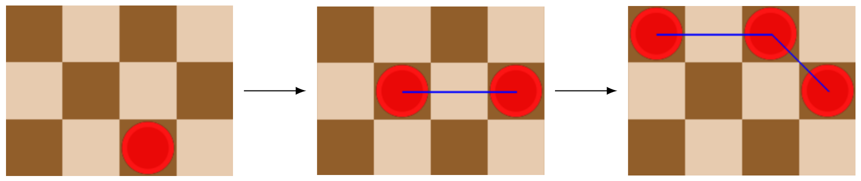

Figure 4: A move where the red player moves his piece into superposition multiple times. The pieces are connected by a blue line to indicate that it is the same piece in superposition.

4.2.1.2 Capturing

- If a piece that is **not** in superposition tries to capture a piece that is in superposition, a measurement happens on the piece that is trying to capture. If it turns out the piece that you are trying to capture is actually there, you will take the opponents piece. If it is not there, nothing happens and your turn is wasted. See Figure 5 for an example.
- If a piece that is in superposition tries to capture a piece that is **not** in superposition, a measurement happens on the piece that it is trying to capture. If it turns out the piece that you are using to capture is actually there, you will take the opponents piece. If it is not there, nothing happens and your turn is wasted.
- If a piece that is in superposition tries to capture a piece that is in superposition, a measurement happens on the piece that is trying to capture. If the measurement reveals that the piece you are using is actually there, a measurement happens on the piece that is captured. Otherwise, your turn ends here and the opponents piece is never measured.

4.2.1.3 Measurement

If a piece needs to be measured, it reverts back to a classical state. This means that of all possible squares the piece can exist in, only one will be

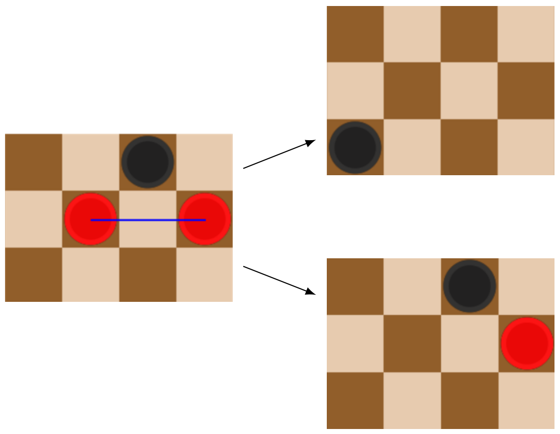

Figure 5: Two outcomes for a black piece trying to take a red piece in superposition for checkers with only superpositions.

true.

- For a piece in a superposition this means that for all possible positions the piece can be in, only one will be true. Only on this square the piece will remain. All other squares will be emptied.

4.2.2 Checkers with entanglement

Introducing entanglement into the game of checkers is already less straightforward. Checkers with superpositions can simply defined as probabilistic or stochastic checkers. Entanglement however, adds a layer of complexity that can't directly be translated to a simple definition. Entanglement happens when the state of one qubit directly relates to the state of another qubit. In quantum chess, entanglement happens when a piece tries to move directly through another piece in superposition [4]. E.g. when a pawn tries to move two tiles as its first move, but a piece in superposition is blocking its path.

In checkers you are never able to move through another piece like in chess, so this is not an option. Therefore, we decided that in quantum checkers entanglement happens when a classical piece tries to capture a piece in superposition. To reduce complexity, a piece can not be entangled with a piece that is already entangled. This does mean that the measurement rule for checkers with superpositions need to be modified. Measurement no longer always happen when trying to capture a piece, since there is the possibility for entanglement to happen. Instead, measurement happens in all the scenario's where entanglement does not happen. E.g. it does not happen when you try to capture a piece in superposition with a classical piece, but it does happen when you try to capture an entangled piece with a classical piece.

In Section 4.2.2.1 the movement of checkers with entanglement will be formally defined. Section 4.2.2.2 defines the capturing mechanics. Finally, Section 4.2.2.3 defines when measurement happens for checkers with superpositions and entanglement.

4.2.2.1 Movement

- An entangled piece is able to do a classical move or move itself into a superposition.

4.2.2.2 Capturing

- If a classical piece tries to capture a piece that is in superposition, this piece will become entangled with the piece that it was trying to capture. This means that the captured piece remains on the board. See Figure 6.
- If a piece that is **not** in superposition/entangled tries to capture a piece that is in superposition/entangled, a measurement happens on the piece that it is trying to capture. If it turns out the piece that it is trying to capture is actually there, the opponents piece will be taken. If it is not there, nothing happens and your turn is wasted.
- If an entangled piece tries to capture a piece that is **not** in superposition/entangled, a measurement happens on the piece that is trying to capture. If it turns out the piece that you are using to capture is actually there, you will take the opponents piece. If it is not there, nothing happens and your turn is wasted.
- If a piece that is in superposition/entangled tries to capture a piece that is in superposition/entangled, a measurement happens on the piece that is trying to capture. If the measurement reveals the piece that is trying

to capture is actually there, a measurement happens on the piece that is being captured.

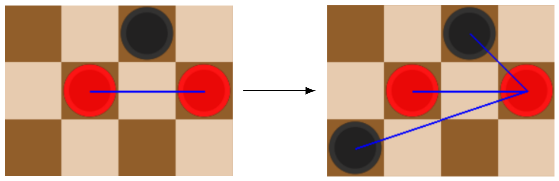

Figure 6: A black piece tries to take an opponents piece in superposition. The black and red pieces become entangled as indicated by the blue line.

4.2.2.3 Measurement

If a piece needs to measured, it reverts back to a classical state. This means that of all possible squares the piece can inhabit, only one will be true. This can mean two things:

- For an entangled piece from the perspective of the piece that was being captured, this can imply that it was initially captured. If it wasn't captured due to the piece never occupying that position, it indicates it occupies another square. Subsequently, all other squares will be emptied. In the event it was positioned on the square that was being captured, the piece will be taken off the board and all other squares will be emptied.
- For an entangled piece from the perspective of the piece that was trying to capture, this can imply that it never captured the piece it was trying to take. If that is the case, its state will collapse to a position where the opponents piece was never taken. If it was successful in taking the opponents piece, their piece will be removed from the board and the piece that was trying to capture will collapse to a position where it was successful in taking the opponents piece.

4.3 No double occupancy rule

Because a piece can exist in superposition, it implies that it is possible for a square to be occupied by two different pieces at the same time. This would make both the implementation of the game and the game itself more complicated. Therefore the no double occupancy rule, which is also used in quantum chess [4], is implemented. This rule states that a square cannot be occupied by two different pieces at the same time.

4.4 Summary of all possible states and moves

When taking all these considerations, the followings moves are possible:

4.4.1 Possible states

There are three possible states for a piece to be in:

- Classical state: A piece that only occupies one square.
- Superposed state: A piece that occupies multiply squares. This state collapses to a classical state when measured.
- Entangled state: A piece that occupies multiply squares. This piece is entangled with an opponents piece and its position depends on the position of the opponents piece. This state collapses to a classical state when the piece itself is measured or the opponents piece is measured.

4.4.2 Possible moves

There are 4 possible moves a piece can do:

- Classic move: If the diagonal on which the piece can move is unoccupied, it can move there.
- Split move: If both diagonals on which the piece can move are unoccupied, it can move into a superposition on both diagonals, regardless of whether the piece is already in superposition or entangled.
- Entangled move: If a **classical** piece attempts to capture a piece in **superposition**, both pieces become entangled.
- Capture move: If a piece attempts to capture an opponent's piece and entanglement will not occur, a measurement is performed on both pieces. This means both pieces return to a classical state. If the measurement confirms the move is legal, the capture is carried out. If not, no pieces move, but all measured pieces remain in their classical state.

4.5 Unitary

To develop quantum checkers, the python library Unitary [15] is used. This library is an extension of Google's Cirq library. It has been designed to make quantum game development more accessible to developers with little experience in quantum mechanics, like myself. Quantum chess has also been made with unitary [4, 15].

5 Experimental setup and analysis

This section discusses the setup and the results for three different experiments. The setup for the different experiments will be explained in Section 5.1.

The three different experiments will be discussed in the following sections:

First, the complexity of the game will be measured in Section 5.2. This is done by letting two random agents play against each other on different board sizes for each level of quantumness and comparing their performances. Next, the performance of the MCTS agent will measured in Section 5.3. This is achieved by letting the MCTS agent playing against random agent and heuristic agent and evaluating its win rate. Lastly, in Section 5.4, a TrueSkill rating will be calculated by letting the random agent, heuristic agent and two types of agents based on MCTS play in a tournament.

The white player starts for all experiments.

5.1 Setup

All experiments will be performed on games of checkers with varying levels of quantumness:

- Classical checkers: Normal checkers as explained in Section 4.1.
- Checkers with superpositions: The version of checkers where superpositions are added to the game as explained in Section 4.2.1.
- Checkers with entanglement: The version of checkers where entanglement is added to the game as explained in Section 4.2.2.

There are four different agents that can be used in each experiment. They will be explained here:

5.1.1 Random agent

The random agent is the easiest opponent to play against, as it always selects a move randomly from all possible legal moves.

5.1.2 Heuristic agent

The heuristic agent employs a straightforward look-ahead strategy. For all possible moves, it recursively evaluates the number of classical pieces on the board. A classic piece is worth one point. A piece in superposition, therefore occupying multiple squares, still counts as one piece. King pieces are worth double. Opponent pieces are worth negative points. Using these scores it selects the move with the highest score.

The heuristic agent used in the experiments will look two moves ahead. One move by itself and one move by its opponent.

5.1.3 MCTS agents

The workings of the MCTS agent are explained in Section 3.1. For the following experiments two types of MCTS agents will be utilised: The *low* MCTS agent and the *high* MCTS agent. The terms low and high refer to the different values for the rollout parameter. the low MCTS agent has a rollout value of 200, while the high MCTS agent has a rollout value of 800. Both agents have an exploration parameter c of $\sqrt{2}$.

5.2 Complexity

First, we measure the performance of two random agents on different board sizes. Important to note is that for all board size, only the first row for each player will be filled with pieces.

We look both at the average number of moves and the average time in seconds it takes for a game to end. These results can be found in Figures 7 through 10. In Figures 7 and 8 the results are obtained for a normal game specified by the rules in Section 4. In Figures 9 and 10 the draw condition for the game is removed. This means that the game will only end once one of the two players has no more legal moves. All averages have been calculated over a 100 games. These figures show that as board size increases, the average number of moves also increase, until a certain point. When there is no draw involved, the average number of moves and average time grow exponentially.

Following these results, the percentage of draws for different board sizes is calculated. These can be found in Table 1. As can be seen, the bigger the

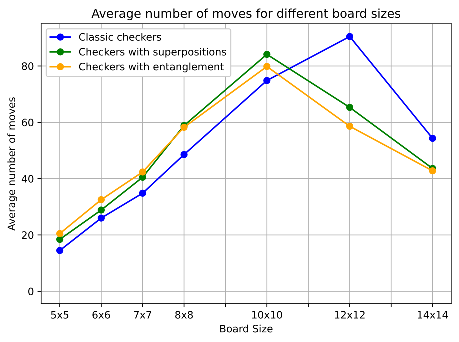

Figure 7: Average number of moves for a game of checkers for two random agents playing against each other for different board sizes.

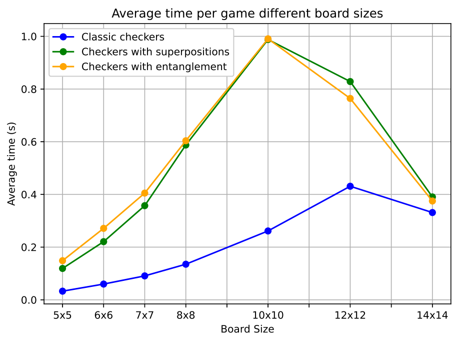

Figure 8: Average time in seconds for a game of checkers with two random agents playing against each other for different board sizes.

board size, the higher the chance two random agents playing against each other will result in a draw.

	5x5	6x6	7x7	8x8	10x10	12x12	14x14
Classic checkers	0.1	1.7	9.2	20.3	52.1	73.4	96.9
Checkers with superpositions	1.8	8.6	27.3	38.0	69.9	92.7	99.7
Checkers with entanglement	2.2	9.4	25.4	43.1	69.9	94.1	99.7

Table 1: Percentage of draws for two random agents over 1000 games.

5.3 MCTS performance

In this section, the parameters for the MCTS agent are defined as follows: a budget of 800, which means that the rollout value for the agent is 800 and a c value of $\sqrt{2}$.

In Figure 11 the performance of the MCTS agent against a random agent can be seen for a 5x5 board. The first thing we can see is that the black agent slightly outperforms the white agent. From this we can conclude that black has a slight advantage on the 5x5 board.

The second thing to notice is that the performance of the MCTS decreases with a higher level of quantumness. For checkers with superpositions, the

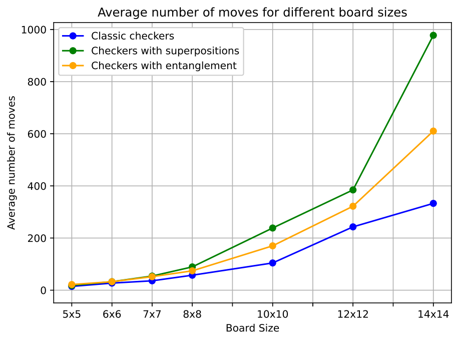

Figure 9: Average number of moves of a game of checkers for two random agents playing against each other for different board sizes without being able to draw.

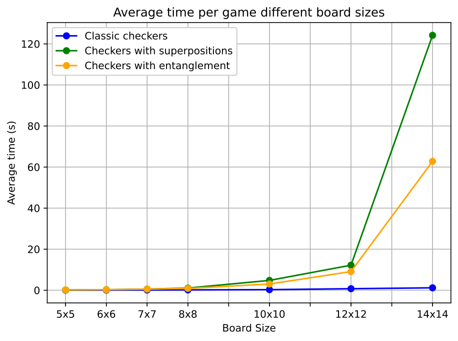

Figure 10: Average time in second a game of checkers for two random agents playing against each other for different board sizes without being able to draw.

MCTS agent only manages to win 81 percent of the time. For checkers with entanglement, the MCTS agent only wins 74 percent of the time. This can also be concluded with Figure 7, which indicates a higher branching factor when more quantumness is added to the game. The performance of MCTS suffers with a higher branching factor.

In another experiment, the MCTS agent played against a random agent on a standard 8x8 checkerboard, both as the black and white pieces. The board was set up with the usual three rows of pieces for each side. The larger board size substantially increases the number of possible moves for the MCTS agent. Additionally, the simulation phase for the MCTS agent continues until a player wins. Both of these factors mean these experiments take significantly longer to run. As a result, only ten games were conducted for each type of checkers (classic, superposition, and entanglement). Despite the increased complexity, the MCTS agent managed to win all the games for each type of checkers, even when playing as white.

This experiment was repeated for the MCTS agent against the heuristic agent for a total of 10 games. For classical checkers, the MCTS agent managed to win all games. For checkers with superpositions, the MCTS agent managed to seven times and drawing three times. For entanglement, the MCTS managed to win all games. It is worth noting that the MCTS agent performed better on checkers with entanglement in comparison to checkers with superpositions.

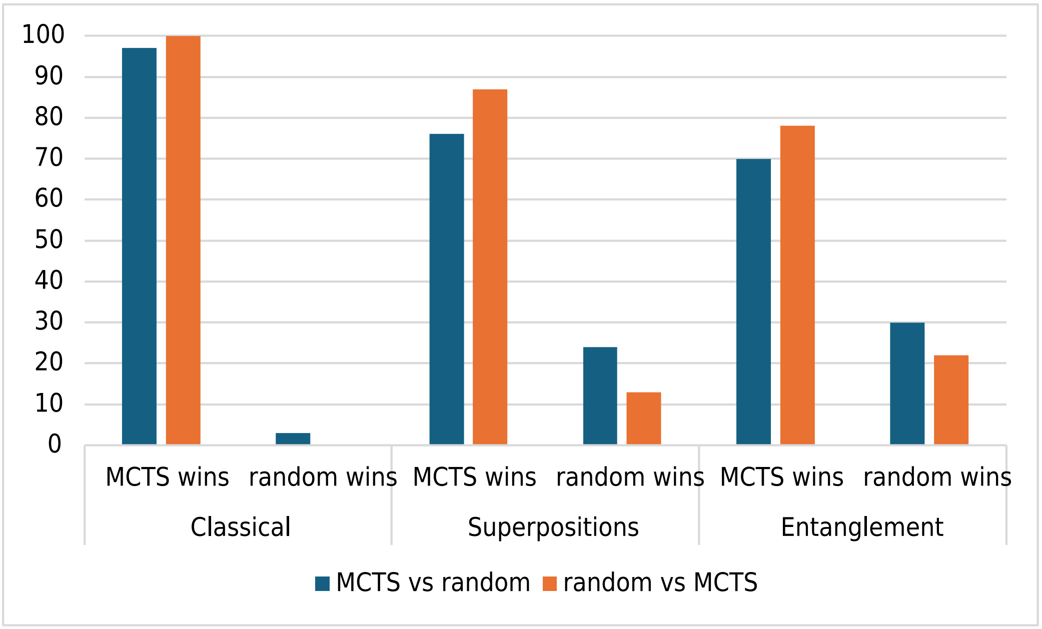

Figure 11: Win rate as percentage of MCTS agent against random agent both as the white and black player on a 5x5 board over a 100 games. The legend should be read as white versus black.

5.4 TrueSkill

TrueSkill is a ranking system for competitive games developed by Microsoft Research [10]. For this experiment, we will use four different agents and compare their performances to each other using the TrueSkill rating system. The four agents are:

- Random agent (Section 5.1.1)
- Heuristic agent (Section 5.1.2)
- Low MCTS agent: 200 rollouts (Section 5.1.3)
- High MCTS agent: 800 rollouts (Section 5.1.3)

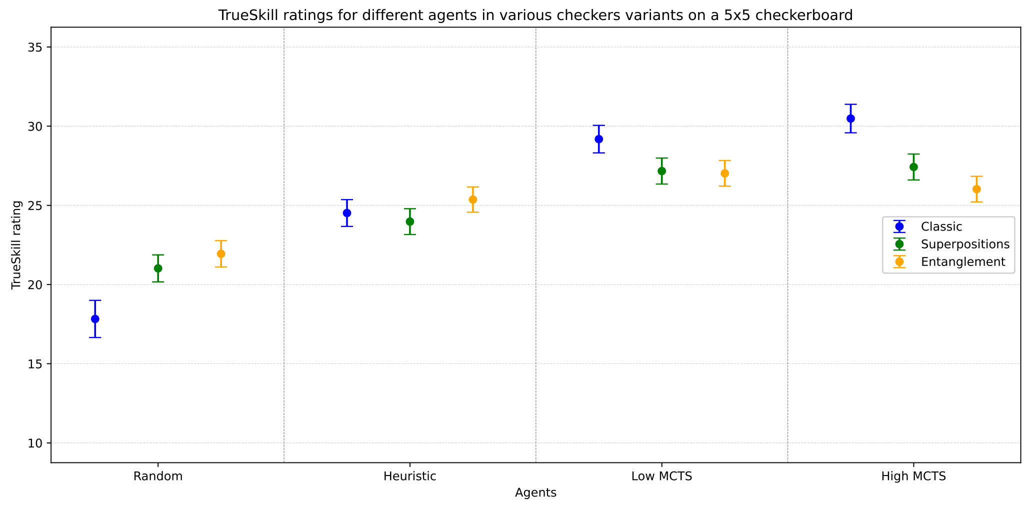

Figure 12: TrueSkill rating for 150 games per agent over a total of 300 matches on a 5x5 checkerboard with one row of pieces.

A players skill is represented as a Gaussian distribution, using two parameters. μ represent the average skill of a player, σ is the confidence of the guessed rating. Each agent will start with the default TrueSkill rating which is a skill level μ of 25 and a confidence rating σ of 8.333. The skill level will be adjusted when a player wins or loses a game. For each game played the confidence rating will go down. A lower confidence rating means that the rating is more accurate.

5.4.1 Results 5x5

The TrueSkill ratings after a tournament on the 5x5 board for each type of checkers (classic checkers, checkers with superpositions, checkers with entanglement) can be seen in Figure 12.

On the 5x5 board, the random agent performs the worst and has therefore the lowest rating across all types of checkers. The rating of the random agent increases as more quantumness is added to the game. The heuristic agent already performs significantly better as the random agent. This is to be expected, as the agent tries to maximise the number of pieces it has. Both MCTS agents outperform the random and heuristic agent, with the high MCTS agent performing the best.

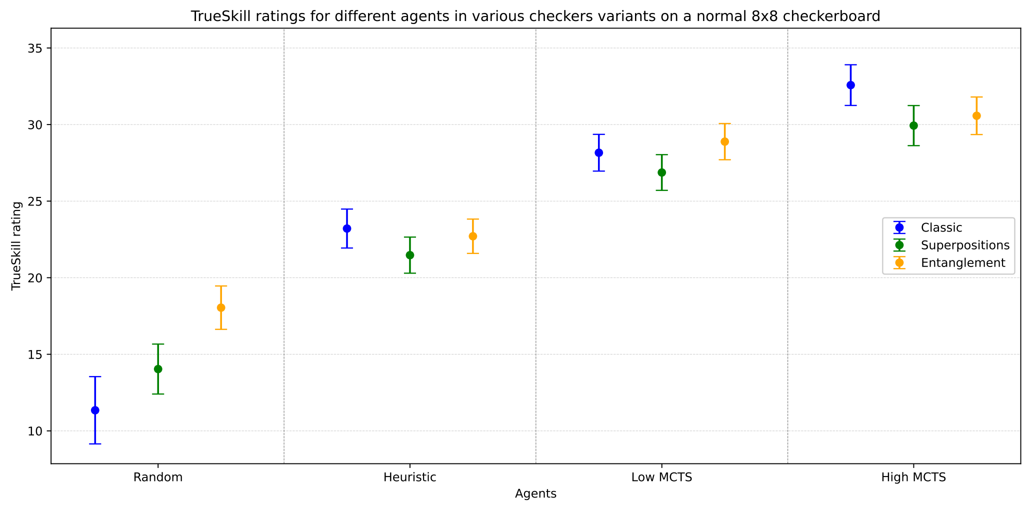

Figure 13: TrueSkill rating for 60 games per agent over a total of 120 matches on a standard 8x8 checkerboard.

When comparing the final ratings to the default ratings the agents started with, both the random and heuristic agent ended up losing some points, while the MCTS agent increased their rating.

When comparing the MCTS agent, their best performance is in classical checkers. Their rating gradually decreases as the quantumness increases. This is caused by the higher branching factor, which was indicated by Figure 7.

For classical checkers, the high MCTS agent outperforms the low MCTS agent. For checkers with superpositions, both agents demonstrate similar performance levels. In checkers featuring entanglement, the low MCTS agent slightly outperforms the high MCTS agent.

5.4.2 Results 8x8

The TrueSkill ratings after a tournament on the standard checkerboard (three starting rows of pieces) for each type of checkers (classic checkers, checkers with superpositions, checkers with entanglement) can be seen in Figure 13.

The standard deviation for each entry in the figure is notably higher. Because the game tree for an 8x8 board is remarkably bigger, the number of games to get a rating had to be decreased. This results in lower confidence for each rating.

Nonetheless, we can still see the same linear trend where the random agent performs the worst, and the high MCTS agent performs the best. Interesting to see here is that the high MCTS agent has a significantly better rating on the 8x8 board in comparison to the 5x5 board, even when taking the standard deviation in account.

Another interesting thing to note is that the trend for the ratings for both MCTS bots differs on the 5x5 board. For the 5x5 board the rating decreases the more quantumness is added to the game. However, on the 8x8 board the agents seems to perform better with the introduction of entanglement. As to why this is will be discussed in next section, Section 6.

6 Discussion

6.1 Complexity

The first thing to notice in Figures 7 and 8 is that the average number of moves and average time increase when superpositions are introduced. This is to be expected, considering introducing superpositions adds a probabilistic nature to the game.

An interesting observation that can be made is that the average time and number of moves for checkers with entanglement is lower, especially when the game doesn't draw, than that for checkers with superpositions. Apparently introducing entanglement to the game also introduce mechanics to the game that allow it to end quicker.

For big enough board sizes, the average number of moves and average time decreases again. This is caused by the fact that on board sizes this big, a draw happens much quicker.

When the draw rule is removed, the number of moves and time per game grows exponentially. (Figures 9, 10). This it to be expected, since the board size also grow exponentially.

Following these results, the percentage of draws for different board sizes is calculated. These can be found in Table 1. As expected, the bigger the board size and the more quantumness added to the game, the more games end in a draw.

From this we can conclude that the complexity of the game of checkers increases by introducing more quantumness to game.

6.2 MCTS performance

As depicted in Figure 5.3 the performance of the MCTS agent decreases on the 5x5 board with increasing levels of quantumness. This decline in performance can be attributed to both the higher complexity, and the increased level of randomness when the quantumness is increased.

Some smaller experiments were executed on the 8x8 board. In the experiments where the MCTS agent played against the random agent the MCTS agent was able to win all games. In the experiments where the MCTS agent played against the heuristic agent, the MCTS agents was able to win all games, except for checkers with superpositions. There the heuristic agent managed to draw three times. The sample size is insufficiently small to draw any conclusions, but it is still interesting to note that the MCTS agent performed better on checkers with entanglement in comparison to checkers with superpositions.

6.3 TrueSkill

6.3.1 5x5 board

As expected, the rating for the random agent is the lowest for all types of checkers. It is worth mentioning that the random agent performs significantly better when superposition and entanglement are introduced to the game. The random agent does not suffer from the increased complexity when introducing quantum mechanics to the game, since there is no strategy involved in choosing its moves. If the agent loses less games due to the increased complexity, and consequently draws/wins more games, its rating will remain higher.

As anticipated, the heuristic agent performs better than the random agent, and the MCTS agents perform better than both of them. Also, the performance of the MCTS degrades as the quantumness of the game increases. This is to be expected, as previous experiments showed introducing more quantumness increases the complexity of the game.

6.4 8x8 board

The most interesting observation from these results is that the MCTS agents perform better when entanglement is introduced in comparison to when only superposition is introduced. This is in contradiction with the results on the 5x5 board. Previously, in Figure 9, we have seen that the average number of moves for a game of two random agents is also lower for checkers with entanglement than for checkers with superpositions. There are two possible explanations for this.

The first one is that introducing entanglement to the game allows for a strategy to emerge. This strategy would not be possible on the 5x5 board, since it is too small and games end to quickly. Both the low and high MCTS agents perform better when entanglement is introduced, so it would seem both agents were able exploit this strategy.

Another possibility is that to prevent losing, agents will keep repeating superposition moves to reduce the possibility of their pieces being captured, and as a logical consequence the games will last longer. However, this strategy should also be feasible to a certain extent in checkers with entanglement.

An analysis of the gameplay itself will need to be done to determine if it is true that the agent is able to exploit a new strategy.

7 Conclusion

The first research question: *How to quantize the game of checkers?* directly relates to the implementation of the game itself.

In this thesis we presented the first quantized version of checkers. This implementation is successful in introducing several concepts of quantum computing. The execution of superpositions and measurement into checkers are straightforward and easy to grasp. However, implementing entanglement was significantly more difficult, as the concept itself is already more difficult to grasp. Nonetheless, the current version successfully introduces entanglement in an accessible manner.

The most difficult part was not the implementation of the game itself, but the implementation of Monte Carlo Tree Search. By introducing more quantumness, probabilities are introduced to the game. The original implementation of MCTS [12] does not work with probabilities, and therefore some modifications had to be done to the algorithm. The experiments show that these modifications were successful, since the MCTS agent was able to outperform the heuristics agent.

The second research question *What new strategies emerge from the quantum version of the game compared to the classical version?* asks if new strategies emerge when introducing quantumness to the game. The experiments show that this is the case. On the 5x5 board, there was a 100 percent chance for the black player to never lose. However, with the introduction of superpositions and entanglement, the white player is able to win. This already demonstrates that new strategies allow the player to win, if they use quantum moves to their advantage.

Also, the improvement of the performance of the MCTS agent for checkers with entanglement in relation to checkers with superpositions shows that the agent was able to use a new strategy that was not yet possible in checkers with superpositions.

7.1 Future work

In this implementation of quantum checkers we introduce the concepts of superpositions, entanglement and measurement. In quantum chess they also implement interference [4]. They claim that interference allows for new strategies to emerge in their game, by deliberately causing constructive and destructive interference. It would be interesting to see how interference could best be represented in the game of quantum checkers.

The experiments showed that a new strategy emerged when introducing entanglement. However, the analysis only showed that this strategy emerged, and not what the strategy is. It would be interesting to see what strategy the MCTS agent was able to take advantage of.

The current implementation of MCTS plays entire games in the simulation phase of a iteration of MCTS instead of having a budget cut-off (e.g. time). This causes the run-time of the algorithm to be unnecessarily long. Especially at the beginning of a game, such extensive simulations are not necessary and waste resources. A heuristic would need to be implemented that can be used to evaluate the board when the algorithm reaches a certain threshold. E.g. when the simulation has been running for a specified maximum duration or when a maximum number of moves has been reached.

References

[1] Selim G Akl. On the importance of being quantum. *Parallel processing letters*, 20(03):275–286, 2010.

[2] Selim G Akl. The quantum chess story. *Int. J. Unconv. Comput.*, 12(2-3):207–219, 2016.

[3] Kyle Burke, Matthew Ferland, and Shang-Hua Teng. Quantum combinatorial games: Structures and computational complexity. *arXiv preprint arXiv:2011.03704*, 2020.

[4] Christoph M. Cantwell. Quantum chess: Developing a mathematical framework and design methodology for creating quantum games. *arXiv: Quantum Physics*, 2019.

[5] LK Chen, Huiling Ang, D Kiang, LC Kwek, and CF Lo. Quantum prisoner dilemma under decoherence. *Physics Letters A*, 316(5):317–323, 2003.

[6] Paul Dorbec and Mehdi Mhalla. Toward quantum combinatorial games. *arXiv preprint arXiv:1701.02193*, 2017.

[7] International Draughts Federation. Rules of the game - IDF — International Draughts Federation — idf64.org. https://idf64.org/rules-of-the-game/. [Accessed 08-04-2024].

[8] Allan Goff. Quantum tic-tac-toe: A teaching metaphor for superposition in quantum mechanics. *American Journal of Physics*, 74(11):962–973, 2006.

[9] Michal Gordon and Goren Gordon. Quantum computer games: quantum minesweeper. *Physics Education*, 45(4):372, 2010.

[10] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.

[11] Steven James, George Konidaris, and Benjamin Rosman. An analysis of monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[12] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[13] Richard E Mayer. Computer games in education. *Annual review of psychology*, 70:531–549, 2019.

[14] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition, 2010.

[15] quantum chess engineering group at google. Unitary. https://github.com/quantumlib/unitary. [Accessed 08-04-2024].

[16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[17] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[18] Luuk van den Nouweland. Quantum checkers. Source code availible at https://github.com/LuukvandenNouweland/quantum_checkers. [Accessed 08-04-2024].

[19] Evert van Nieuwenburg. Quantum tiq taq toe. https://quantumtictactoe.com/play/, 2019. [Accessed 08-04-2024].