Project Proposal: Parallel Computation of Nash Equilibria in N-player Games

Ally Du, Michael Cui

Project url: https://github.com/Ally1230/Parallel-NE

Summary:
The goal of our project is to implement and evaluate the algorithm proposed in the paper *Parallel Computation of Nash Equilibria in N-Player Games*. This algorithm leverages parallel computation techniques to efficiently compute Nash equilibria in non-cooperative games with multiple players. Nash equilibrium, a critical concept in game theory, represents a steady state where no player can unilaterally improve their payoff by changing their strategy.

This project explores the scalability and performance of parallelizing this computation using modern parallel programming paradigms such as OpenMP, MPI, and CUDA.

The key challenges we address include load balancing for subproblems, optimizing communication overhead, and improving memory access efficiency. By comparing different parallelization approaches and hardware architectures, the project will provide insights into effective methods for tackling large-scale game-theoretic computations.

Background:
A Nash equilibrium is a fundamental concept in game theory, describing a stable state in a strategic interaction where no player can improve their outcome by unilaterally changing their strategy. Named after mathematician John Nash, who formalized the concept, it applies to games with two or more players, each with a set of possible strategies and preferences over outcomes. At equilibrium, each player's strategy is the best response to the strategies chosen by others, meaning every participant is making an optimal decision given the choices of their counterparts. Nash equilibria are widely used in economics, political science, biology, and artificial intelligence to model competitive and cooperative scenarios, offering insights into behavior in markets, negotiations, and multi-agent systems. The concept is particularly valuable in understanding how rational agents interact in complex systems where individual decisions collectively shape outcomes.

Parallelizing the computation of Nash equilibria in n-player games is crucial due to the inherent complexity and high-dimensional nature of these problems. As the number of players and strategies increases, the computational demand grows exponentially, making it infeasible to solve large-scale games using sequential algorithms. Parallel computation allows for the division of the problem into smaller, more manageable subproblems, which can be processed concurrently. Additionally, parallelization facilitates real-time or near-real-time applications, such as optimizing multi-agent

systems or decision-making in dynamic environments, where timely computation is essential. By leveraging modern multicore processors and distributed systems, researchers and practitioners can efficiently explore the equilibrium landscape of complex n-player games.

Challenge:
Computing Nash equilibria in N-player games involves solving a combinatorial problem where the number of possible strategy combinations grows exponentially with the number of players and their strategy options. This makes the problem computationally expensive, especially for real-world applications in economics, artificial intelligence, and network design.

To parallelize this computation, we may encounter the following challenges:

1. Load Balancing:
   The subproblems created during the computation (e.g., solving subgames or evaluating payoffs) have varying computational complexity. Poor workload distribution can lead to resource underutilization and inefficiencies.
2. Communication Overhead:
   For distributed systems, frequent communication between processes (e.g., synchronizing convergence checks or sharing updated payoffs) can create bottlenecks.
   For GPUs, inter-thread communication and synchronization can degrade performance if not managed efficiently.
3. Memory Access Patterns:
   Accessing large payoff matrices and strategy profiles in parallel requires careful optimization to avoid contention and maximize cache efficiency on CPUs or memory bandwidth on GPUs.
4. Nonlinear Dependencies:
   The algorithm may require sequential dependencies at certain stages, complicating the task of parallelizing the entire process effectively.

Resources: We will reference this paper *Parallel Computation of Nash Equilibria in N-Player Games*. They have pseudo-code available, but no actual code.

Goals & Deliverables:
Goals:

1. Implement the parallel Nash equilibrium computation algorithm in OpenMP, MPI, and CUDA.
2. Benchmark and analyze performance for different problem sizes.

3. (if time allows) Explore advanced optimizations, such as graph-based decomposition or task-based parallelism.

Deliverables:

1. Parallel implementations (OpenMP, MPI, CUDA) of the Nash equilibrium algorithm.
2. Performance benchmarks comparing programming paradigms and hardware architectures.
3. Analysis of load balancing, communication overhead, and memory optimization techniques.
4. Final report and presentation summarizing our findings.

Platform Choice:

Multicore CPUs and GPUs excel at parallelizing tasks, allowing the simultaneous evaluation of multiple candidate equilibria or the decomposition of complex game structures. This is essential for handling the exponential growth in computational requirements as the number of players and strategies increases. GPUs, in particular, are optimized for matrix and vector operations, which are central to solving game-theoretic problems involving payoffs and mixed strategies. C++ is known for its high performance, it provides fine-grained control over memory and parallelization, making it ideal for computationally intensive tasks.

Schedule (per week):

- Week 1 (18 - 24 Nov): Literature review and familiarization with the algorithm and tools.
- Week 2 (25 Nov - 1 Dec) Implement sequential baseline and OpenMP parallel CPU implementation.
- Week 3 (2 - 8 Dec) Implement MPI version, analyze results, optimize implementations, and finalize the report.