Project Milestone Report

Summary of Work Completed

Since the project's commencement, we have focused on setting up the foundational aspects of the chess AI engine. We successfully implemented the basic components of a sequential chess engine. We initially attempted to build these components from scratch but soon realized the complexity involved in creating an efficient and error-free chess engine. This led us to decide on using the “libchess” library, which provides robust functionalities for managing chess game states and generating legal moves and will also enable us to more easily utilize a GUI for demonstration purposes later on.

Progress Relative to Goals

Our initial proposal aimed to develop a parallel chess AI capable of achieving a significant speedup in move selection by leveraging multi-core processing with OpenMP. While we have laid down the groundwork for the chess engine using `libchess`, the parallelization of the minimax algorithm with alpha-beta pruning is still in the early stages.

Current Status

- Basic sequential chess engine functionalities implemented including evaluation and search.
- Preliminary implementation of the sequential minimax algorithm with alpha-beta pruning.

Adjusted Goals:

- Complete the parallelization of the minimax algorithm.
- Implement and optimize a parallel board evaluation function.
- Achieve a minimum of 2x speedup on 8 cores initially, with further optimizations aimed at reaching higher speedups.

Challenges

The primary challenges faced include the intricacies of efficiently parallelizing the minimax algorithm. Dependency of node evaluations and synchronization requirements have introduced complexity in achieving effective load balancing and minimizing communication overhead.

Revised Schedule

- Week 1/2:
  - Completed: Implementation of basic chess functionalities with `libchess`.
  - Completed: Basic sequential minimax algorithm.
- Week 3:
  - Task: Implement parallel version of minimax using OpenMP (Priyanshi)
  - Task: Begin integration of alpha-beta pruning into the parallel framework (Teddy)
- Week 4:
  - Task: Optimize load balancing and reduce synchronization overhead. (Priyanshi)
  - Task: Parallelize and optimize the board evaluation function. (Teddy)
- Week 5:
  - Task: Testing and debugging of parallel algorithms. Assigned to: Both

- Task: Performance tuning and final adjustments based on test results. Assigned to: Both

Deliverables for Poster Session
At the poster session, we plan to:
- Demonstrate the AI's decision-making process in real-time against attendees.
- Display comparative performance graphs highlighting the speedups achieved by our parallel AI against the sequential version.
- Discuss the parallelization techniques employed and their impact on performance and any interesting tradeoffs we observe.

Preliminary Results
No substantial performance data is available yet as the parallelization is still under implementation.

Concerns and Future Work
The main concerns revolve around ensuring effective parallelism without excessive overhead due to synchronization and ensuring that the AI remains competitive in terms of gameplay strength.

By the completion of the project, we aim to have a fully functional parallel chess AI that not only performs faster due to computational distribution across multiple cores but also maintains or improves the strategic depth of its gameplay compared to a sequential version.