Opleiding Informatica

# Efficacy and Characteristics of Search Algorithms in Quantum Tic-Tac-Toe

Kah Ming, Wong, s2641976

Supervisors:
Dr. E.P.L. van Nieuwenburg

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

31/07/2023

## Abstract

Quantum Tic-Tac-Toe is a game that introduces chance and luck into its game mechanics due to quantum mechanics. As such, the game is much harder to play correctly. This thesis puts the Monte Carlo algorithm against the expecti-minimax algorithm to see which of the algorithms performs better in Quantum Tic-Tac-Toe; as a means, a computer program has been created. In the program, surprisingly, both algorithms beat the other convincingly as long as they are player 1. This implies that player 1 holds a greater advantage over player 2 in Quantum Tic-Tac-Toe. This, however, is not explicitly the scope of this paper and should be researched further. Further improvements can be made by acquiring a real quantum computer, instead of using a random generator for the quantum mechanics; as the real quantum computer includes realistic noise. The playouts and depth, too, can be increased for potential better performance of the algorithms.

# Contents

**1 Introduction** **1**
1.1 Thesis overview . . . 1

**2 Related Work** **2**
2.1 Quantum and Quantum Games . . . 2
2.1.1 Superposition . . . 2
2.1.2 Entanglement . . . 2
2.1.3 Collapse . . . 3
2.1.4 Quantum Games . . . 3
2.2 Classic Tic-Tac-Toe . . . 4
2.3 Quantum Tic-Tac-Toe . . . 5
2.3.1 Minimal Quantumness . . . 6
2.3.2 Moderate Quantumness . . . 7
2.3.3 High Quantumness . . . 8
2.4 Monte Carlo Search . . . 8
2.5 Minimax Search . . . 9
2.6 Alpha-Beta Pruning . . . 12
2.7 Expecti-Minimax Search . . . 13

**3 Design** **13**
3.1 The Game Modes . . . 14
3.2 The Tic-Tac-Toe Class . . . 14
3.3 The Board . . . 15
3.3.1 The Random Agent . . . 15
3.3.2 The Monte Carlo Agent . . . 16
3.3.3 The (Expecti-)Minimax Agent . . . 16
3.4 Heuristic . . . 16

**4 Experiments** **17**
4.1 Analyzing the Expecti-Minimax Search . . . 17
4.2 Agent vs Agent . . . 19
4.2.1 Monte Carlo Agent vs Random Agent . . . 19
4.2.2 Expecti-Minimax Agent vs Random Agent . . . 21
4.2.3 Expecti-Minimax Agent vs Monte Carlo Agent . . . 23

**5 Conclusions and Further Research** **27**

**References** **28**

# 1 Introduction

Tic-Tac-Toe is a well-known board game that is played by many. Here, two players take turns marking empty spaces of a 3-by-3 board. The player who gets a line of their marks first, whether horizontally, vertically, or, diagonally, wins the game. Tic-Tac-Toe is also known as a combinatorial game and a solved game. Combinatorial, because of its deterministic nature, wherein, chance and luck does not play a role; and, furthermore, due to the fact that no information is hidden from either player. There exists perfect information in the game. It is solved, because at any given state of the game, the outcome of it can be correctly predicted; this, given that both players play optimally.

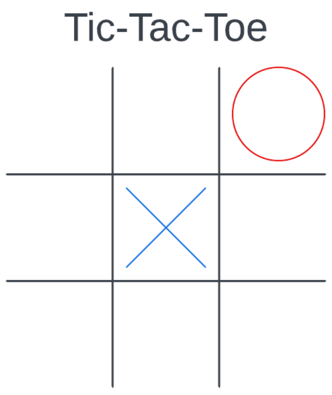

Figure 1: Representation of Tic-Tac-Toe.

Quantum Tic-Tac-Toe is Tic-Tac-Toe with quantum mechanics as game mechanics. This, in contrast to Tic-Tac-Toe, introduces chance and luck into the game. Making it more difficult to be played optimally.

The scope of this thesis is to answer the research question: “Does the expecti-minimax algorithm perform better than the Monte Carlo algorithm in Quantum Tic-Tac-Toe?”.

## 1.1 Thesis overview

The following sections can be found in this thesis:

- Section 1: In this section, introduction to the subject and the research question is given.
- Section 2: In this section, all the necessary background information and related work is given.
- Section 3: In this section, the design of the program will be elaborated on.
- Section 4: In this section, the experiments will be elaborated on, and the results will be presented.
- Section 5: In this section, conclusions will be made through the results of the experiments, and potential further research will be given.

## 2 Related Work

This section aims to provide insights on all the related information surrounding Quantum Tic-Tac-Toe. This will give an explanation on what Quantum Tic-Tac-Toe exactly is, what it is related to, and, furthermore, how we will continue forward with our research on Quantum Tic-Tac-Toe.

### 2.1 Quantum and Quantum Games

Needless to say, the game *Quantum Tic-Tac-Toe* is based on quantum mechanics. Quantum Tic-Tac-Toe can, therefore, in a way be called a *Quantum Game*. Prior existences of Quantum Games have existed before Quantum Tic-Tac-Toe. These so called Quantum Games all have the same principles in quantum mechanics, which, each and every single one of them makes use of:

- Superposition
- Entanglement
- Collapse

#### 2.1.1 Superposition

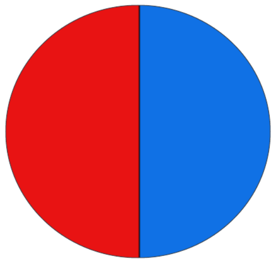

Figure 2: Example of an entity in two-state 50-percentage superposition.

*Superposition* is the principle in quantum mechanics such that an entity can be in two or more states at the same time. The idea is that said entity has a certain 'amplitude' to be in a given state. The amplitude is a complex number (e.g. $z$) whose absolute square (e.g. $|z|^2$) is the probability for the entity to be in that state. This gives precedence that the entity is not both states for certain, as it is those states for some amount of probability. It is, therefore, important to note that while in superposition, an entity cannot be fully assigned to some state. An entity in superposition begins with two states. However, these states can increase, such that an entity can be many different states for a certain amount of probability.

#### 2.1.2 Entanglement

*Entanglement* is another principle in quantum mechanics. This gives the notion of correlation between entities, such that, the state of one entity is directly correlated to the state of an entity which it is *entangled* to. The degree of entanglement can vary, however, it always starts between two entities; a two-entity entanglement. To expand on that, an entanglement can have more and more entities entangling with each other; virtually endlessly, and all those entities' state are directly correlated to each other. Entanglement actually quantifies correlations that go beyond classical correlations. For the sake of this thesis however, entanglement can be thought of as correlations.

### 2.1.3 Collapse

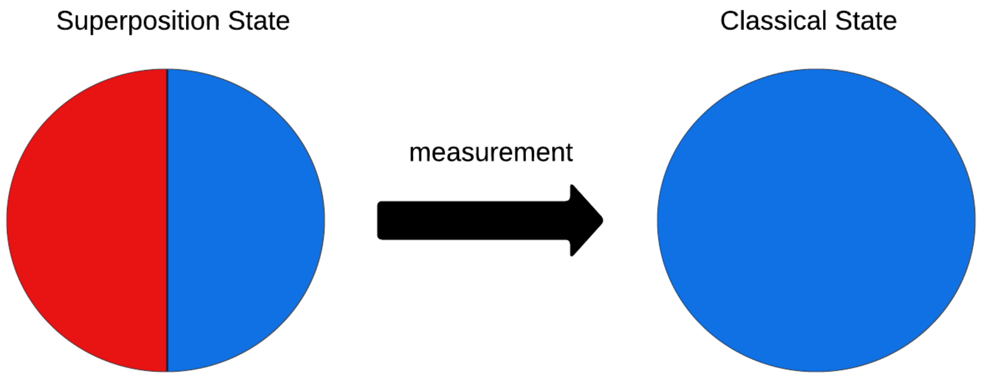

Figure 3: Example of a measurement.

The previous two principles, superposition and entanglement, both are based on chances. The state of an entity in either phases is always uncertain. However, through *collapse*, entities turn into one state for certain. This is done by first *measuring* the states. In the case of an entity in superposition, a measurement will observe in what state the entity is for certain. The entity in superposition, thus, turns into an entity which is in two or more states for a certain amount of percentage, into an entity that is in one and only one place for 100-percentage; making it an entity with a *classical* state. In the case of entities in an entanglement, only one of the entities need to be measured. As the states are directly correlated to each other, all the other entities will collapse to its correct state by observing the state of one entity.

### 2.1.4 Quantum Games

A Quantum Game is, thus, a game that bases its mechanics on the previous mentioned principles of quantum mechanics. While quantum games all have the same foundation in quantum mechanics, they can all be implemented in a different way. This is done by varying the amount of *quantum* in a game. For example, a limitation can be set on the degree of superposition and entanglement. To illustrate this, one game can limit itself to two-state superposition and two-entity entanglements (minimal quantumness), while another game sets no limits to the amount of entity superpositions and entanglements (high quantum). Hence, two games that are identical to each other in terms of mechanics and goal, can still turn wildly different in terms of playing the game through quantum; such that the optimal strategy in one game no longer applies to the other.

Essentially, any game that makes use of states can be turned into a variant of itself, which, instead, makes use of quantum mechanics; turning it into a quantum game. The well-known board game *chess* also has a quantum variant, unsurprisingly, called Quantum Chess. There exists different kinds of implementations of Quantum Chess. One implementation of Quantum Chess is by S. Akl, in which a piece begins with two different piece states; once a piece moves it will reveal its piece state. This implementation is briefly touched upon in S. Akl's paper *Technical Report No. 2010-568 on the Importance of being Quantum* [Akl10]. A recent implementation is done by Christopher Cantwell in his paper *Quantum Chess: Developing a Mathematical Framework and Design Methodology for Creating Quantum Games* [Can19]. Notably, in this paper Cantwell briefly mentioned Quantum Tic-Tac-Toe and described it as deterministic. This, however, is regarding a different implementation of Quantum Tic-Tac-Toe. This paper will focus on Quantum Tic-Tac-Toe that is not deterministic in the slightest, more on this later.

Quantum Fox-In-A-Hole is another Quantum Game, and is the quantum variant of the classic puzzle Fox-In-A-Hole. In the classic version there are 5 holes. A fox in hidden in one of those holes. Each night the fox moves either left or right into the neighbouring holes. The player guesses each morning where the fox is. The player wins if he guesses correctly, and loses if he does not do so in a certain amount of days. In the quantum variant, the fox moves both left and right each night. This, thus, results into superpositions of holes, in which, the fox has a certain amount of percentage to reside in. Whenever the player guesses, a measurement is taken place.

## 2.2 Classic Tic-Tac-Toe

To explain Quantum Tic-Tac-Toe, knowledge of Tic-Tac-Toe is first required. Classic Tic-Tac-Toe is a game, in which, two players take turns marking their plays on a 3-by-3 board. Of which, Player 1 marks their plays with $X$ and Player 2 marks their plays with $O$. Player 1, the one who marks the board with X, always begins first; marking one empty cell of the 3-by-3 board. Player 2, by the same process, then marks one empty cell on the 3-by-3 board with an O; this process repeats. A player wins if they get a 3-in-a-row of their marks either horizontally, vertically or diagonally. A draw is forced when the board is full, and, neither players have a 3-in-a-row.

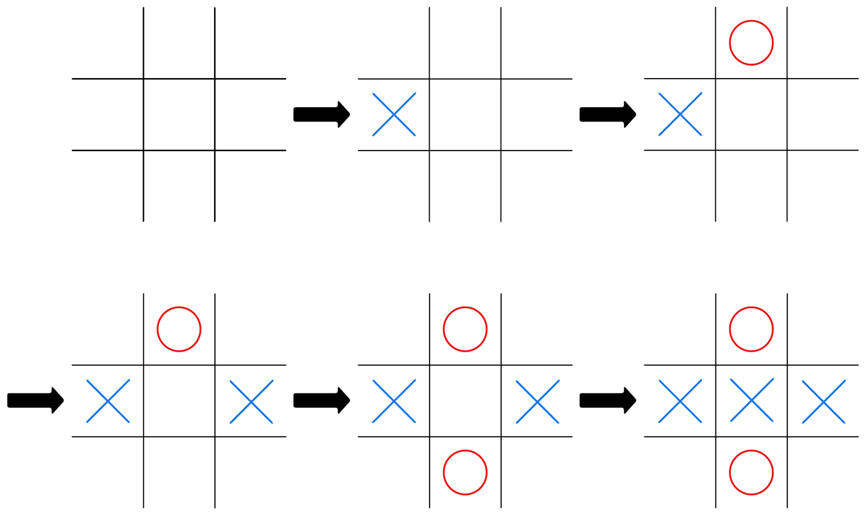

Figure 4: Example of a play through of Classic Tic-Tac-Toe.

Figure 4 shows a play through example of a Classic Tic-Tac-Toe game. In which, Player 1 wins in the end, as they have created a 3-in-a-row. In Classic Tic-Tac-Toe luck or chance does not play a role in its mechanics. Both player can fully observe the board at any time, and each player know what the opposing player can and can not do. Classic Tic-Tac-Toe is, therefore, a game with perfect information; and, thus, a combinatorial game. Furthermore, Classic Tic-Tac-Toe with its relatively small amount of states is a solved game. Such that, as long as each player keeps playing optimally, the game will always result in a draw.

## 2.3 Quantum Tic-Tac-Toe

Quantum Tic-Tac-Toe incorporates quantum mechanics into the Classic Tic-Tac-Toe. The implementation this paper uses is TiqTaqToe, which, currently, can be accessed in browser form [vN]. This implementation of Quantum Tic-Tac-Toe consist of various game modes. The following list denotes the game modes, of which, increases the quantum with each step:

1. No Quantumness: Classic Tic-Tac-Toe
2. Minimal Quantumness: Tic-Tac-Toe with two-state superposition
3. Moderate Quantumness: Tic-Tac-Toe with two-state superposition, and entanglement with classic state
4. High Quantumness: Tic-Tac-Toe with two-state superposition, and entanglement with superposition

It is important to note, with each increase in quantum, the next game mode is essentially the same as the previous one; however, with extra features. Moreover, this implementation of Quantum Tic-Tac-Toe is setting a hard limitations on a few aspects of quantum. Namely, if an entity is in superposition (and not entangled) it can only be in a two-state superposition. In a similar vein, limitations are set on the entanglement, such that infinite entanglement can not occur. Moreover, a player can only create an entanglement with a move from the opposing player.

On a separate note, as Tic-Tac-Toe gets incorporated with quantum mechanics, it turns from a perfect information game, into an imperfect information game. That is because, as long as superposition moves are on the board, the board can never with certainty be assessed which state it will hold after the measurement; and, thus, chance and luck does play a role in Quantum Tic-Tac-Toe with superposition moves.

### 2.3.1 Minimal Quantumness

In this stage of Quantum Tic-Tac-Toe, superposition moves are introduced to Classic Tic-Tac-Toe. For clarity, a move refers to the marks of the players they make each turn. A move from the Classical Tic-Tac-Toe is referred to as a classic move, as they do not have any quantumness. In Minimal Quantumness, each turn a player can choose between two moves. Either a classic move or a superstition move. A superstition move in Minimal Quantumness is limited to empty cells, and no entanglements may occur. Specifically, a superposition move in Minimal Quantumness is made by marking two empty cells in a turn.

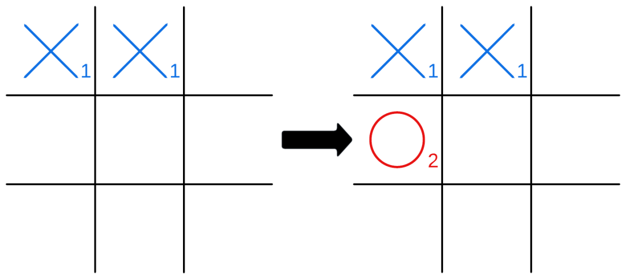

Figure 5: Example of a play through of Minimal Quantumness.

Figure 5 shows an example, in which, Player 1 makes a superposition move. These moves are marked with the current turn. $X_1$ is in this case at the marked cells for both 50-percentage. Player 2, in the following turn, decides to make a classic move marked as $O_2$; as it is now the second turn of the game. This process repeats itself until there is a classical 3-in-a-row, or, if the board is full. At a full board, the superposition moves are measured. In the case of Figure 5, one of the $X_1$ will be removed. Turning $X_1$ in a classic move. After a measurement takes place, one of the following three scenarios can happen:

1. Board is measured, however, no players have won; and, there are still empty cells.
2. Board is measured, one player got a 3-in-a-row after measurement took place.
3. Board is measured, both player got a 3-in-a-row after measurement took place.

In the first case, since no one has won and there are still empty cells, the game will continue with the pre-existing rules; this until the board is full with classical moves, or a player has won through 3-in-a-row. For the other two cases the same rule applies, a player receives a point if they get a 3-in-a-row.

It is important to note that a 3-in-a-row can only be formed with classical moves. Thus, if there is a 3-in-a-row, but, made from one or more superposition move; then it is invalid. Figure 6 illustrates this.

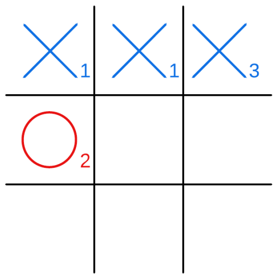

Figure 6: Example an invalid 3-in-a-row.

Here, Player 1 has a 3-in-a-row. However, this 3-in-a-row is invalid as $X_1$ is a superposition move. The game will, thus, continue on until there is a 3-in-a-row made of classical moves, or, the board is full of classical moves.

### 2.3.2 Moderate Quantumness

In Minimal Quantumness there was a hard limit set on superposition moves, namely, a superposition move can only be made on two empty cells. Moderate Quantumness expands on that, such that superposition moves can now also be made on a classical move of the opposing player; causing entanglements to occur. Figure 7 begins with a classical move $X_1$. Player 2 then makes a superposition

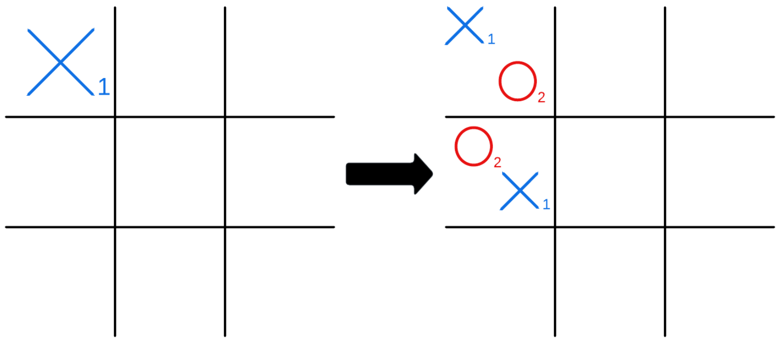

Figure 7: Example an entanglement in Moderate Quantumness.

move, of which, one of the cell contains the classical move of Player 1. Both $X_1$ and $O_2$ is in either cell for a 50-percentage probability. Once the board is full, a measurement is taken place. Moreover, it can be clearly seen that once either $X_1$ or $O_2$ is observed to be in one cell, then the other mark collapses into the remaining cell.

### 2.3.3 High Quantumness

In a similar vein, High Quantumness unlocks another limitation of Moderate Quantumness by, now, allowing superposition moves on a superposition move of the opposing player.

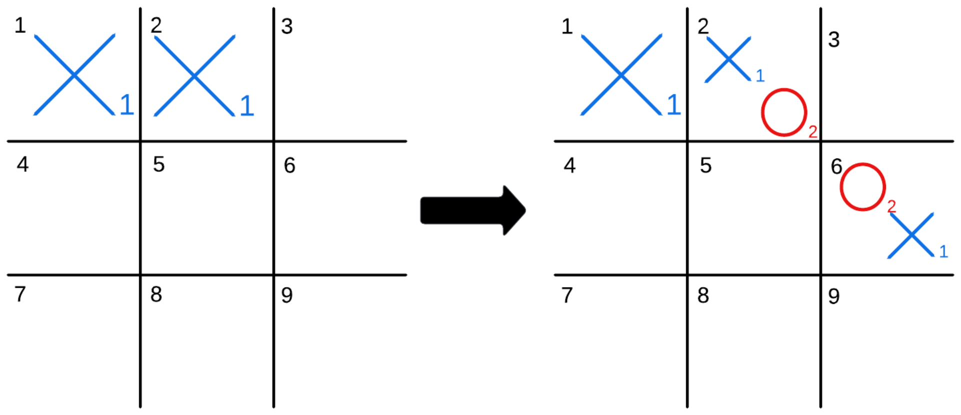

Figure 8: Example an entanglement of High Quantumness.

Figure 8 begins with a superposition move from Player 1 in cell 1 and 2. Player 2 in the following turn also makes a superposition move, with one of those moves being cell number 2; the cell containing a superposition move from Player 1. An entanglement occurs between entity $X_1$ and $O_2$. For $O_2$ the same thing as last time applies, in which, it has a 50-percentage probability being in cell 2, and a 50-percentage probability being in cell 6 after a measurement takes place. However, this does not exactly apply to $X_1$. Namely, $X_1$ in cell 1 still has the 50-percentage probability from the initial superposition move. The percentage probability of $X_1$ in cell 2, though, is in turn split again. Thus, causing the $X_1$ in cell 2 and cell 6 to have a probability of 25-percentage each instead, after the entanglement.

## 2.4 Monte Carlo Search

A common algorithm to explore the most likely optimal move is done through Monte Carlo Search. This algorithm does this through many random sampling. In regards to Quantum Tic-Tac-Toe, the randomness of the game is accounted for by a large amount of random sampling. To make use of Monte Carlo Search, a root first has to be determined, which, usually, is the current state of the game. From the root all available moves need to be determined. Once that has been done, a certain amount of playouts will be played with each available move as the first move, and, after the initial move, the game will be played with random moves. A playout refers to a simulation of the game being played until the end, resulting into either a win, a loss or a draw.

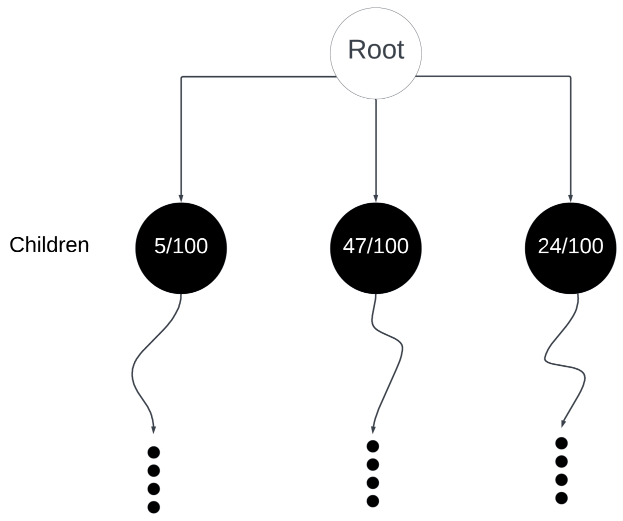

Figure 9: Example a Monte Carlo Search.

Figure 9 shows a root with 3 available moves, which, thus, leads to 3 children nodes. In those nodes, a certain amount of playthroughs will be played. In this case, specifically, an amount of 100. The numbers in the children nodes denotes the amount of game they have won. From left to right, the first one has won an amount of 5 out of the 100, the second one has won 47 out of the 100, and lastly, the third one has won 24 out of the 100. In Monte Carlo Search, picking the node with the highest amount of won playthroughs is sufficient to determine the ”most likely to be the optimal” move.

## 2.5 Minimax Search

Minimax Search is a well-known algorithm in the field of Artificial Intelligence. In a Minimax algorithm there exists a maximizing player (henceforth, MAX) and a minimizing player (henceforth, MIN). A MAX player, as the name suggests, aims to maximize the score in a search tree. In contrast, a MIN player aims to minimize the score in a search tree. Through choosing the highest and lowest scores, the MAX and MIN players, essentially, keep making their optimal move at each turn. The scores of a minimax tree is determined at the leaves. A minimax search is recursive, in the sense that the MAX and MIN each take turn choosing the highest and lowest score of their children respectively.

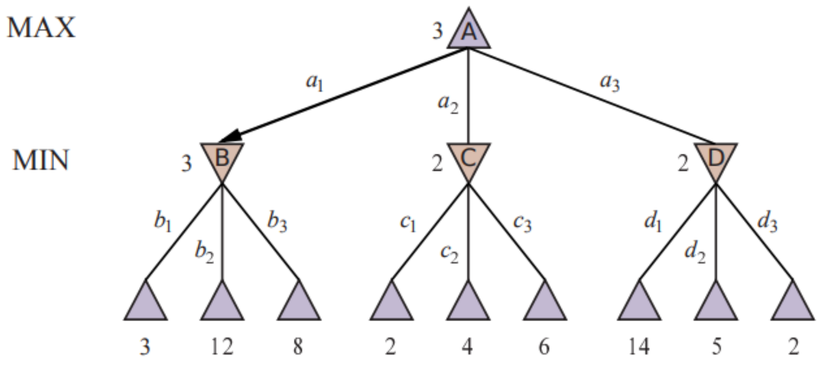

Figure 10: Two-player minimax game tree[RN20].

Figure 10 shows an example of a game tree. It starts from the root $A$, the MAX player. This root, in particular, has three moves, namely $a_1$, $a_2$, and $a_3$. These moves lead to the children nodes B, C, and D. These children nodes are from the MIN player. In a larger tree, the levels will keep interchanging between the MAX and MIN player.

From the root, a depth-first search to the leaves will be utilized. The leaves can be seen as the end-state of a game. For example, in Tic-Tac-Toe, it would be when a player has won the game, or, the game ended in a draw when the board has been filled completely with no winning player. To these states a score can be assigned, of which, the MAX and MIN players will keep choosing from. For instance, node B has the children nodes 3, 12 and 8 to choose from. Since node B is a MIN player it will, naturally, choose the lowest score of the children nodes. This being the 3. The same applies for node C and node D, but, of course, with different children nodes to choose from. Node A, being a MAX player, will choose the highest score amongst its children, and, thus, picks node B.

**Algorithm 1** Pseudo-code Minimax Algorithm

```
Minimax(depth, bestMove):
if Max depth is reached or leaf is reached then
    return Evaluation of node
end if
if MAX then
    maxEval = -$\infty$
    for Each available move of node do
        Play the move
        eval = minimax(depth-1, dummyMove)
        if eval $>$ maxEval then
            maxEval = Eval
            bestMove = current move
        end if
        Undo the move
    end for
    return maxEval
else
    minEval = $\infty$
    for Each available move of node do
        Play the move
        eval = minimax(depth-1, dummyMove)
        if eval $<$ minEval then
            minEval = Eval
            bestMove = current move
        end if
        Undo the move
    end for
    return minEval
end if
```

Notably, in the pseudo-code the bestMove is called by reference, and for the recurring calls of Minimax a dummyMove is passed instead. If not, the bestMove can potentially be assigned to a move deeper into the tree, which, is not the intention. Furthermore, in the pseudo-code, a move is being explicitly undone. However, in different Programming Languages, for instance, C++, a copy of the current object can instead be made. Thus, removing the need of undoing a move. The bestMove can be played once this has been determined through the algorithm.
The depth variable is, frankly, unneeded in a simple game such as Tic-Tac-Toe. However, Quantum Tic-Tac-Toe increases the search space significantly. Therefore, demanding the need of this variable, to end the search algorithm earlier; or, otherwise, risk an extremely long and unfeasible computation time.

## 2.6 Alpha-Beta Pruning

The depth variable is not the only way to alleviate the problem of unfeasible computation time in a minimax algorithm. Alpha-beta pruning, namely, is a technique to reduce the amount of nodes that is being evaluated; and, thus, reducing the computation time. This is done by keeping track of the lowest and highest score in a certain sub-tree; also called the alpha and the beta respectively. The two optimizing players, MAX and MIN, then compare its children with the alpha and the beta. If it is found that the children of a node leads to a worse score than the alpha and the beta, then that sub-tree can essentially be cut off. As, evidently, there appears to be a better and more optimal play that had been evaluated prior.

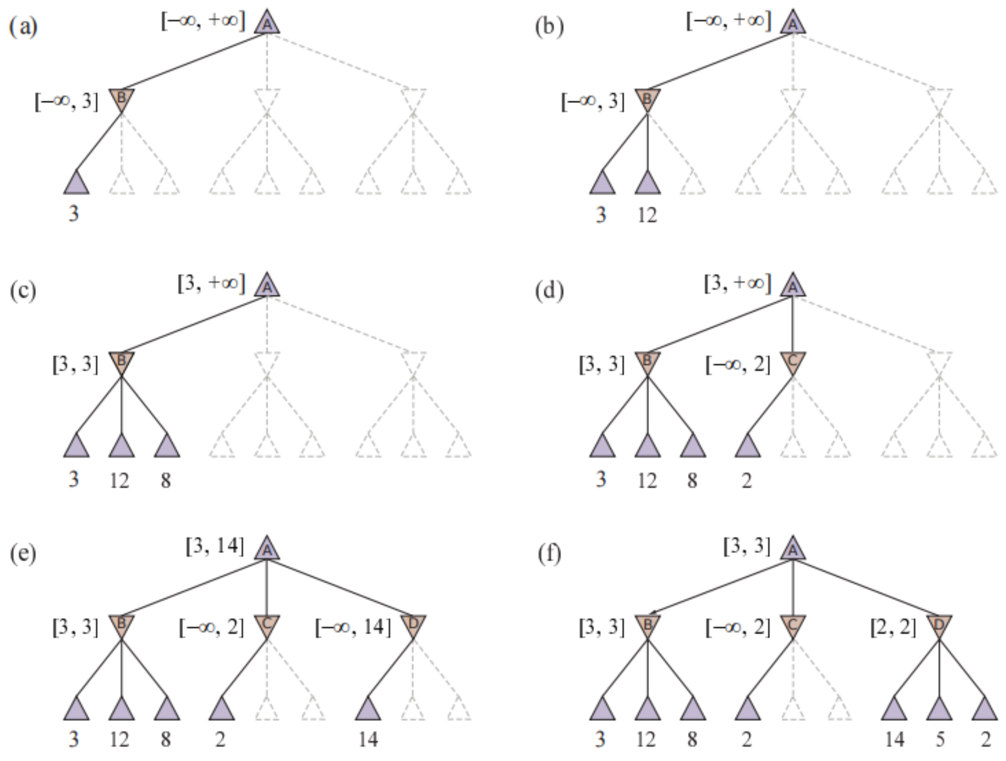

Figure 11: Example of Alpha-Beta pruning[RN20].

Figure 11 illustrates the alpha-beta pruning. Alpha is denoted by -$\infty$, while beta is denoted by +$\infty$. Moreover, the root A is a MAX player, while node B is a MIN player. In Figure 11*a*, node B evaluates its first child, which, appears to have a value of 3. Node B, whom is a MIN player, therefore has a value of *at most* 3. The beta for the sub-tree with B at its root is therefore 3.

In Figure 11*b*, B evaluates its second child with the value of 12. Again, since node B is a MIN player, it will not choose 12; and, thus, 3 is still the highest score of node B. Node B has evaluated all its children in Figure 11*c*, in which, it can be said for certain that the score will be 3; thus, turning both alpha and beta into a 3 of node B. Furthermore, since the score of B is now known, the alpha of root A can be adjusted. Considering that root A is a MAX player, it then can be said that root A has a score of *at least* 3; thus, changing its alpha into a value of 3.

Figure 11*d* shows the first and only instance of pruning in this example. Namely, node C evaluates its first child which has the value 2. Node C, whom is a MIN player, has a score of at most 2. In the perspective of root A then, whom has a score of at least 3, node C has a worse highest score than its (root A's) lowest score. Hence, the remaining nodes of node C can be pruned, decreasing the amount of computation time.

Figure 11*e* and Figure 11*f* shows how pruning can still be avoided depending on the ordering of the evaluations. Namely, node D first evaluates the child with the value of 14. Node D has, therefore, at most a value of 14. This value is higher than the lowest score of A. There is, thus, a need to evaluate further. Node B then evaluate its child with the value of 5, whom, again, has a higher score than the lowest possible score of root A. Node D then evaluate its last child with the value of 2, which is the child node D will end up choosing as node D is a MIN player. The score of node D ended up being worse than a score, of which, root A had already evaluated prior. Had, for instance, the child with the value of 2 been the first child that node D evaluated, then alpha-beta pruning could have prevented the algorithm from evaluating every single child; essentially, wasting computation time.

## 2.7 Expecti-Minimax Search

Expecti-minimax search is a variant of minimax search, which accounts for chance or luck in a game such as Quantum Tic-Tac-Toe. Herein, chance nodes are created to account for certain chance or luck elements in a game. In the standard minimax search the MAX and MIN node attains the value of the highest or lowest score of its children. In expecti-minimax, however, the MAX and MIN node can instead be a chance node. If so, then the MAX and MIN node attains not the value of one of its children, but, attains the average of all its children. Note that this means that every possible outcome still has to be computed in the case of expecti-minimax.

# 3 Design

This section will shed light on the computer program that will eventually be used to conduct experiments with. The main aspects that will be highlighted are the various agents, the game modes, the search algorithms, and other key aspects that make the computer program. The aim is to elaborate what exactly these key aspects are, and why they have been designed in such a way. Moreover, the computer program is programmed in C++. The main reason as to why C++ was picked, in favor of other programming languages, is primarily due to familiarity, speed, and the built-in libraries that can be used for the computer program.

## 3.1 The Game Modes

Section 2 provides a detailed overview of the game modes, namely Classic Tic-Tac-Toe, Minimal Quantumness Tic-Tac-Toe, Moderate Quantumness Tic-Tac-Toe, and High Quantumness Tic-Tac-Toe. All these game modes will be included into the computer program, on which, various agents will be played. The Classic Tic-Tac-Toe is here to provide an extra mean of benchmark, while the remaining three are the main point of interests. In particular, how the seach algorithms perform on the various different variants of Quantum Tic-Tac-Toe, and how well they take account for the chance and luck that gets introduced in each step of quantum increase.

## 3.2 The Tic-Tac-Toe Class

The Tic-Tac-Toe class is the class in the computer program for both the classical and quantum Tic-Tac-Toe. This class consists of various member-functions to play the game with, for instance, a function to automatically play a game with two agents, a function to play a move, a function to empty the board, a function to apply measurement on the board, etc. The class itself keeps track of a few important variables, of which includes:

- The current player as a bool variable, in which, false applies to Player 1 (X), while true applies to Player 2 (O)
- The current game mode as an integer variable, in which, 0 equals classic Tic-Tac-Toe, 1 equals Minimal Quantumness, 2 equals Moderate Quantumness, and, lastly, 3 equals High Quantumness
- The current turn of the game as an integer, starting with 0

Notably, since the *turn* variable starts with 0, in case of superstition or entanglement, Player 1 (X)'s turns will always be sub-scripted with an even integer, while Player 2 (O)'s turns will always be sub-scripted with an odd integer.
Another variable that is passed to the class, or, more specifically, the function to automatically play the game, is the game type of the two players. This variable is an integer, and, each player has their own integer parameter that gets passed to the class. The following applies to both integer parameters:

- If the player is a random agent, this integer will be 0
- If the player is a human agent, this integer will be 1
- If the player is a Monte Carlo agent, this integer will be 2
- If the player is an (expecti-)minimax agent, this integer will be 3

## 3.3 The Board

The board that the game will be played on is a standard 3-by-3 board, identical to the standard Classic Tic-Tac-Toe. However, there will be a distinction made in the computer program, specifically, between the board of a classic Tic-Tac-Toe, and, the board of the Quantum Tic-Tac-Toes. The board of the classical Tic-Tac-Toe with only classical moves will be called as *board*, while the quantum variant of this board will be called *qtBoard*; which, is to be used by the three variants of Quantum Tic-Tac-Toe. The (classical) board in the computer program is an integer array of the size 3-by-3. The following applies in the classical board:

- Player 1 (X) will be denoted with the integer 1
- Player 2 (O) will be denoted with the integer 2
- An empty cell will be denoted with the integer 0

The qtBoard, on the other hand, requires a lot more information than just who played which classical move where. For instance, a cell of a qtBoard needs to accommodate to the fact that two players can occupy that specific cell at the same time. In a similar vein, a qtBoard needs to keep track of superstition moves, entanglement, the percentage probabilities of each superstition or entanglement, and, which moves are exactly connected with each others. To illustrate an unwanted outcome, when measurement takes place of the move $X_2$, an unwanted outcome would be removing one of the other superstition move (e.g.: $X_4$) instead of removing the other $X_2$ move.

In any case, it is evident that there is need for a board that holds more information than the classical board. To do this, a new structure is made called the qtCell, and, the 3-by-3 qtBoard will consist of 9 of these qtCells. The qtCell struct hold the following member variables:

- A vector of turns (integers)
- A string
- The percentage in integer of Player 1
- The percentage in intger of Player 2
- the cell number

### 3.3.1 The Random Agent

The random agent is the agent which plays their moves randomly. There is no other reasoning behind its plays, that is, other than being chosen randomly. There are two main reasons for the creation of a random agent. One being the fact that it will provide the ability to construct a Monte Carlo Search algorithm with it, as, a Monte Carlo Search's playout is played randomly after the first initial move. The random agent, also, will provide a rough benchmark on the two main search algorithms, namely, (expecti-)minimax and Monte Carlo Search; and their effectiveness.

### 3.3.2 The Monte Carlo Agent

The Monte Carlo Agent, as the name suggests, is the agent that will play the game with a Monte Carlo Search. A variable that passes with each game is the *playouts* variable. Through this, the amount of playouts can be adjusted that the agent will simulate for each possible move.

### 3.3.3 The (Expecti-)Minimax Agent

The (expecti-)minimax agent is the agent that will play the game with either the expecti-minimax or the minimax. Specifically, the agent will exclusively use the alpha-beta minimax search algorithm if it is the classic Tic-Tac-Toe, and, exclusively use the expecti-minimax variant if the game mode is anything other than the classic Tic-Tac-Toe. That is because, as explained before, the Quantum Tic-Tac-Toe introduces chance and luck into the game. And, thus, requiring the necessity to take account for these probabilistic mechanics. In this program, a chance node represents a node, in which, measurement has taken place. The reasoning behind this is that because it is exactly in this state, that the board turns from a state of probabilities into a state of certainties; and, from these state of certainties it branches, again, into state of probabilities due to superposition moves.

One of the vital functions in the minimax algorithm is the one that determines the children nodes of a chance node. As evaluating a quantum board relies on chance, and, thus, can not manually determine every single possible classical board of a quantum board. This crucial function is called *determineChild* in the program, and, looks through the board and ascertains in which positions a turn can be found in recursively; and, converts it into an element of a sequence. A sequence is, thus, effectively, a board made of classical moves in string form. If a turn is found at multiple positions (e.g.: superposition or entanglement), then the function will call itself the same amount of times; essentially, adding another sequence on top. By the time every turn has been evaluated once, the sequence will be saved. Afterwards, each of these sequences can be converted back into boards, and gets passed as children nodes of a chance node.

## 3.4 Heuristic

A heuristic is needed to evaluate the board in case it does not reach an appropriate leaf for the expecti-minimax algorithm. There are many different manners to evaluate a board that is filled with superposition moves and entanglements, which may end up confusing instead. Though, there is an important aspect of Quantum Tic-Tac-Toe to keep in mind, namely, that a player can only earn points (win) through a three-in-a-row consisting of *classical* moves only. As such, it is more sensible to evaluate the boards of *classical moves* that the board can result into. In other words, every possible classical board will be determined through the quantum board; each of which will get evaluated individually.

The classical boards will then be evaluated by looking at each three-in-a-row line. Namely, for each of these line the amount of moves from each player will be counted: the x_counter and the o_counter. Depending on these counters a value will be added to or subtracted from the evaluation of the classical board. The following scheme will be applied:

- if x_counter=2 and o_counter=0: +25

- else if x_counter=0 and o_counter=2: -25
- else if x_counter=2: +5
- else if o_counter=2: -5
- else if x_counter=3: +50
- else if o_counter=3: -50

The scheme essentially prioritizes on making sure that the minimax player has more likelihood to win, since boards with more potential for a three-in-a-row is evaluated higher than boards that do not. Once every derivations of the quantum board has been evaluated, the evaluations of these derivations will be summed up and divided by the amount of derivations. The following formula sums the heuristic up:

$$\text{Evaluation of qtBoard} = \frac{\text{Sum evaluation of derivations}}{\text{Amount of derivations of initial board}} \tag{1}$$

# 4 Experiments

Here, various experiments will be conducted and analyzed. Additionally, the program will be run on Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz processor.

## 4.1 Analyzing the Expecti-Minimax Search

For this experiment, the first move of the expecti-minimax will be analyzed. This is done by looking at the very first move of a new game in which player 1 is the expecti-minimax agent, the second player will be omitted as it is not important for this specific analysis. Here, the time will be taken as an average of 10 runs and denoted in seconds.

| **Depth** | **Nodes** | **Time(s)** |
|---|---|---|
| 2 | 46 | 0.003 |
| 3 | 338 | 0.060 |
| 4 | 1904 | 0.163 |
| 5 | 9749 | 1.583 |
| 6 | 42001 | 3.314 |

Table 1: Results of analyzing first move of expecti-minimax (Minimal Quantumness). Time is average of 10 runs.

| Depth | Nodes | Time(s) |
|---|---|---|
| 2 | 46 | 0.003 |
| 3 | 338 | 0.063 |
| 4 | 4488 | 0.809 |
| 5 | 53516 | 17.896 |
| 6 | 333475 | 90.520 |

Table 2: Results of analyzing first move of expecti-minimax (Moderate Quantumness). Time is average of 10 runs.

| Depth | Nodes | Time(s) |
|---|---|---|
| 2 | 46 | 0.003 |
| 3 | 249 | 0.064 |
| 4 | 3072 | 0.723 |
| 5 | 61749 | 22.568 |
| 6 | 209061 | 73.244 |

Table 3: Results of analyzing first move of expecti-minimax (High Quantumness). Time is average of 10 runs.

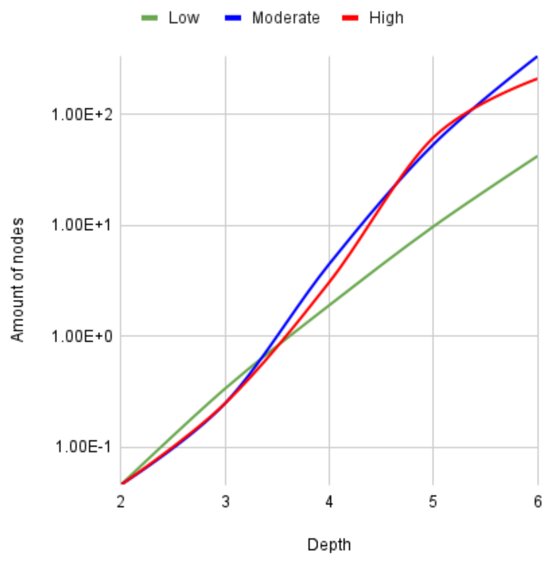

Figure 12: Amount of nodes with each increase in depth. Y-axis in logarithmic scale.

Figure 12 depicts the amount of nodes of Table 1, Table 2, and Table 3 into one graph. Here, it is evident that the amount of nodes increase exponentially with each increase in depth for the low quantumness. The moderate and high quantumness does not seem to strictly increase exponentially, as they are not a straight line in the graph; this could be due to pruning. Moreover, it appears that the amount of nodes do not necessarily increase between the same depths of different quantumness level. For example, depth 4 of moderate quantumness has a higher amount of nodes than depth 4 of high quantumness. Yet, for the depth 5 the amount of nodes of high quantumness is higher than the amount of nodes of moderate quantumness. Depth 6 seems to, again, follow the same trend as depth 4. One could reason that the amount of nodes in a depth would, in contrast to the graph, increase with each quantumness level; this, due to the increasing amount of possible moves that opens up with each increase in quantumness. Though, these anomalies could also be the result of alpha-beta pruning; which prunes more nodes at certain quantumness depths.

Table 1, Table 2, and Table 3 also illustrates a rough expectation on the amount of computation time needed for an experiment. For instance, an experiment with 100 playthroughs in high quantumness and with a depth of 6, would take at worst at least around 2 hours due to the very first move alone.

## 4.2 Agent vs Agent

The experiments will be conducted between the different agents. For every experiment the following variables will be analyzed:

- Amount of games played: 100
- Playout (for Monte Carlo): 100, 200, 300
- Depth (for expecti-minimax): 2, 3, 4
- Game mode: Low Quantumness, Moderate Quantumness, High Quantumness

Furthermore, if the experiment is between the Monte Carlo agent and the minimax agent, distinction will be made between which of the agent is player 1 (X) and which of the agent is player 2 (O). Player 2 will always be the random agent, if the experiment includes one. The results that will mainly be looked at are the following:

- Player 1 win
- Player 2 win
- Draw win: both players have a 3-in-a-row
- Draw lose: neither players have a 3-in-a-row

These results will be denoted in the amount of games that applies to its description of the 100 games in an experiment.

### 4.2.1 Monte Carlo Agent vs Random Agent

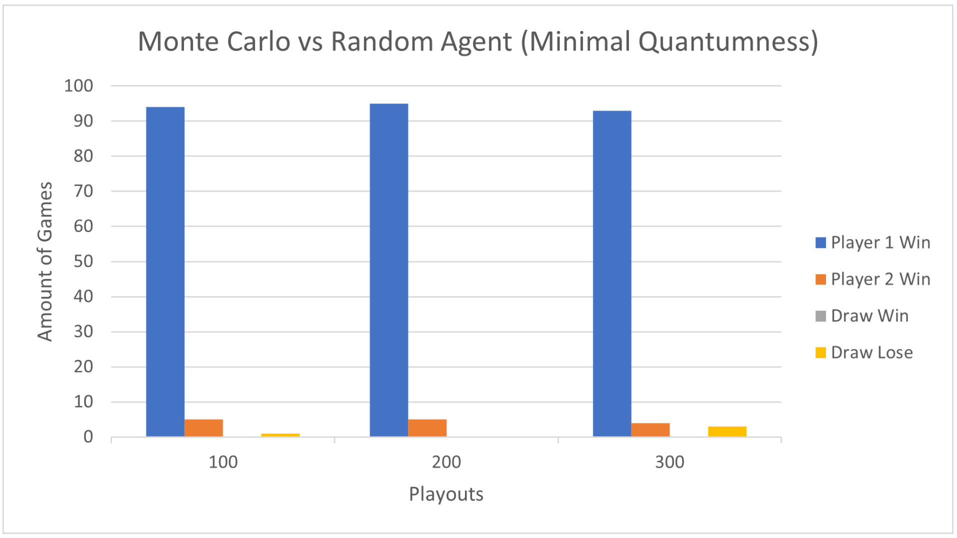

Figure 13: Monte Carlo agent (player 1) vs random agent (player 2) in Low Quantumness.

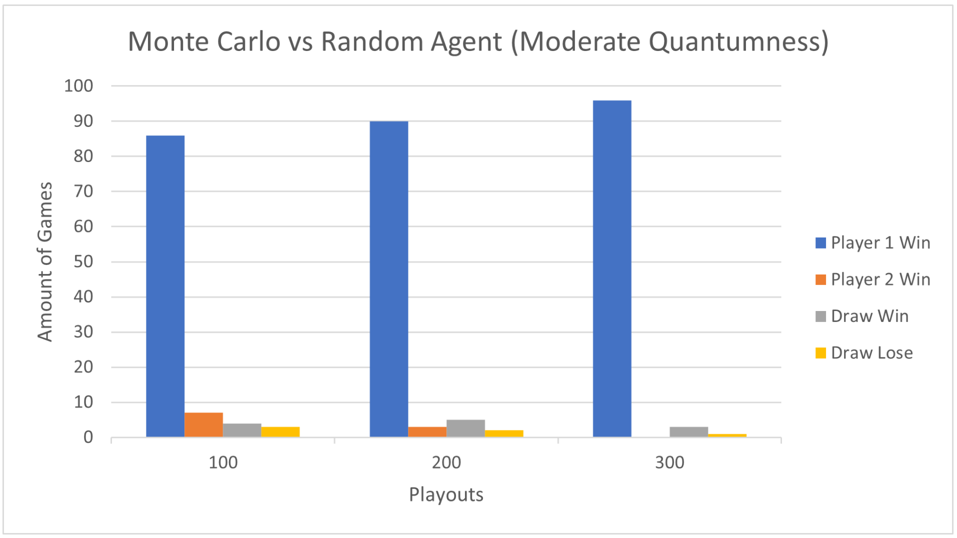

Figure 14: Monte Carlo agent (player 1) vs random agent (player 2) in Moderate Quantumness.

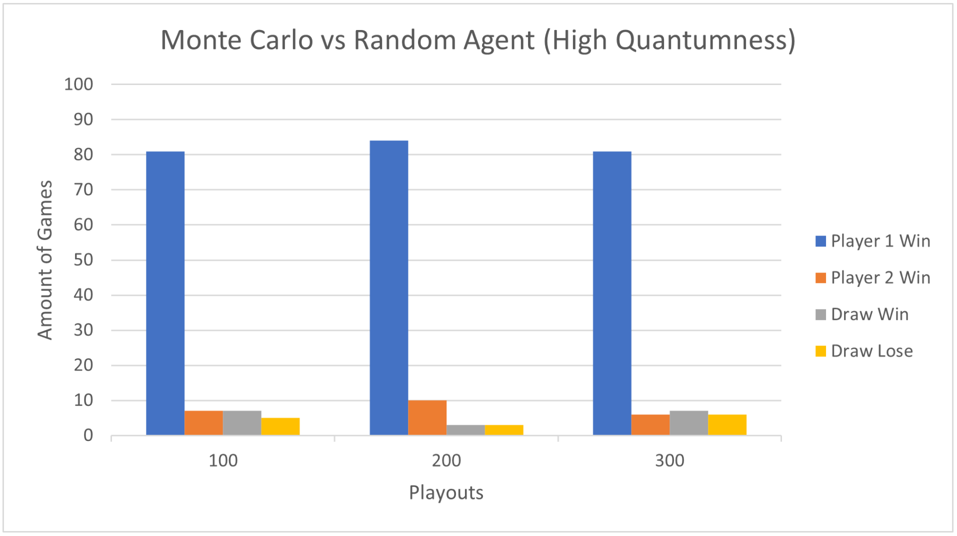

Figure 15: Monte Carlo agent (player 1) vs random agent (player 2) in High Quantumness.

Figure 15 illustrates the performance of the Monte Carlo agent, which, performs well across the board. There, however, does not seem to be much improvement with each increase in the amount of playouts in every quantumness level, as the amount of games player 1 wins is roughly the same. The graph of moderate quantumness does seem to imply as if the steps in playouts do gradually improve the performance of the agent; however, that might as well just be an anomaly of the three, rather than an indication of such.

Notably, the run-time for these experiments varies from 122 seconds to 512 seconds. The variable that increases the run-time the most being the amount of playouts of the experiment.

#### 4.2.2 Expecti-Minimax Agent vs Random Agent

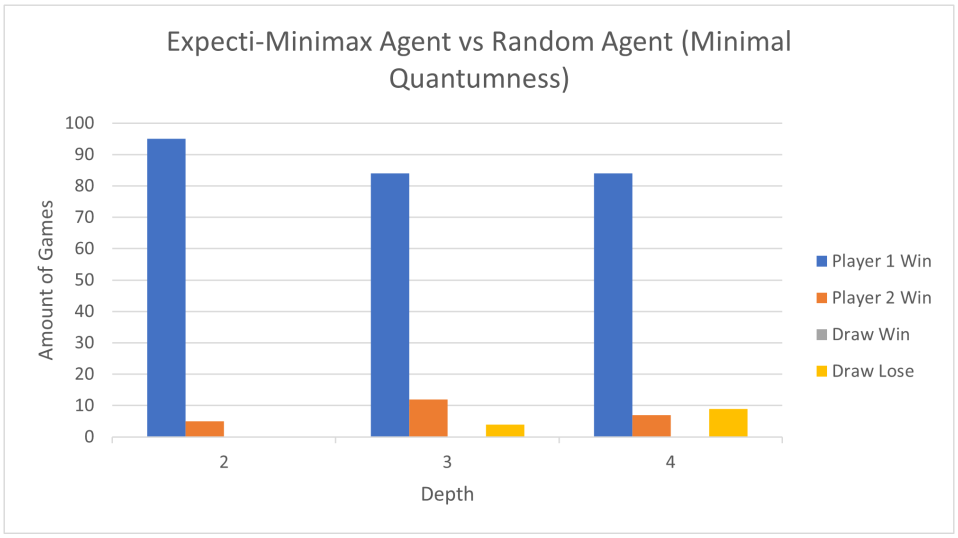

Figure 16: Expecti-minimax (player 1) vs random agent (player 2) in Low Quantumness.

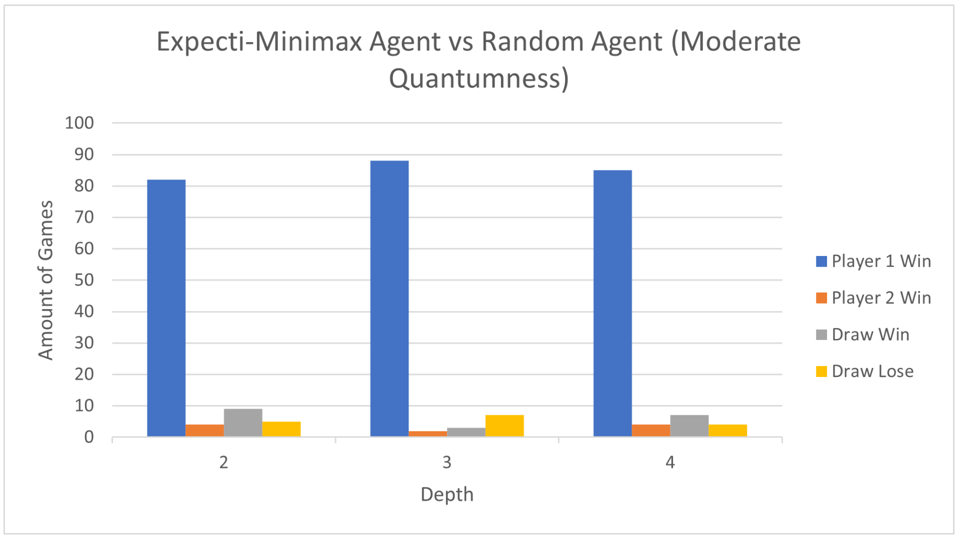

Figure 17: Expecti-minimax (player 1) vs random agent (player 2) in Moderate Quantumness.

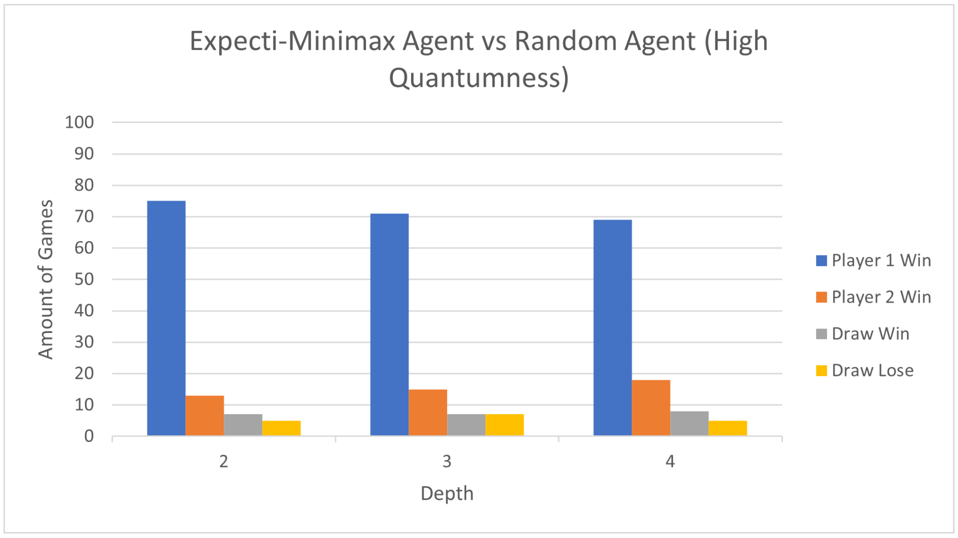

Figure 18: Expecti-minimax (player 1) vs random agent (player 2) in High Quantumness.

Figure 16, Figure 17, and Figure 18 reiterates the same findings as Section 4.2.1, as the increase in depth does not necessarily correlate to better results of the agent. Notably, the performance of the expecti-minimax agent in high quantumness seems to do a bit worse, compared to the other two quantum levels. Though, overall, the expecti-minimax agent seems to perform well against the random agent.

The run-time of these experiments varies from 0.5 seconds to 217 seconds. The variable that increases the run-time the most being the depth of the experiment.

### 4.2.3 Expecti-Minimax Agent vs Monte Carlo Agent

Various experiments have been conducted with every combination between the variables. Overall, it seems that the findings of Section 4.2.1 also applies here. As such, the amount of playouts does not change the results of these specific experiments by a notable amount, and, especially, does not change the *winner* between the agents; of which, is what the research paper is ultimately looking for. For this reason, in an attempt to lessen the amount of redundancy and improve clarity, only the results of 100 playouts will be depicted.

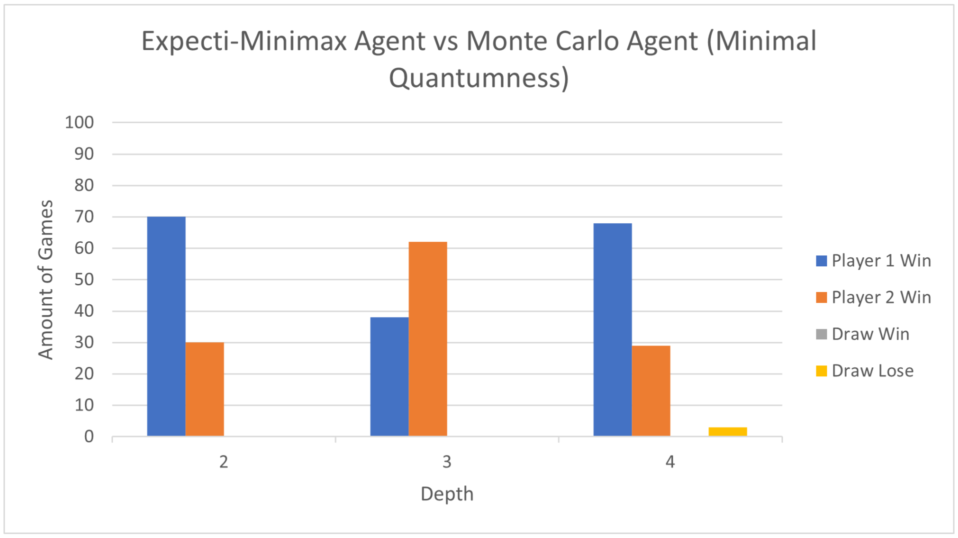

Figure 19: Expecti-minimax (player 1) vs Monte Carlo agent (player 2) with 100 playouts in Low Quantumness.

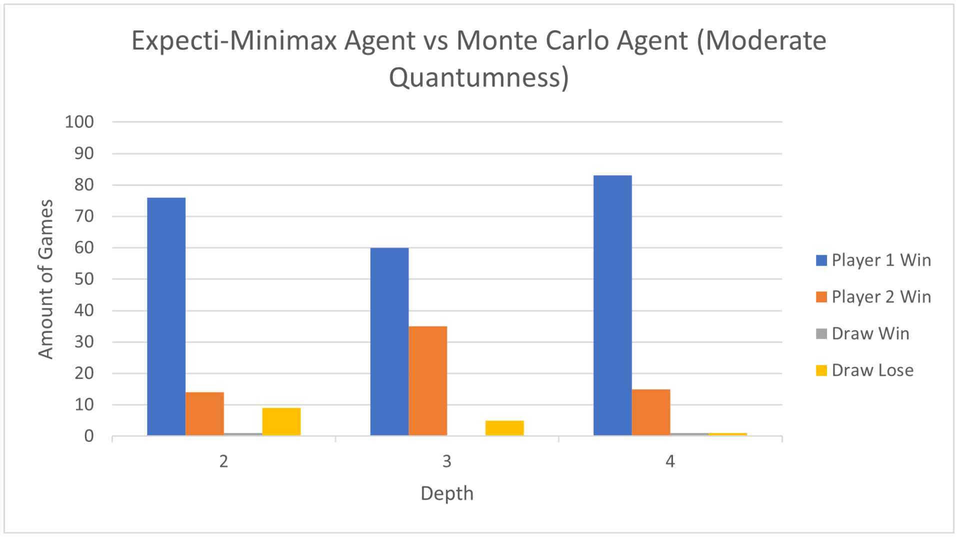

Figure 20: Expecti-minimax (player 1) vs Monte Carlo agent (player 2) with 100 playouts in Moderate Quantumness.

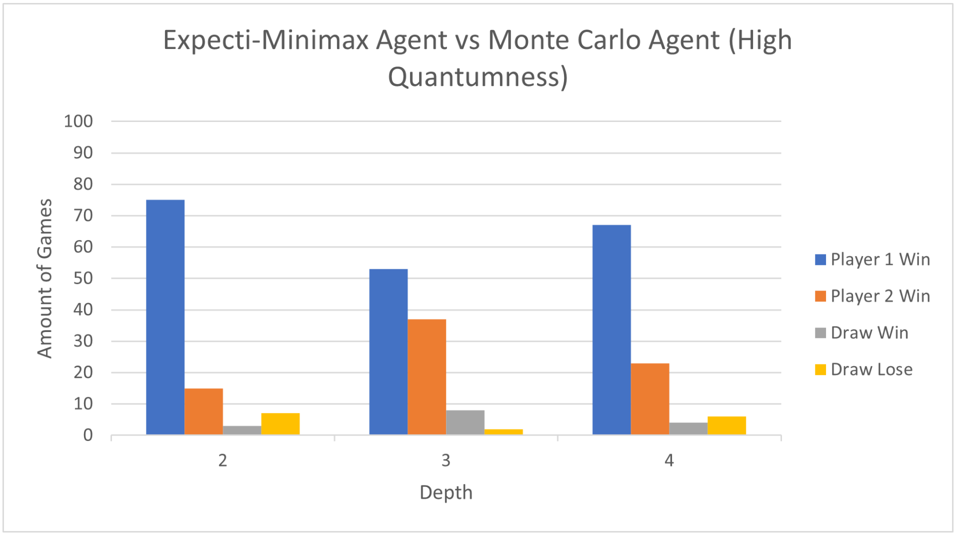

Figure 21: Expecti-minimax (player 1) vs Monte Carlo agent (player 2) with 100 playouts in High Quantumness.

Figure 19, Figure 20, and Figure 21 depicts the results of the experiments with 100 games, 100 playouts, and the different depths. Herein, it can be observed that the expecti-minimax agent (player 1) consistently wins over the Monte Carlo agent throughout every quantumness level. This, with the exception to the depth 3 of minimal quantumness, where the Monte Carlo agent convincingly beats the expecti-minimax agent. Another interesting observation to be made is that depth 3 performs worse across the quantum levels. However, depth 4 for every quantum level performs adequately again, in comparison to depth 3. As such, a greater depth does not necessarily signify a worse performance for the expecti-minimax agent.

One could conclude from Figure 19, Figure 20, and Figure 21 that the expecti-minimax agent performs on average better than the Monte Carlo agent. This, however, does not seem to be the case. The following experiment shows why.

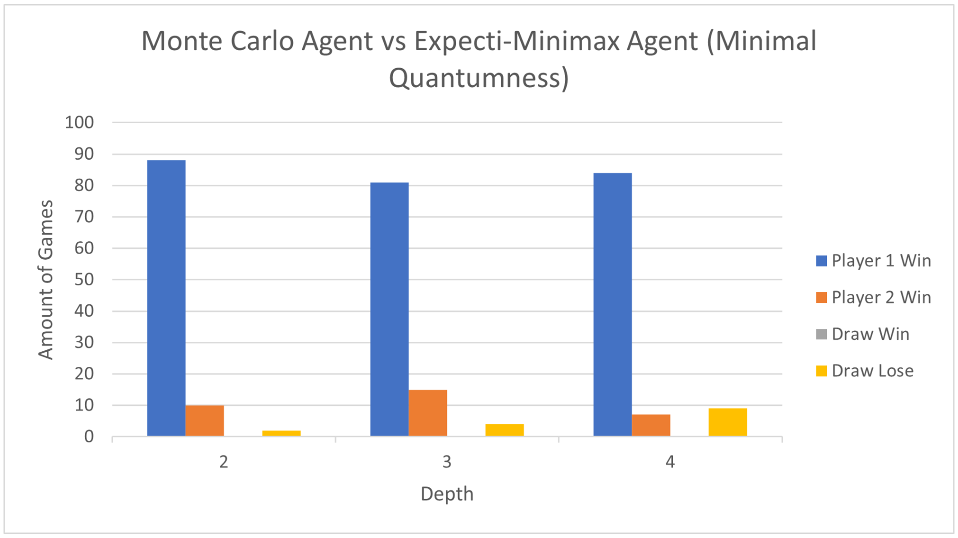

Figure 22: Monte Carlo agent (player 1) vs expecti-minimax agent (player 2) with 100 playouts in Low Quantumness.

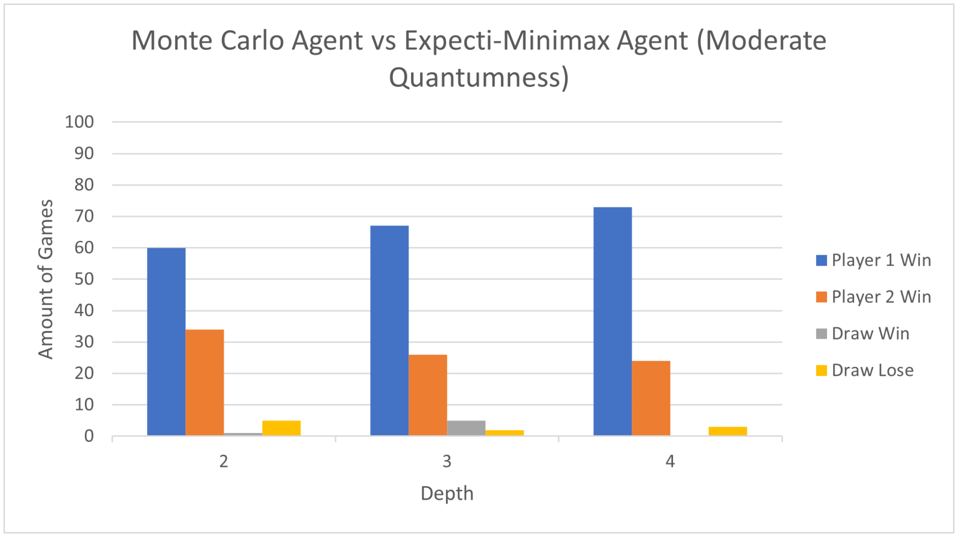

Figure 23: Monte Carlo agent (player 1) vs expecti-minimax agent (player 2) with 100 playouts in Moderate Quantumness.

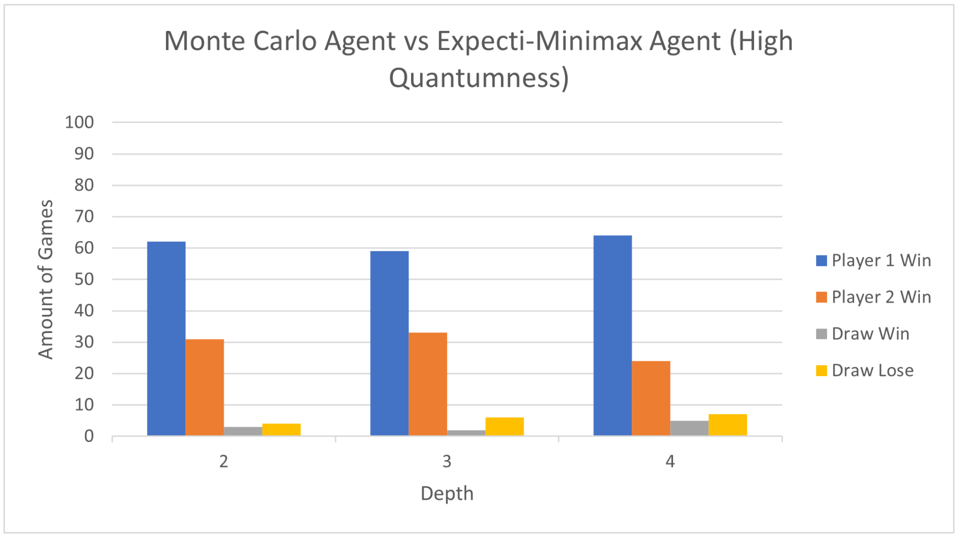

Figure 24: Monte Carlo agent (player 1) vs expecti-minimax agent (player 2) with 100 playouts in High Quantumness.

Figure 22, Figure 23, and Figure 24 are the same exact experiments as in Figure 19, Figure 20, and Figure 21, with the exception that the Monte Carlo agent is now player 1, and the expecti-minimax agent is player 2. Here, it can be observed that the Monte Carlo agent consistently outperforms the expecti-minimax agent; by winning more games than the expecti-minimax agent across every depth and quantum level. Another interesting observation that can be made is that the performance of Monte Carlo agent (player 1) decreases as the quantumness increases above the minimal quantumness. This can be seen by looking at the graph of minimal quantumness where the amount of games won by the Monte Carlo agent is at least 80. However, decreases down to 60 as the quantumness increases. Though, rather than saying that the performance of Monte Carlo agent decreases, it can also be said that the expecti-minimax agent performance increases with the quantum level; as it wins more games at higher quantumness compared to the minimal quantumness.

## 5 Conclusions and Further Research

This research paper aims to answer its research question "Does the expecti-minimax algorithm perform better than the Monte Carlo algorithm in Quantum Tic-Tac-Toe?". To this end, a program has been written in C++ with the algorithms as agents and various game modes as implemented by the Quantum TiqTaqToe implementation [vN]. Here, the agents are analyzed and are put against each other. The agents were first put against a random agent to establish a baseline of their respective performance.

The Monte Carlo agent performs well against the random agent across all quantum levels. Though, the amount of playouts of the Monte Carlo agent did not necessarily correlate to a better performance. In fact, the playouts seems to not affect the results in a notable amount at all. The expecti-minimax agent, too, performs well against the random agent across all quantum levels. However, its performance does seem to suffer against the random player as the quantumness increases.

Afterwards, the expecti-minimax agent and the Monte Carlo agent were put against each others. Notably, a distinction was made which agent was player 1 and player 2. Had this not been the case, then the wrong conclusions could, otherwise, have been made. As expecti-minimax agent would have convincingly perform better than the Monte Carlo agent, barring at one specific depth in one quantum level. Yet, by switching the players it had appeared that the Monte Carlo agent, who is now player 1, convincingly performs better than the expecti-minimax agent.

The conclusion that can be made is, thus, that the expecti-minimax algorithm does not necessarily perform better than the Monte Carlo algorithm for the used depths. As the winner seems to be directly correlated to which of the player is player 1. However, note that if the full depth were to be used, that the expecti-minimax should theoretically be better than the Monte Carlo algorithm; as, at that point, it would be brute-force equivalent. The results from the experiments seems to, further, imply that in every level of Quantum Tic-Tac-Toe, player 1 holds a greater advantage over player 2. Though, this is something that should be further researched as that was not explicitly in the scope of this research paper.

One of the limitations of this research paper is the lack of a real quantum computer. Instead, a random generator was made of use to replicate the workings of collapsing an entanglement. As such, further research into Quantum Tic-Tac-Toe with a real quantum computer could prove to be more insightful. Moreover, removing the limitation set on the amount of entanglements, depth, and playouts could also be interesting to research. Notably, the potential of an increase in performance of the algorithms by increasing either the depth or the playouts. However, this would increase the computation time significantly; making it infeasible with the current *consumer* computer.

## References

[Akl10] Selim G. Akl. Technical report no. 2010-568 on the importance of being quantum. 2010.

[Can19] Christopher Cantwell. Quantum chess: Developing a mathematical framework and design methodology for creating quantum games. 2019.

[RN20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach.* Pearson, 4 edition, 2020.

[vN] Evert van Nieuwenburg. Quantum tiqtaqtoe. https://beta.tiqtaqtoe.com/start. Accessed: 2023-05-22.