

Computing efficiency of parallelism in AI algorithms using adversarial games

Team members

18BCE1009 - Madhurima Magesh

18BCE1026 - Ashwin Guptha

18BCE1170 - Sibi Akkash

18BCE1171 - Siddarth S

Objective

To study effective parallelization of AI algorithms on Connect-4. Our project focuses on using parallelism to compute the efficiency of each AI algorithm and see which one provides the best results.



Scope

This project focuses on a fairly simple game but in reality there are many more games consisting of way more complex algorithms and graphics which require the better optimization methods.



Abstract

The project investigates the efficiency of AI algorithms such as minimax, alpha-beta, progressive deepening for search in a game tree. The game used in this case study is Connect-4.

The suggested parallel computational model exploits tree partitioning at width for each level of the game tree and is based on combination of the parallel algorithmic paradigms. Performance comparison has been made for hybrid (multilevel), flat and multithreaded parallel programming models. Speedup and efficiency as well as scalability in respect to the size of the multicomputer and its impact on the performance of the parallel system will be estimated on the basis of experimental results.



Literature review

- Parallel Minimax Tree Searching on GPU Kamil Rocki and Reiji Suda
http://olab.is.s.u-tokyo.ac.jp/~kamil.rocki/rocki_ppam09.pdf
- Parallel Implementation and Optimization of the Minimax Algorithm with Alpha-Beta Cutoffs in the context of the game Othello, Amy S. Biermann, CRPC Summer Research Student, Caltech
<http://www.pressibus.org/ataxx/autre/minimax/paper.html>
- Comparative study of performance of parallel Alpha Beta Pruning for different architectures, Shubhendra Pal Singhal M. Sridevi
<https://arxiv.org/pdf/1908.11660.pdf>
- Parallelization of Alpha-beta Pruning Algorithm for Enhancing the Two Player Games Akanksha Kumari, Shreya Singh, Shailja Dalmia, Geetha V
http://www.iraj.in/journal/journal_file/journal_pdf/12-344-149466366874-81.pdf
- Alpha-Beta Pruning in Mini-Max Algorithm –An Optimized Approach for a Connect-4 Game Rijul Nasa, Rishabh Didwania, Shubhranil Maji, Vipul Kumar
<https://www.irjet.net/archives/V5/i4/IRJET-V5I4366.pdf>

Software Requirements

1. C++
2. OpenMP



Short Description of the Project

In our project we aim to build Connect 4 where the user plays against the computer. The computer plays its moves using AI algorithms like Alpha Beta pruning, Min-Max, Progressive Deepening. To improve the performance, efficiency and reduce the response time we will be doing parallel computing. Thus we will be able to observe and infer how parallel computing is very helpful to solve larger objects.



Min-Max Algorithm

Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the other player is also playing optimally.

It is used in games such as tic-tac-toe, go, chess, Isola, checkers, and many other two-player games.

Such games are called games of perfect information because it is possible to see all the possible moves of a particular game.

There can be two-player games which are not of perfect information such as Scrabble because the opponent's move cannot be predicted.

It is similar to how we think when we play a game: “if I make this move, then my opponent can only make only these moves,” and so on.

Minimax is called so because it helps in minimizing the loss when the other player chooses the strategy having the maximum loss.

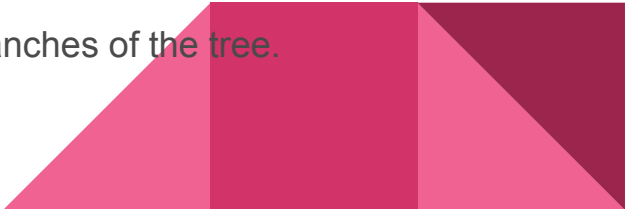


Alpha-Beta Pruning

Alpha Beta Pruning is all about reducing the size (pruning) of our search tree. While a brute-force approach is an easier approach to use, it doesn't necessarily mean it is the most optimal approach. Many times, one doesn't need to visit all possible branches to come up with the best possible solution in hand.

Thus we need to provide the min-max algorithm with some stopping criteria using which it would stop searching a region of the tree once it finds the guaranteed minimum or maximum at that level. This would prevent the algorithm from using additional computational time, making it much more responsive and fast.

The original min-max algorithm performs traversals of the tree in a left to right fashion while also going to the deepest possible depth of the tree. This essentially is a depth-first approach. It then discovers values that must be assigned to nodes directly above it, without ever looking at other branches of the tree.



Progressive Deepening

In computer science, iterative deepening search or more specifically iterative deepening depth-first search (IDS or IDDFS) is a state space/graph search strategy in which a depth-limited version of depth-first search is run repeatedly with increasing depth limits until the goal is found. IDDFS is optimal like breadth-first search, but uses much less memory; at each iteration, it visits the nodes in the search tree in the same order as depth-first search, but the cumulative order in which nodes are first visited is effectively breadth-first.

