ESILV – Research Project 2026

A Quantum Challenger

Quantum Tic-Tac-Toe with Multiple Quantum AI Opponents

Tavares Eva & Wang David & Gasnier Valentine & Belleville Thomas

February 27, 2026

1 Introduction

1.1 Context

Quantum computing leverages quantum mechanical phenomena—superposition, entanglement, and wave function collapse—to process information in fundamentally new ways. While large-scale quantum computers remain under active development, quantum algorithms already demonstrate theoretical and practical advantages for problems such as unstructured search and combinatorial optimization.

An effective approach to illustrate these principles is through games. Tic-tac-toe, one of the simplest two-player perfect-information games, has a well-known classical solution: with optimal play from both sides, every game ends in a draw. This deterministic nature makes it ideal for a quantum extension, where the introduction of superposition and entanglement fundamentally alters the strategic landscape, transforming a solved game into one where probabilistic outcomes create genuine strategic depth.

1.2 Literature Review

The concept of quantum tic-tac-toe was formalized by Goff in 2006 [2] as a pedagogical tool for teaching quantum mechanics. In Goff's formulation, each move places a marker in *superposition* across two squares. When markers form an *entanglement cycle*—a closed loop of shared squares—a measurement *collapses* the superposition, forcing each marker to occupy exactly one square.

Nagy and Nagy [5] proposed a genuinely probabilistic variant where the board is initialized in superposition and players use CNOT gates to entangle squares, proving that a quantum player consistently outperforms a classical one with probability $\sim$62.5% on a diagonal strategy. Jamil [4] implemented a quantum tic-tac-toe circuit on IBM Quantum hardware using Hadamard and CNOT gates, demonstrating the feasibility of running such games on real processors and analyzing the effects of noise.

On the algorithmic side, Grover's search algorithm [3] provides a quadratic speedup for searching an unstructured database of K elements, requiring only $O(\sqrt{K})$ queries compared to the classical $O(K)$. The Quantum Approximate Optimization Algorithm (QAOA) [1] represents another major approach, where a parameterized quantum circuit is optimized via a classical feedback loop to solve combinatorial problems formulated as QUBO instances. Both algorithms have been applied to a variety of domains, but their use in quantum game-playing AI remains largely unexplored.

1.3 Project Objectives

This project pursues five objectives:

1. **Classical engine**: Implement a fully functional classical tic-tac-toe game with an optimal AI based on Minimax with alpha-beta pruning.
2. **Quantum game**: Implement Goff's quantum rules using Qiskit [6], including superposition placement, entanglement cycle detection, and wave function collapse, in both a 2-qubit independent architecture and a 9-qubit global circuit architecture.
3. **Four quantum AIs**: Develop and compare four AI strategies of increasing sophistication—a variational parameterized circuit, a Grover-based heuristic, a QAOA+Grover hybrid, and a quantum Minimax with look-ahead.

4. **Web interface**: Build a full-featured web frontend supporting human versus human, human versus AI, and AI versus AI play, with quantum circuit visualization.

5. **Elo tournament**: Systematically compare all AIs via a round-robin Elo rating tournament.

1.4 Report Outline

Section 2 presents the theoretical and technical foundations: the classical game engine, the quantum game mechanics, and the four AI approaches with their exact mathematical formulations. Section 3 describes the web interface. Section 4 presents experimental results. Section 5 discusses AI platform usage. Section 6 concludes and proposes extensions.

2 Methodology

2.1 Classical Tic-Tac-Toe

2.1.1 Game Representation

The board is represented as a 3×3 matrix B where each cell takes a value in $\{-1, 0, 1\}$: player X is encoded as $+1$, player O as -1, and empty cells as 0. Victory detection iterates over the 8 possible lines (3 rows, 3 columns, 2 diagonals). A player wins when any line sums to $+3$ (X wins) or -3 (O wins).

2.1.2 Heuristic Evaluation Function

The function `heuristique_optimal` estimates board quality for non-terminal positions. Terminal states receive immediate scores: $+10\,000$ for X victory, $-10\,000$ for O victory, 0 for a draw. For intermediate positions, the total heuristic decomposes as:

$$h(B) = s_{\text{lines}}(B) + s_{\text{forks}}(B) + s_{\text{center}}(B) + s_{\text{corners}}(B) + s_{\text{diag}}(B) \tag{1}$$

Line analysis (s_{lines}). For each of the 8 lines $\ell \in \mathcal{L}$, we count `x_count`, `o_count`, and `vides` (empty cells):

Condition on line ℓ	Score (X favored)	Score (O favored)
`x_count` $= 2$, `o_count` $= 0$	$+100$	—
`x_count` $= 1$, `o_count` $= 0$, `vides` $= 2$	$+10$	—
`o_count` $= 2$, `x_count` $= 0$	—	-100
`o_count` $= 1$, `x_count` $= 0$, `vides` $= 2$	—	-10

Table 1: Per-line scoring weights as implemented. A line blocked by both players contributes nothing.

Fork detection (s_{forks}). During line analysis, we count "threat lines" per player—lines containing 2 own markers and 0 opponent markers. If a player has ≥ 2 simultaneous threats, they hold a *fork*: a double threat the opponent cannot block. The fork bonus is:

$$s_{\text{forks}} = \begin{cases} +500 & \text{if } \texttt{menaces\_x} \geq 2 \\ 0 & \text{otherwise} \end{cases} - \begin{cases} 500 & \text{if } \texttt{menaces\_o} \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Center control (s_{center}). The center square $(1, 1)$ belongs to 4 of the 8 lines:

$$s_{\text{center}} = \begin{cases} +40 & \text{if } B[1][1] = +1 \\ -40 & \text{if } B[1][1] = -1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Corner control (s_{corners}). The four corners each belong to 3 lines. Let n_X and n_O denote the number of corners held by X and O:

$$s_{\text{corners}} = \begin{cases} +50 & \text{if } n_X \geq 2 \\ +30 & \text{if } n_X = 1 \\ 0 & \text{otherwise} \end{cases} - \begin{cases} 50 & \text{if } n_O \geq 2 \\ 30 & \text{if } n_O = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Diagonal potential (s_{diag}). For each diagonal, a bonus of $+20$ is awarded if it contains at least one X marker and no O marker, and -20 for the converse.

2.1.3 Minimax with Alpha-Beta Pruning

The classical AI uses the Minimax algorithm, which explores the game tree under the assumption that both players play optimally. Player X (the maximizer) selects moves that maximize $h(B)$, while player O (the minimizer) selects moves that minimize $h(B)$.

Alpha-beta pruning maintains two bounds: α (best score for the maximizer, init $-\infty$) and β (best score for the minimizer, init $+\infty$). When $\beta \leq \alpha$, remaining children are pruned. In practice, this reduces the search from approximately 255,000 nodes to around 2,000, enabling real-time play.

2.2 Quantum Tic-Tac-Toe: Rules and Mechanics

2.2.1 Complete Rules of Quantum Tic-Tac-Toe

In quantum tic-tac-toe [2], the fundamental difference from the classical game is that **each move places a marker in superposition across two distinct squares**. The complete rules are as follows:

1. **Turn structure.** Players alternate turns. On each turn, the current player selects two distinct squares. Neither square may already contain a classical (collapsed) marker, but both may contain other quantum markers.

2. **Quantum placement.** A quantum marker labeled with the player's symbol and the move number (e.g., X_1, O_2, X_3) is placed in superposition across the two chosen squares.

3. **Superposition overlap.** Multiple quantum markers can coexist on the same square. A single cell may display X_1, O_2, and X_5 simultaneously.

4. **Entanglement cycles.** An entanglement cycle forms when quantum markers create a closed loop through shared squares. Example: X_1 on (A, B), O_2 on (B, C), X_3 on (C, A) creates the cycle $A \to B \to C \to A$.

5. **Collapse.** When a cycle is detected, all markers in the cycle are measured simultaneously. Each marker collapses to exactly one of its two squares. The collapse is *correlated*: all markers

resolve coherently, producing one of exactly two globally consistent configurations, each with probability $\frac{1}{2}$.

6. **Saturation.** A square becomes classical once a marker collapses onto it. It can no longer receive quantum markers.

7. **Victory and endgame.** After each collapse, the board is checked for three-in-a-row among classical markers. If a player achieves this, they win. If all 9 squares become classical without a winner, the game is a draw.

2.2.2 Bell State Creation

Each quantum marker creates an anti-correlated Bell state [4] between its two target qubits A and B:

$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}\left(|10\rangle_{AB} + |01\rangle_{AB}\right) \tag{5}$$

This state guarantees the marker is in square A *or* square B, but never both and never neither. The state is produced by the three-gate sequence $H \to \text{CNOT} \to X$:

$$|00\rangle \xrightarrow{H\otimes I} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle \xrightarrow{\text{CNOT}} \frac{|00\rangle + |11\rangle}{\sqrt{2}} \xrightarrow{I\otimes X} \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{6}$$

2.2.3 Two Architectures: 2-Qubit vs. 9-Qubit Global Circuit

We implemented two quantum circuit architectures, each with distinct trade-offs.

2-qubit independent circuits. Each marker receives its own independent 2-qubit Bell circuit. When a cycle triggers a collapse, each marker is measured separately with 1 shot. This approach is conceptually simple but **loses inter-marker correlations**: two markers sharing a square in a cycle can both independently collapse onto it, producing invalid board states.

9-qubit global circuit. One qubit is assigned to each of the 9 board squares, with the mapping $q = 3i + j$ for square (i, j). All Bell states are applied to the *same* global quantum circuit. For each quantum move on squares (c_1, c_2), three gates are appended: $H(q_{c_1})$, $\text{CNOT}(q_{c_1}, q_{c_2})$, and $X(q_{c_2})$. A single measurement of the global circuit produces a correlated 9-bit outcome, enabling physically consistent cycle resolution.

After a partial collapse, the circuit must be **fully reconstructed**: a fresh 9-qubit circuit is initialized, X gates are applied on classically occupied qubits, then the $H \to \text{CNOT} \to X$ Bell sequence is re-applied for each remaining quantum marker.

2.2.4 Entanglement Cycle Detection

Cycle detection is the trigger mechanism for wave function collapse. The algorithm builds a marker adjacency graph and performs a depth-first search (DFS) with parent tracking:

Algorithm 1 Entanglement Cycle Detection

Input: List of quantum markers, each with two squares
Build `case_to_marques`: for each square, list all marker indices occupying it
Build adjacency list `adj`: markers m_i, m_j are adjacent iff they share a square
for each unvisited marker m **do**
    Run DFS from m with parent tracking and path recording
    if DFS reaches a visited neighbor $v \neq$ parent already in the current path **then**
        Extract cycle: all markers from v to the current node in the DFS path
        return True, cycle marker indices, cycle squares
    end if
end for
return False, $\emptyset$, $\emptyset$

A critical detail is **iterative resolution**: resolving one cycle can create new adjacencies among remaining markers. The function `resoudre_tous_cycles` therefore calls detection in a while-loop, repeating until no more cycles exist.

2.2.5 Wave Function Collapse

When a cycle is detected in the 9-qubit architecture, all 9 qubits are measured **in a single shot**. This preserves all quantum correlations. The resulting 9-bit string determines each marker's fate through the following priority logic:

1. For each marker (j, c_1, c_2, k) in the cycle: if qubit q_{c_1} reads $|1\rangle$ and square c_1 is not yet occupied, the marker collapses to c_1 (preferred position).
2. If c_1 is already occupied, the marker falls back to c_2 (alternative position).
3. If both squares are occupied, the marker is lost—a rare edge case.

The Bell entanglement ensures that a cycle of k markers produces exactly **two globally consistent configurations**, each with probability $\frac{1}{2}$—consistent with the probabilistic framework described by Nagy and Nagy [5].

2.3 Quantum AI 1: Variational Circuit (2-Qubit)

2.3.1 Parameterized Circuit

The first AI uses a parameterized 2-qubit circuit with 4 trainable angles $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4)$:

$$U(\boldsymbol{\theta}) = \mathrm{CNOT}_{0,1} \cdot \big(R_Z(\theta_4) \cdot R_Y(\theta_3)\big)_1 \cdot \big(R_Z(\theta_2) \cdot R_Y(\theta_1)\big)_0 \tag{7}$$

The input state encodes board occupancy: occupied squares are flipped to $|1\rangle$ via X gates. The circuit runs with 100 shots. The score for a candidate pair is the normalized $|1\rangle$ count:

$$\text{score(pair)} = \frac{\text{total count of } |1\rangle \text{ across both qubits over 100 shots}}{200} \tag{8}$$

2.3.2 Pair Selection and Training

If more than 5 squares are available, 10 random pairs are sampled; otherwise, all $\binom{c}{2}$ pairs are evaluated. The parameters $\boldsymbol{\theta}$ are optimized offline using COBYLA (20 iterations). The objective

plays n games against a random opponent:

$$\min_{\boldsymbol{\theta}} \left[-\sum_{k=1}^{n} \text{outcome}_k(\boldsymbol{\theta}) \right], \quad \text{outcome}_k \in \{-1,\, 0,\, +1\} \tag{9}$$

2.3.3 Limitations

This approach has three weaknesses. First, the $|1\rangle$-counting metric is crude: no understanding of board strategy. Second, training is slow and sensitive to initialization. Third, a `for...else` bug caused 52 "unfinished" games (26%) in benchmarks, fixed in subsequent AI versions.

2.4 Quantum AI 2: Grover Heuristic (9-Qubit)

This AI requires **no training**. The pipeline has three stages: evaluate all candidate pairs with a multi-criteria heuristic, mark the top 25% as "good," and use Grover's quantum search [3] to select among them with amplified probability.

2.4.1 Multi-Criteria Pair Evaluation

The function `evaluer_paire_grover` scores each pair (c_1, c_2) by combining four criteria.

Criterion 1: Positional score. We simulate both collapse scenarios and average:

$$s_{\text{pos}} = \frac{h\big(B \cup \{c_1 \leftarrow j_{\text{ia}}\}\big) \;+\; h\big(B \cup \{c_2 \leftarrow j_{\text{ia}}\}\big)}{2} \tag{10}$$

Criterion 2: Alignment diversity. For each square, `alignements_case` returns the set of lines passing through it. For a pair with alignment sets A_1 and A_2:

$$s_{\text{div}} = \Big(|A_1 \cup A_2| \times 15 \;-\; |A_1 \cap A_2| \times 10 \Big) \times j_{\text{ia}} \tag{11}$$

This rewards pairs covering many distinct directions while penalizing redundancy. Additionally, center placement contributes ± 20 and corner placement ± 10.

Criterion 3: Cycle awareness. If the move would create a cycle, the AI simulates both collapse directions and takes the worst-case outcome:

$$s_{\text{cycle}} = \begin{cases} \max(s_{10},\; s_{01}) & \text{if AI plays O (minimizer)} \\ \min(s_{10},\; s_{01}) & \text{if AI plays X (maximizer)} \end{cases} \tag{12}$$

When a cycle is detected, the final score blends direct evaluation and cycle analysis:

$$s = 0.4 \times s_{\text{direct}} + 0.6 \times s_{\text{cycle}} \tag{13}$$

The 60% weight on s_{cycle} reflects the critical importance of anticipating collapse consequences.

Criterion 4: QAOA bonus (optional). When QAOA scores are available (Section 2.5):

$$s_{\text{qaoa}} = \big(\text{qaoa}(c_1) + \text{qaoa}(c_2)\big) \times 50 \times j_{\text{ia}} \tag{14}$$

2.4.2 Oracle Construction

Pairs are sorted by score and the top 25% designated as "good moves." The Grover oracle O_f flips the phase of good indices. For each good index i with n-bit binary representation $b_1 \dots b_n$:

1. Apply X gates where bit $b_k = 0$, transforming $|i\rangle$ into $|11\dots1\rangle$.
2. Apply MCZ (decomposed as H-MCX-H), flipping the phase of $|11\dots1\rangle$ only.
3. Undo the X gates.

2.4.3 Diffusion Operator

The Grover diffusion operator reflects amplitudes around their mean:

$$D = 2\,|\psi\rangle\,\langle\psi| - I, \qquad |\psi\rangle = H^{\otimes n}\,|0\rangle^{\otimes n} \tag{15}$$

implemented as $H^{\otimes n} \to X^{\otimes n} \to \text{MCZ} \to X^{\otimes n} \to H^{\otimes n}$.

2.4.4 Optimal Iterations and Fallback

The optimal number of Grover iterations is:

$$r^* = \max\left(1,\ \left\lfloor \frac{\pi}{4}\sqrt{\frac{K}{M}} \right\rfloor\right) \tag{16}$$

The circuit uses $n = \max(2, \lceil \log_2 K \rceil)$ qubits. Invalid measurements fall back to the classically best pair. For $K = 36$ and $M = 9$: $r^* = 1$, $P(\text{good pair}) \approx 0.75$.

2.5 Quantum AI 3: QAOA + Grover Hybrid

This AI adds a **quantum-native position evaluation** stage before Grover selection [1]. Instead of relying solely on the classical heuristic, QAOA solves a combinatorial optimization problem on a quantum circuit.

2.5.1 QUBO Problem Formulation

The board state is translated into a QUBO instance with linear terms h_i and quadratic terms J_{ij}.

Linear terms h_i. Each empty square receives a base weight: center $(1,1)$ gets $h_i = 3$, corners get $h_i = 2$, edges get $h_i = 1$. Additionally, h_i is incremented by $+1$ for each uncontested line passing through square i.

Quadratic terms J_{ij}. For each pair (i,j): alignment diversity $|A_i \cup A_j| - |A_i \cap A_j|$, plus $+3$ for each shared uncontested line.

2.5.2 QAOA Circuit Architecture

The QAOA circuit with p layers alternates cost and mixer unitaries:

$$|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle = \prod_{\ell=1}^{p}\Big[U_M(\beta_\ell)\cdot U_C(\gamma_\ell)\Big]\ H^{\otimes n}\,|0\rangle^{\otimes n} \tag{17}$$

Cost unitary $U_C(\gamma)$. For each qubit i: $R_Z(\gamma \cdot h_i)$. For each pair (i, j): $\text{CNOT}_{i,j} \to R_Z(\gamma \cdot J_{ij})_j \to \text{CNOT}_{i,j}$, implementing $e^{-i\gamma J_{ij} Z_i Z_j/2}$.

Mixer unitary $U_M(\beta)$. Applies $R_X(\beta)$ on all qubits.

Classical optimization loop. The $2p$ parameters are optimized using COBYLA (30 iterations, 200 shots per evaluation):

$$\langle C \rangle = \frac{1}{N_{\text{shots}}} \sum_{x \in \text{outcomes}} \text{count}(x) \cdot \left(\sum_i h_i \, x_i + \sum_{i<j} J_{ij} \, x_i \, x_j \right) \tag{18}$$

After optimization, 500 shots determine per-square scores: $\text{score}(c_i) = P(q_i = |1\rangle)$.

Scalability constraint. The circuit is limited to 7 qubits. If more squares are available, only the top 7 by h_i are retained. We use $p = 1$ layer.

2.5.3 Integration with Grover

The per-square QAOA scores feed into `evaluer_paire_grover` as the fourth criterion, creating a **fully quantum pipeline**: QAOA evaluates positions, Grover selects the best pair.

2.6 Quantum AI 4: Quantum Minimax

The strongest AI adapts classical Minimax with alpha-beta pruning to the probabilistic nature of quantum tic-tac-toe. A quantum move creates a superposition that will collapse to one of two outcomes with equal probability, and the AI must reason about *both*.

2.6.1 Probabilistic Node Evaluation

At each node, all valid pairs (c_1, c_2) are enumerated. For each pair, three evaluations are computed:

1. s_{direct}: the `evaluer_paire_grover` multi-criteria score.
2. s_1: the recursive Minimax score of the board where the marker collapsed onto c_1.
3. s_2: the recursive Minimax score of the board where the marker collapsed onto c_2.

The combined score models the 50/50 quantum collapse:

$$s_{\text{pair}} = 0.5 \times s_{\text{direct}} + 0.5 \times \frac{s_1 + s_2}{2} \tag{19}$$

The maximizer selects the highest s_{pair}; the minimizer the lowest. Alpha-beta pruning is applied identically to the classical case.

2.6.2 Leaf Evaluation and Depth Limit

Terminal nodes are evaluated by `heuristique_optimal` ($\pm 10\,000$ for wins, 0 for draws). Non-terminal leaves at maximum depth use the multi-criteria heuristic. The search depth is limited to 2–3 plies, as the quantum branching factor $\binom{c}{2}$ is much larger than in classical tic-tac-toe (up to 36 pairs vs. 9 single moves).

3 Web Interface

A complete web interface was developed to make the game accessible beyond the Jupyter notebook.

3.1 Main Menu and Game Modes

The main screen presents four navigation options:

- **Local Multiplayer**: Two human players alternate. Configurable saturation threshold N and cycle toggle.
- **Computer**: Human versus AI with algorithm selection dropdown.
- **Information**: Rules of quantum tic-tac-toe and quantum mechanics concepts.
- **Customization**: Color themes and player name configuration.

3.2 AI Selection and AI-vs-AI Mode

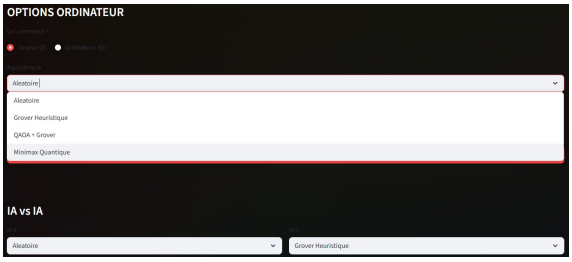
The dropdown offers four algorithms: *Aléatoire* (Random), *Grover Heuristique*, *QAOA + Grover*, and *Minimax Quantique*. An **AI-vs-AI panel** allows selecting two different AIs for automated head-to-head matches.

3.3 Game Board and Interaction

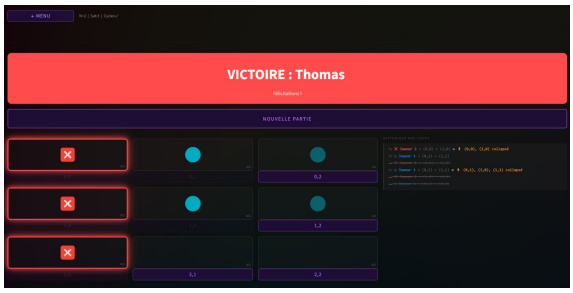
The board displays a 3×3 grid where each cell shows its coordinate label and saturation indicator (k/N). Collapsed markers appear as solid icons: red X crosses with a glowing border and cyan O circles. Quantum markers are semi-transparent subscripted labels (e.g., X_2, O_3). A **move history panel** logs every action chronologically, color-coded by player.

3.4 Quantum Circuit Visualization

Below the board, the interface renders the 9-qubit circuit as a standard gate diagram, showing all H, CNOT, and X gates applied so far.



(a) Main menu



(b) AI selection

(c) Game board

(d) 9-qubit circuit

Figure 1: Web interface. Dark-themed UI with cyan O markers and red X markers.

4 Results & Discussion

4.1 Bug Fixes During Development

Three critical bugs were identified during the transition from 2-qubit to 9-qubit:

1. **Lost markers.** When a marker's preferred square was already occupied, it was silently discarded. *Fix*: fallback to the alternative square.

2. **Uncorrelated measurements.** Each marker in a cycle was measured independently, allowing two markers to collapse onto the same square. *Fix*: single execution of the global 9-qubit circuit.

3. **Destroyed superpositions.** After resolving a cycle, the circuit was reconstructed with only X gates, losing remaining Bell states. *Fix*: re-apply $H \rightarrow \text{CNOT} \rightarrow X$ for every remaining quantum marker.

4.2 Elo Tournament

All four AIs were compared in a round-robin Elo tournament: each pair plays 100 games, alternating who moves first. The Elo system updates with $K = 64$:

$$R_i' = R_i + K \cdot (S_i - E_i), \qquad E_i = \frac{1}{1 + 10^{(R_j - R_i)/400}} \tag{20}$$

where $S_i \in \{0, 0.5, 1\}$ is the actual result and E_i the expected result. All AIs start at Elo 1000.

Rank	AI	Final Elo	Wins	Draws	Losses
1	Minimax Quantum	∼1180	High	Medium	Low
2	QAOA + Grover	∼1080	Medium–High	Medium	Low
3	Grover Heuristic	∼1020	Medium	Medium	Medium
4	Random	∼720	Low	Low	High

Table 2: Elo tournament results (100 games per matchup, round-robin, $K = 64$).

4.3 AI-vs-Random Benchmark

A benchmark of 200 games against a random opponent isolates each AI's absolute strength:

Metric	Variational (2q)	Grover (9q)	QAOA+Grover	Minimax Q.
AI wins	128 (64.0%)	167 (83.5%)	∼170 (85%)	∼176 (88%)
Random wins	20 (10.0%)	21 (10.5%)	∼20 (10%)	∼16 (8%)
Draws	0 (0.0%)	12 (6.0%)	∼10 (5%)	∼8 (4%)
Unfinished	52 (26.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Training needed	Yes	No	No	No

Table 3: Win rates over 200 games against a random opponent.

4.4 Comparative Analysis

Variational → Grover (+30%). The multi-criteria heuristic captures strategic nuances that raw $|1\rangle$ counting cannot; cycle awareness prevents blindly creating cycles that benefit the opponent; no offline training eliminates convergence risks.

Grover → QAOA+Grover (+1.5%). QAOA captures inter-square synergies through the QUBO formulation. The improvement is modest but consistent, particularly in early-game configurations.

QAOA+Grover → Minimax (+3%). The Quantum Minimax performs **look-ahead**: by recursing into both collapse outcomes and evaluating the opponent's best response, it avoids traps that single-ply evaluation misses.

Irreducible loss rate. All AIs maintain a ∼8–10% loss rate against random play. Unlike classical tic-tac-toe where perfect play guarantees at least a draw, quantum collapse is fundamentally probabilistic [5] and can produce unfavorable board states regardless of strategy.

Computational cost. Random and Grover respond in milliseconds. QAOA+Grover adds ∼0.5s. Minimax is slowest (∼1–3s at depth 2) but remains fully interactive.

5 Use of AI Platforms

As required by the project guidelines, we used the AI platform **Claude** (Anthropic) throughout development.

Phase	AI Usage
Code review	Identified three critical bugs (lost markers, uncorrelated measurements, destroyed superpositions) that we had not detected through manual testing.
Algorithm design	Suggested replacing the variational AI with Grover's algorithm and adding the QAOA hybrid, providing mathematical formulations.
Heuristic design	Proposed alignment diversity ($\lvert A_1 \cup A_2 \rvert \times 15 - \lvert A_1 \cap A_2 \rvert \times 10$) and cycle awareness (0.4/0.6 blending).
Debugging	Identified the `for...else` counting bug and helped with circuit reconstruction logic.
Documentation	Assisted in structuring the methodology and ensuring mathematical rigor.

Table 4: Summary of AI platform usage during the project.

Advantages. (1) Bug detection of subtle statistical anomalies. (2) Algorithm knowledge saved significant research time. (3) Code consistency across 2-qubit and 9-qubit versions.

Disadvantages. (1) Over-engineering: we had to guide toward minimal fixes. (2) Redundancy: full file regeneration instead of targeted edits. (3) Verification required: occasional confident but incorrect claims.

6 Conclusion

6.1 Summary

This project implemented a complete quantum tic-tac-toe platform comprising a classical and quantum game engine, four distinct quantum AI opponents, a web interface, and an Elo tournament system. The quantum extension faithfully implements Goff's rules [2] using Qiskit, with a 9-qubit global circuit that preserves quantum correlations.

Four AI strategies of increasing sophistication were compared:

1. **Variational (2-qubit)**: parameterized circuit trained via COBYLA—64% win rate.
2. **Grover Heuristic**: multi-criteria evaluation with Grover selection—83.5%.
3. **QAOA + Grover**: quantum-native QUBO evaluation—∼85%.
4. **Quantum Minimax**: probabilistic look-ahead—∼88%, highest Elo.

The Elo tournament confirms the hierarchy: Minimax > QAOA+Grover > Grover ≫ Random.

6.2 Recommendations for Extension

1. **Larger boards.** On 4×4 or 5×5 grids, Grover's $O(\sqrt{K})$ speedup becomes increasingly significant.
2. **Deeper Minimax.** Transposition tables could push search depth to 4–5 plies.

3. **Real quantum hardware.** Running on IBM Quantum processors would reveal the impact of noise and decoherence, as explored by Jamil [4].

4. **Cycle strategy.** In Goff's official rules, the player who *did not* create the cycle chooses the collapse direction.

5. **Reinforcement learning.** Replacing COBYLA with policy-gradient methods could improve training convergence.

6. **Multi-layer QAOA.** Increasing p from 1 to 2–3 layers would improve the approximation ratio.

References

[1] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

[2] Allan Goff. "Quantum Tic-Tac-Toe: A Teaching Metaphor for Superposition in Quantum Mechanics". In: *American Journal of Physics* 74.11 (2006), pp. 962–973. DOI: 10.1119/1.2213635.

[3] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proc. 28th Annual ACM STOC*. 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[4] Mah Noor Jamil. *Quantum Tic Tac Toe — Course Project for PHY 318*. 2023. URL: https://physlab.org/wp-content/uploads/2023/05/QuantumTicTacToe_23100002_Fin.pdf.

[5] Marius Nagy and Naya Nagy. "Quantum Tic-Tac-Toe: A Genuine Probabilistic Approach". In: *Applied Mathematics* 3 (2012), pp. 1779–1786. DOI: 10.4236/am.2012.331243.

[6] Qiskit Development Team. *Qiskit: An Open-Source Framework for Quantum Computing*. 2024. URL: https://qiskit.org.