**Artificial Intelligence – Assignment 03 – Task 03**

**Name:** Ayaan Khan

**Roll Number:** 22i-0832

**Section:** CS - K

**1. Overview of the Program**

This program implements Ultimate Tic Tac Toe using a PyQt6 GUI for visualization and interaction. The board is structured as a 3×3 grid of 3×3 subboards. The game rules require that a move in one subboard directs the opponent to the corresponding subboard. The game state is maintained by two primary constructs:

* **Main Board:** A list of small boards (each of 9 cells).
* **Small Board Winners:** A list holding the result (win for X, O, or draw) for each subboard.

The program sets up several AI agents (via a dictionary called AI\_CLASSES), each implementing a different search strategy to decide moves.

**2. AI Variants Implemented**

The implementation includes the following AI variants:

**a. Default – (d=4) / BestAI Variant**

* **Method:** Hybrid minimax with alpha–beta pruning.
* **Techniques Used:**  
  – Uses a depth-limited search (depth 4 by default).  
  – Optionally integrates an MRV (Minimum Remaining Values) heuristic in move ordering.
* **Characteristics:**  
  This variant leverages an aggressive pruning strategy via alpha–beta search. Its move ordering (which uses a simple count of empty cells on boards) indirectly serves as a form of constraint enforcement.

**b. Minimax (d=4) Variant**

* **Method:** Basic minimax search without alpha–beta pruning.
* **Techniques Used:**  
  – Recursively evaluates board states up to the depth limit.  
  – Uses the same evaluation function as BestAI.
* **Characteristics:**  
  Due to the lack of alpha–beta pruning, it expands more nodes, but its simpler structure sometimes makes it more robust on smaller search trees.

**c. LCV AI (d=4) Variant**

* **Method:** An alpha–beta pruning agent with Least Constraining Value (LCV) heuristic.
* **Techniques Used:**  
  – Orders moves by simulating the options available to the opponent after the move (preferring moves that restrict the opponent).
* **Characteristics:**  
  This agent reorders the moves so that moves offering fewer responses (and therefore limiting the opponent) are evaluated first, which can often lead to stronger tactical play.

**d. Degree AI (d=4) Variant**

* **Method:** Alpha–beta pruning with a degree heuristic.
* **Techniques Used:**  
  – Prioritizes moves based on the number of available moves in the target board that the opponent would have.
* **Characteristics:**  
  By considering the degree (number of available moves) in the subsequent board, this variant selects moves that leave the opponent with fewer options, typically leading to a more constrained game state.

**e. CSP-FC-AC3-MRV (d=4) Variant (CSPSolver)**

* **Method:** A Constraint Satisfaction Problem approach combining:  
  – **Forward Checking (FC):** Reduces the domain of moves after applying a tentative move.  
  – **Arc Consistency (AC3):** Runs a full AC–3 algorithm to eliminate inconsistent move assignments.  
  – **MRV Heuristic:** Orders moves by estimating the “tightness” (or number of remaining valid moves) for the opponent. – **Alpha–Beta Pruning:** Integrates this pruning method in the backtracking search.
* **Characteristics:**  
  The solver first builds an explicit CSP representation of the current state (defining variables for each valid move with domains and constraints such as active board rules, win opportunities, and blocks). It then applies full AC–3 propagation over all arcs in this CSP to prune the search space before recursively backtracking over moves.

**3. Performance Comparison Among Variants**

**BestAI / Default Variant and Its Derivatives (LCV AI, Degree AI)**

* **Effective Search Depth:**  
  These variants focus on using alpha–beta pruning to cut off non-promising branches and often search deeper because they can process more nodes within the same time budget.
* **Heuristic Move Ordering:**  
  Both LCV AI and Degree AI use specialized heuristics to reorder moves. This indirect approach (via evaluation of future moves) is lightweight and allows the algorithm to maintain high throughput.
* **Implicit Constraint Enforcement:**  
  They rely on the get\_valid\_moves function and board state updates (like checking for wins or draws) which function as implicit forward checking. They do not incur the full overhead of constructing a CSP and performing a comprehensive AC–3 check.

**Basic Minimax Variant**

* **Without Pruning:**  
  Although the basic minimax variant expands more nodes due to the absence of alpha–beta, its simple structure allows it to evaluate the board quickly. However, it may be less efficient in practice when compared with the optimized BestAI variants.
* **Performance Impact:**  
  Its simplicity sometimes results in unexpectedly strong play on smaller or medium-depth searches since it avoids the overhead of additional heuristics.

**CSP-FC-AC3-MRV Variant (CSPSolver)**

* **Heavy Overhead:**  
  The explicit construction of a CSP representation, combined with the full implementation of AC–3 and forward checking, adds significant computational overhead at each search node.
* **Reduced Effective Depth:**  
  The extra work done to enforce full arc consistency—especially in a dynamic adversarial game where the state changes after every move—means the solver ends up exploring fewer nodes overall. Consequently, it sometimes does not search as deeply as the other agents.
* **Over-Pruning Risk:**  
  The full AC–3 algorithm, while theoretically powerful for pruning inconsistent domains, may be too aggressive or costly in this context. In some cases, it can prune branches that, although appearing inconsistent in the local CSP formulation, might have provided strong adversarial play.
* **Indirect vs. Explicit Constraint Propagation:**  
  Whereas the BestAI and its LCV or Degree variants perform constraint checking indirectly (by simply updating the board state and evaluating win/draw conditions), the CSP solver does so explicitly. This explicit method—although it can theoretically guarantee consistency—results in slower move calculation and a shallower search.

**4. Analysis: Why the AC3-Enabled CSP Variant Is Underperforming**

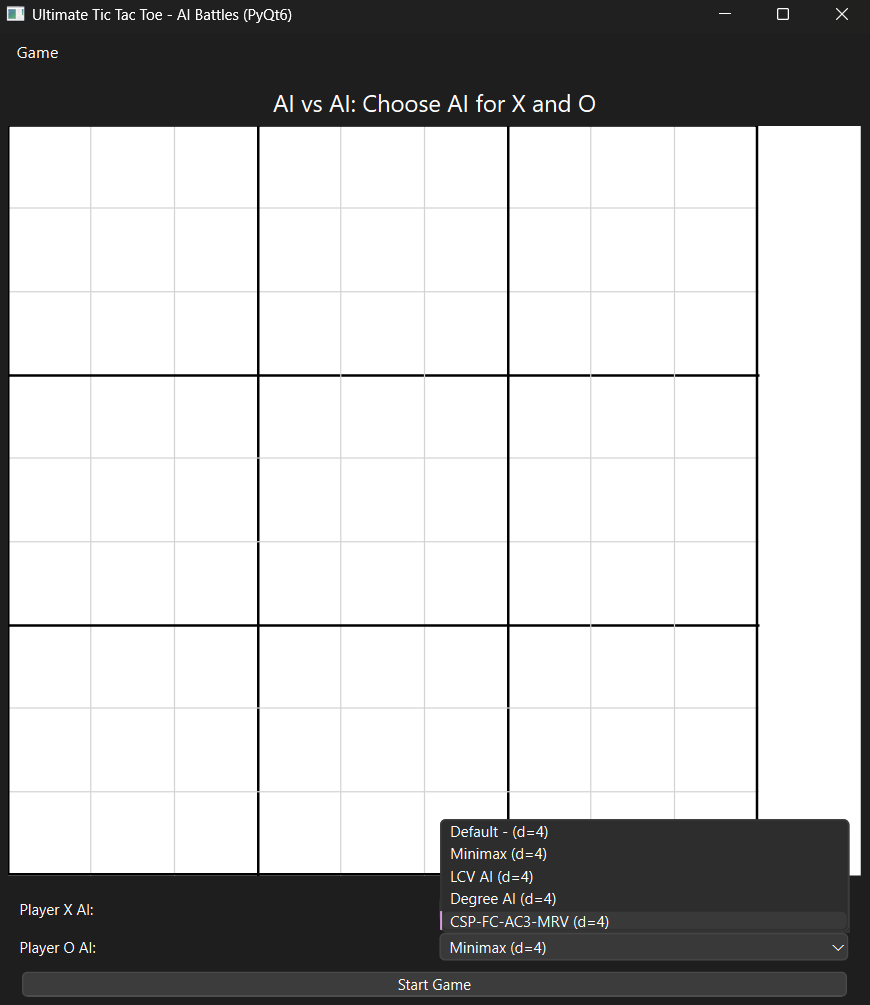
* **Computation Overhead:**  
  The overhead of constructing CSP variables and applying a full AC–3 routine at each decision node significantly reduces the number of nodes the solver can explore within the same time budget. In a game with many branching options, this lower effective search depth can lead to suboptimal move decisions.
* **Over-Aggressive Constraint Propagation:**  
  The explicit AC–3 implementation aims to prune inconsistent moves but can sometimes eliminate promising moves, especially when used with a strict MRV heuristic. In contrast, the other variants rely on simpler checks that work “good enough” without incurring as much cost.
* **Indirect Benefits in Other Variants:**  
  The BestAI, LCV, and Degree AI variants embed elements of constraint propagation (such as checking for valid moves and evaluating board scores) without the full overhead of solving a separate CSP problem. Their backtracking search naturally incorporates win/draw and occupancy rules at every step without explicitly enforcing all constraints with a separate AC–3 process. This balance allows them to efficiently explore deeper search trees and make stronger decisions.

**5. Conclusion**

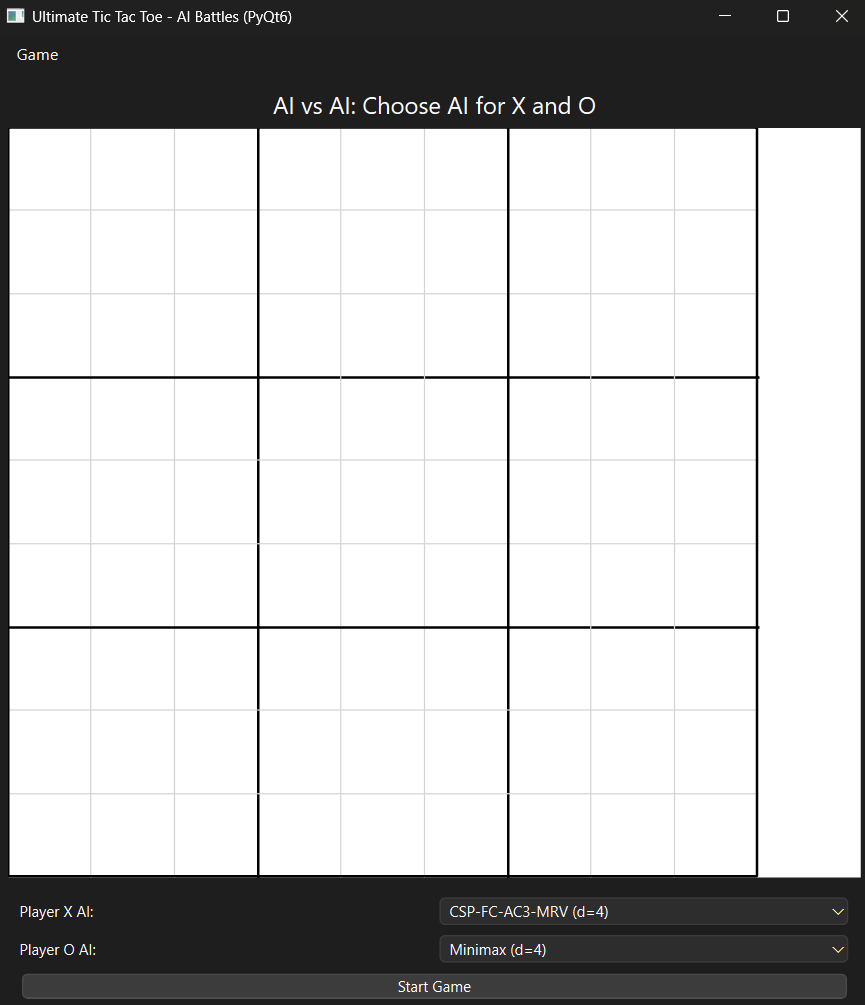
Overall, while the CSP-FC-AC3-MRV variant is attractive from a constraint-satisfaction perspective, its heavy-handed implementation leads to poorer performance compared to the more streamlined BestAI variants that indirectly enforce constraints. Optimizing or selectively applying the heavy CSP components may help improve its performance while still enjoying the benefits of deeper constraint propagation.

**6. Screenshots – Demo**

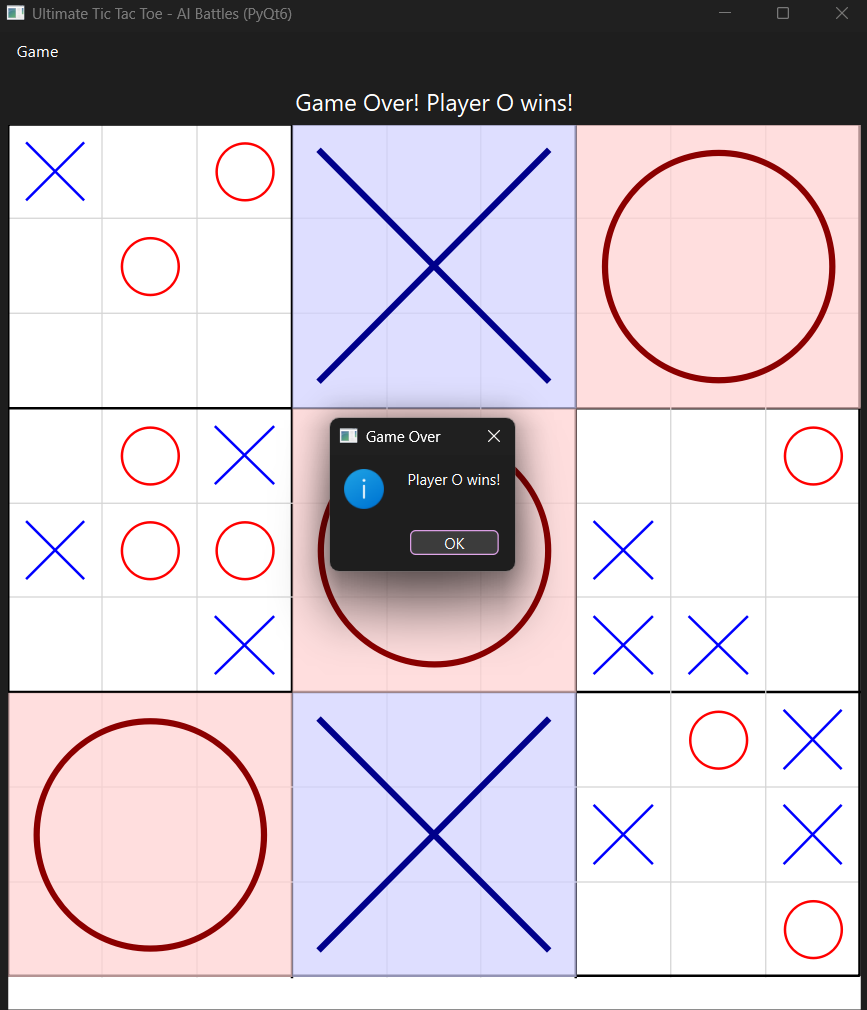
**Selecting AIs**

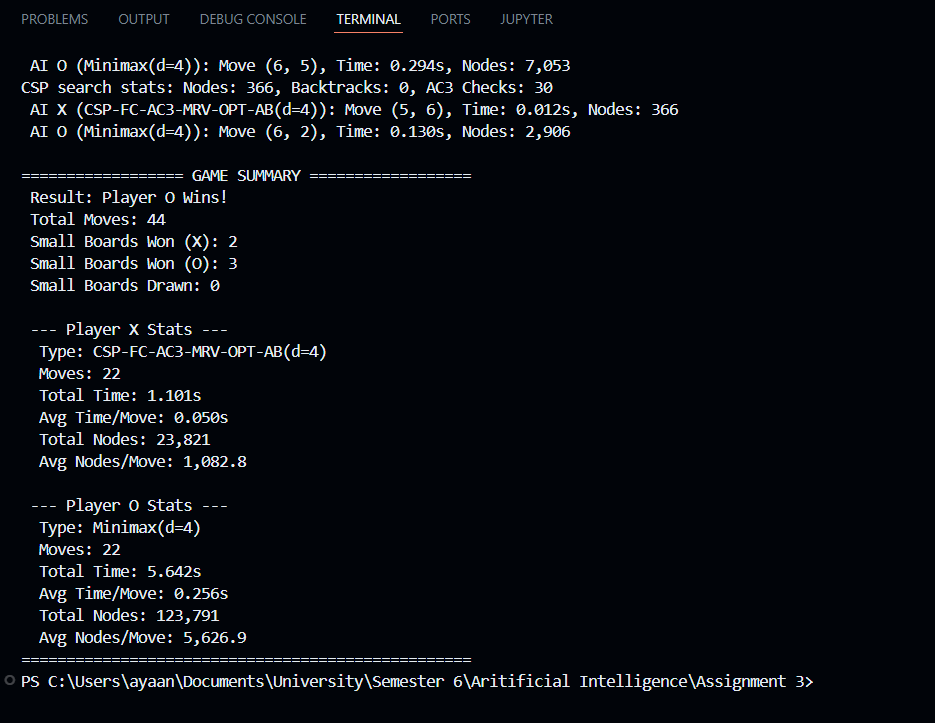
****

**Starting Game:**

****

**Game Finished:**

****

**Performance:  
**