MATH 8650

Project Report:

Quarto Board Game Computer vs Human : Using Minimax Algorithm

By
Netra Inamdar
Mihir Phatak

CONTENTS

INTRODUCTION & MOTIVATION

Board games are one of the oldest major streams of AI (Shannon and Turing 1950). Board games present a very abstract, rule pruned and pure form of dualistic competition between opponents that clearly require a form of “intelligence”. The states of a game are easily represented by a relevant choice of data structure. The probable actions of the players are well-defined in a complete representation of the game state.

Realization of a board game as a search problem is algorithmically solvable. The individual states after every move are fully accessible to the logical representation of the algorithm. It is nonetheless a contingency problem, because the characteristics of the opponent are not known in advance.

Good game programs possess the property that they delete irrelevant and redundant branches of the game tree, use good evaluation functions for in-between states, and look ahead as many moves as possible. We chose Quarto since it is the least explored alternative for AI based game program. Also its limited set of pieces adds additional challenges to its search space strategy.

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Initial call with MAX-VALUE(*initial-state*, $-\infty$, $+\infty$)

Fig. 1.1 Alpha-beta-Search for Quarto [5]

WHAT IS QUARTO?

Quarto is a Game Board played an a 4 X 4 board. There are 16 pieces on an individual board. Pieces are either black or white, small or tall, round or square, and solid or hollow.

The goal of the game is to have four pieces on a row (horizontally, vertically or diagonally) such that they all have one of the four properties in common, e.g. four tall pieces.

An additional twist that has a major impact on the game's complexity, is that players do not in turn pick a piece and put it on the board. Instead, the piece that one player must place on the board is chosen by its opponent.

The game thus starts with one of the players choosing a piece and giving it to his opponent. The opponent puts it on the board and then chooses a piece to be put ne

Fig 1.2 Quarto Board Complete Set

MINIMAX ALGORITHM WITH ALPHA BETA PRUNING

The minimax algorithm is a way of finding an optimal move in a two player game. Alpha-beta pruning is a way of finding the optimal minimax solution while avoiding searching subtrees of moves which won't be selected. In the search tree for a two-player game, there are two kinds of nodes, nodes representing your moves and nodes representing your opponent's moves.

Nodes representing your moves are generally drawn as squares (or possibly upward pointing triangles). These are also called MAX nodes. The goal at a MAX node is to maximize the value of the subtree rooted at that node. To do this, a MAX node chooses the child with the greatest value, and that becomes the value of the MAX node.

Nodes representing your opponent's moves are generally drawn as circles (or possibly as downward pointing triangles). These are also called MIN nodes. The goal at a MIN node is to minimize the value of the subtree rooted at that node. To do this, a MIN node chooses the child with the least (smallest) value, and that becomes the value of the MIN node.

```
function MAX-VALUE(state, game, α, β) returns the minimax value of state
  inputs: state, current state in game
          game, game description
          α, the best score for MAX along the path to state
          β, the best score for MIN along the path to state

  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
      α ← MAX(α, MIN-VALUE(s, game, α, β))
      if α ≥ β then return β
  end
  return α

function MIN-VALUE(state, game, α, β) returns the minimax value of state

  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
      β ← MIN(β, MAX-VALUE(s, game, α, β))
      if β ≤ α then return α
  end
  return β
```

1

Fig. 1.3 MINIMAX Algorithm

IMPLEMENTATION STRATEGY (DATA STRUCTURES & ALGORITHMS FOCUS)

1. Implementation of minimax algorithm:

We have implemented the above mentioned minimax algorithm to minimize the min's advantage and maximize the max's advantage to present an optimal computer program that competes against a human opponent. The approach constructs a game tree which looks for the most optimal moves using depth first search.

2. Implementation of a heuristic evaluation function:

In Quarto, the higher the number of lines that can be completed with a given piece (rows, columns, or diagonals), the higher the chance that one of these lines can be ultimately completed. Line implies any row column or diagonal / co-diagonal. So, depending on whose turn is to be evaluated, the number of such possible lines or its negation is expected to serve well as a heuristic function. By comparing the heuristic assigned to the value of a draw (0), the AI may tend to enforce a draw on its opponent or take a counter-risk in a certain direction.

3. Using alpha beta pruning for minimax optimization

Alpha-beta pruning gets its name from two bounds that are passed along during the calculation, which restrict the set of possible solutions based on the portion of the search tree that has already been seen. Specifically, Beta is the minimum upper bound of possible solutions,Alpha is the maximum lower bound of possible solutions. Thus, when any new node is being considered as a possible path to the solution, it can only work if: alpha<= N <= Beta, where N is the current estimate of the value of the node.

COMPLEXITY ANALYSIS AND RESULTS:

Game trees are, in general, very time consuming to build, and it's only for simple games that it can be generated in a short time. If there are b legal moves, i.e., b nodes at each point and the maximum depth of the tree is m, the time complexity of the minimax algorithm is of the order $O(b^m)$.

Alpha-Beta pruning, an optimization technique for minimax algorithm, reduced the computation time by a huge factor. This allowed us to search much faster and even go into deeper levels in the game tree. It cut off branches in the game tree which were not needed to be searched because there was a better move available. The complexity with alpha beta pruning was reduced to $O(b^{m/2})$.

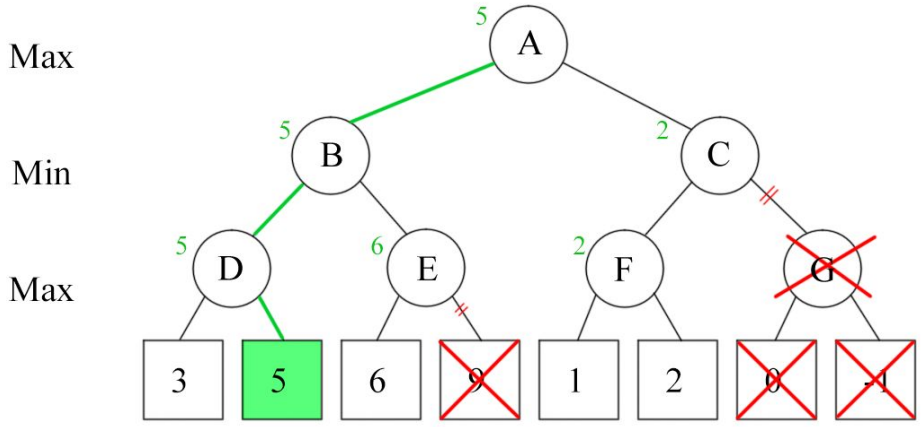

Fig 1.4 Alpha beta pruning example

As shown in the below figures, with alpha beta pruning optimization technique, the time required for computation of best possible next move reduces to a great extent, compared to minimax algorithm with same depth and number of computer's moves. The plot is not linear because, for every node of min and max, alpha & beta values get updated and the search in decision tree varies for each move.

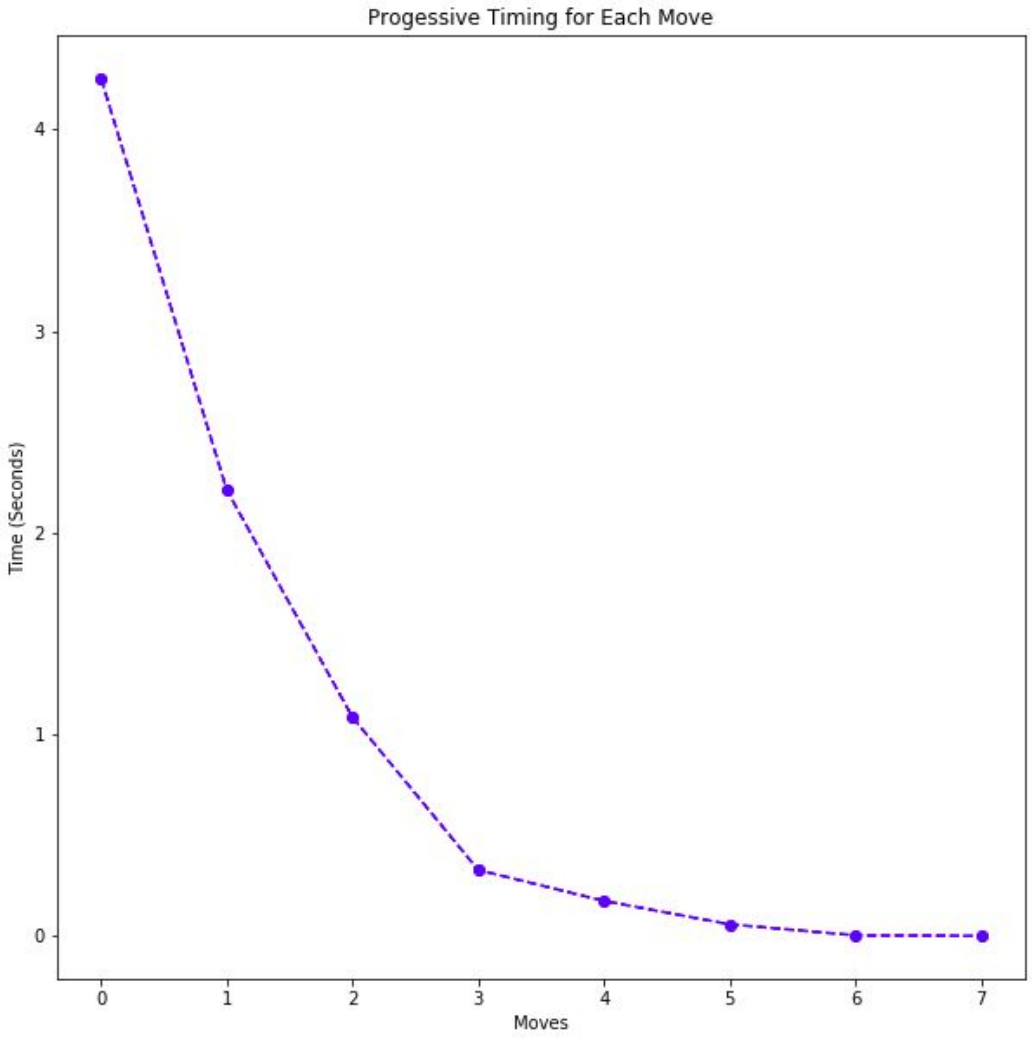

Fig 1.5 Computation time analysis with depth as 3, number of turns= 8

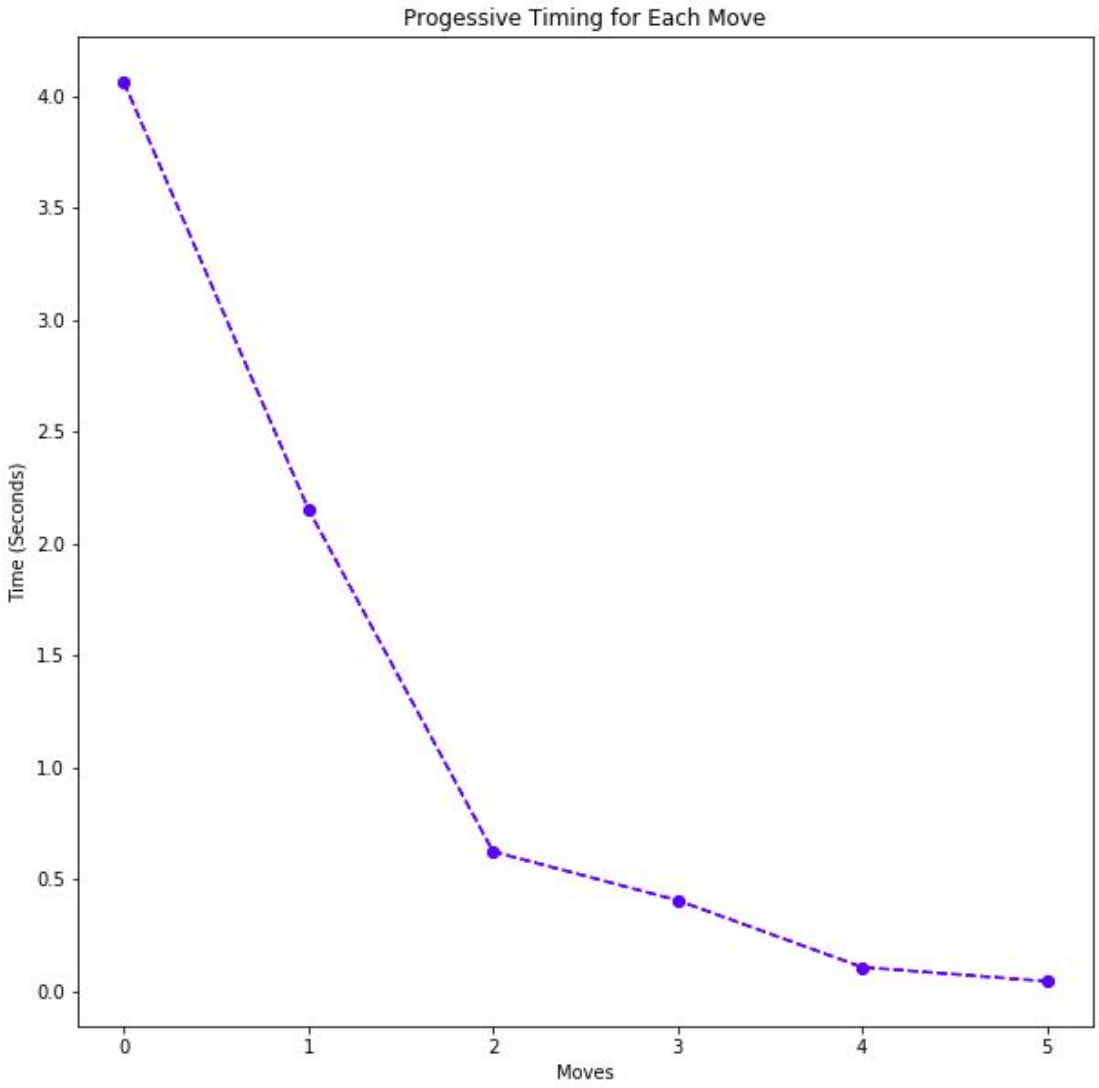

Fig 1.6 Computation time analysis with depth as 3, number of turns= 6

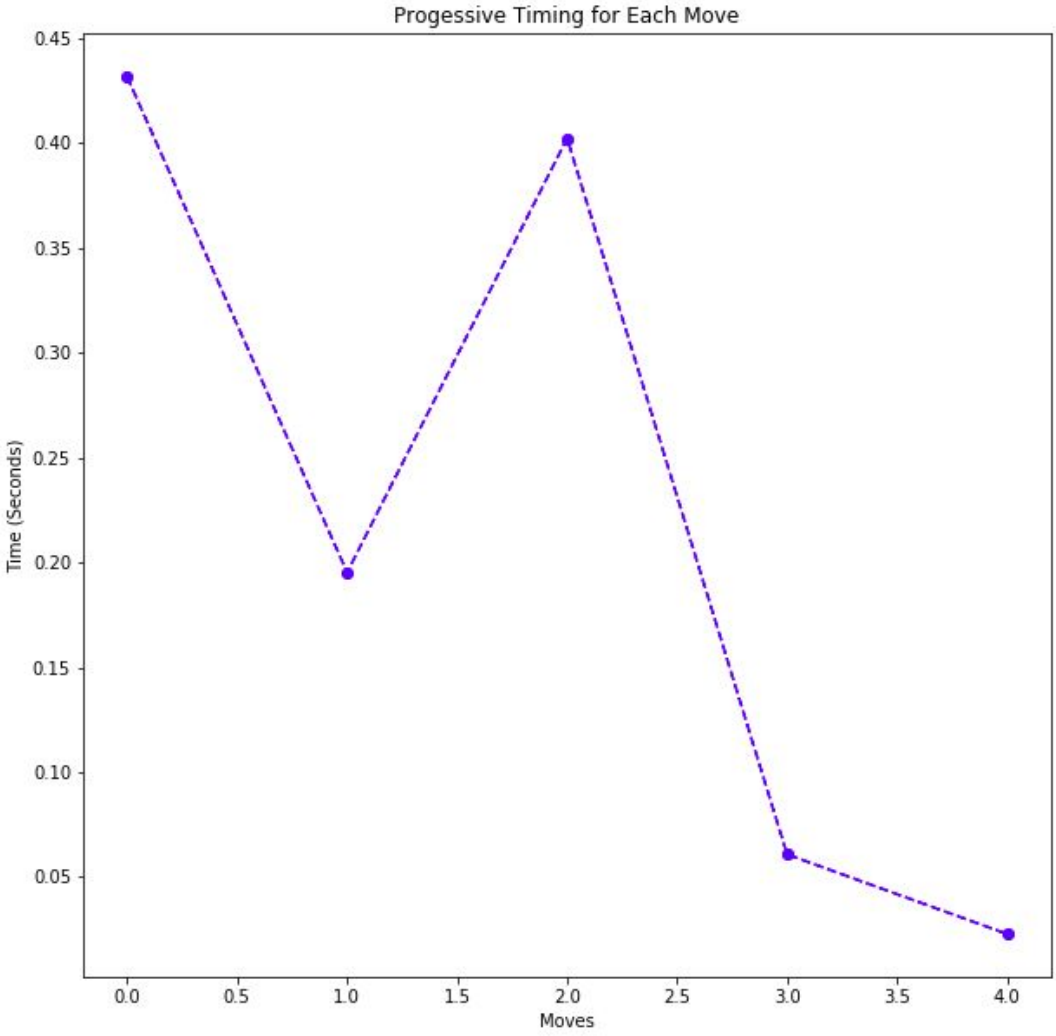

Fig 1.7 Computation time analysis with depth =2, number of turns= 5

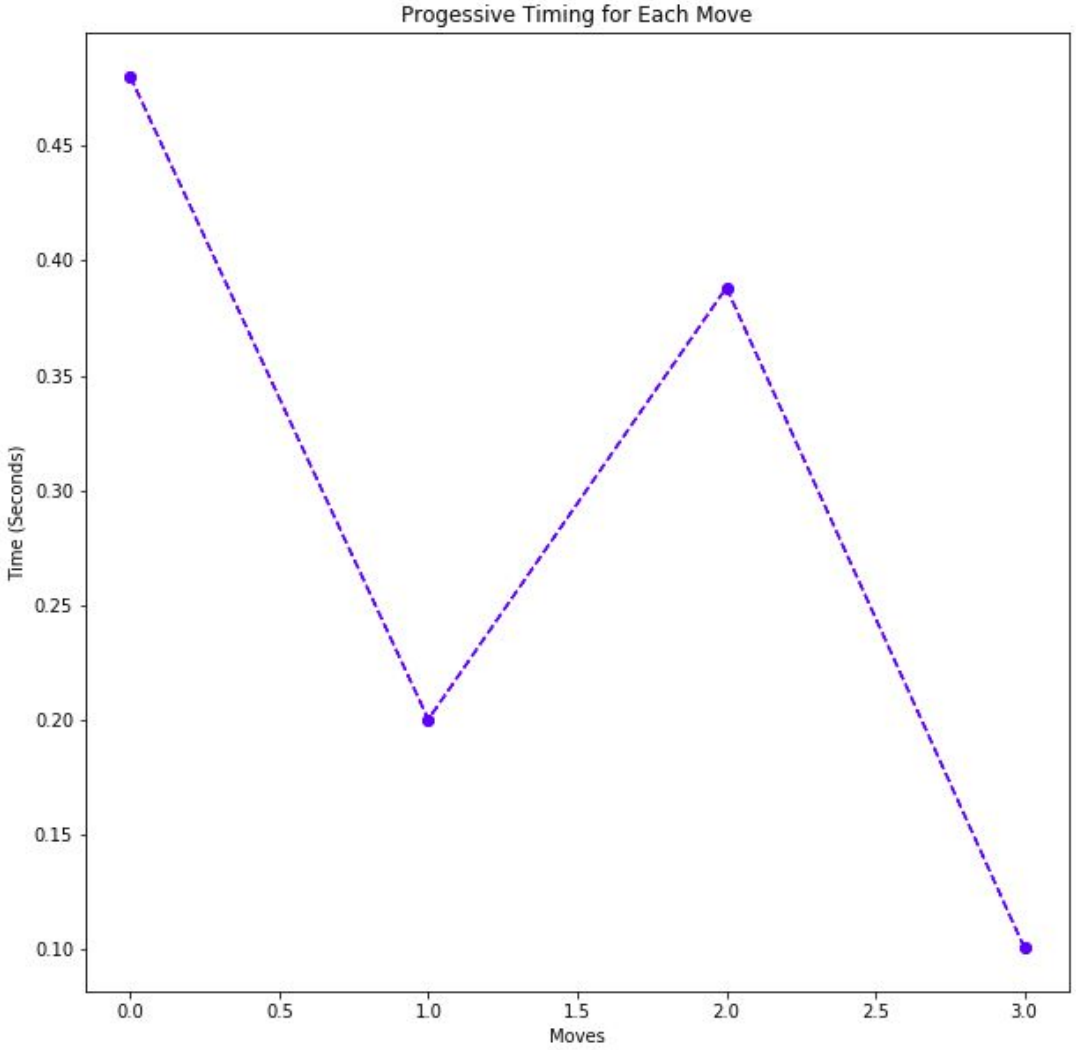

Fig 1.8 Computation time analysis with depth =2, number of turns= 4

FUTURE SCOPE:

1. Symmetry search algorithm:

We can consider two kinds of symmetries in Quarto: piece symmetries board symmetries. By resolving these, different states can be mapped to indistinguishable equivalence states whose number is much smaller than that of all possible states. Equivalent boards are determined by rotating or mirroring. Altogether, each equivalence board subsumes 32 symmetric boards.

One way to find all board symmetries is to rearrange the pieces on the fully occupied board in all possible ways and save lineups that share the same structure as the original lineup. This structure is composed of ten combinations (four possible full lines along rows, four along columns, and two along diagonals). Two game boards are equivalent if each combination of four pieces on the original board is present on the equivalence board. Hereby, the order within the four pieces does not matter.

Using symmetries to reduce the search space of the game tree:

The inherent symmetry of a square game board arises out of its vertical, horizontal, circular and other symmetries. Types of Symmetries : 1) Counter Clockwise 2) Clockwise and 3) Diagonal.

Symmetries help in reducing the redundant searches in a decision tree by ruling out a few legal moves which can be verified in a traversed node by virtue of board symmetry.

2. Transposition table for symmetries:

Simply caching the (state, Minimax value) pairs, so when we have to evaluate the same state again instead of expanding the whole sub-tree we can quickly take the value from the transposition table.

REFERENCES :

1. https://en.wikipedia.org/wiki/Quarto_(board_game)
2. http://wouterkoolen.info/Talks/Quarto.pdf
3. http://web.archive.org/web/20041012023358/http://ssel.vub.ac.be/Members/LucGoossens/Quarto/Quartotext.htm
4. https://boardgames.stackexchange.com/questions/8922/are-there-guaranteed-winning-strategies-for-Quarto
5. http://gki.informatik.uni-freiburg.de/teaching/ss14/gki/lectures/ai06.pdf
6. http://grantbartel.com/blog/playing-strategy-games-minimax/
7. http://suendermann.com/su/pdf/pppj2013.pdf
8. http://www.massey.ac.nz/~mjjohnso/notes/59302/all.html
9. https://github.com/hugomailhot/pyquarto