15-418/618 Spring 2023
Parallel Computation of Nash Equilibria - Project Milestone

Xiaoyu Zhang: xiaoyuz4@andrew.cmu.edu
Yizhou Chen: yizhouch@andrew.cmu.edu

URL

https://famishedfish.github.io/Parallel-Computing-Project/15618_milestone.pdf

SUMMARY

So far, we have formally defined the Nash equilibrium game, abstracted the problem into characteristics like payoffs, strategies and so on. We have also implemented the sequential version of three algorithms to find Nash equilibrium in 2-player games, the following is our detailed work.

Enumeration

Enumeration algorithm works by iterating all possible combination of strategies and determinate if they are Nash equilibria. The determination function iterate over all players in the game. For each player, it calculates the expected payoff for their current strategy. The expected payoff is the sum of the payoffs for each outcome of the game. Then our program checks if there is a different strategy that gives this player a higher payoff. It does this by iterating over all possible strategies for the player, except their current one. For each alternate strategy, it calculates the expected payoff as before, and compares it to the current expected payoff. If the new expected payoff is higher, the function returns false, indicating that the given strategy profile is not a Nash equilibrium.

Compute equilibria via simplicial subdivision

The method constructs a grid with triangles covering the space of mixed strategy profiles and applies a path-following technique to calculate an approximate fixed point, which is used to further refine the grid. The algorithm iteratively repeats this process with increasingly finer grids until it identifies a mixed strategy profile where the maximum regret is minimal.

Global Newton Method

The global Newton method is a numerical optimization technique used to compute Nash equilibria in non-cooperative games. The algorithm starts with an initial guess for the Nash equilibrium and iteratively improves the guess by solving a system of nonlinear equations that represent the necessary conditions for a Nash equilibrium, using a variant of the Newton-Raphson method. The method can converge to a Nash equilibrium from any initial guess, but can be computationally expensive and may converge to local optima.

Issues

Enumeration

Parallelizing enumeration algorithms can be relatively easy due to the fact that the enumeration process inherently involves searching through a large search space for a solution, and each iteration is independent, which makes the algorithm to be easily parallelized across multiple threads or processors.

Compute equilibria via simplicial subdivision

Simplicial subdivision method relies on the recursive subdivision of simplices, synchronization issues can arise when multiple processors or threads access the same simplex simultaneously. We may consider using openMP tasks to parallelize the recursive part of this algorithm.

Global Newton Method

Parallelizing the global Newton method involves distributing the computation of the Jacobian matrix, which requires careful design to ensure correctness, achieve load balancing, and reduce communication overhead.

Global Newton method can converge slowly for some games, which leads to expensive computations. We may want to adopt some novel techniques, e.g., pipelining, to improve performance.

Deliverables and Goals

This final deliverable goal is to show the demo of our paralleled version of finding Nash equilibrium. Furthermore, we plan to package our program into the python library to make it easier to use.
We are currently about 45% of the way to the goal set in the proposal. We have a general idea of how to parallelize and expand the algorithms to more generalized situations, such as support multi-players, but we have not yet implemented it. As for the ”nice to haves” in the proposal, which now seem a bit ambitious, we will try hard, but there is no guarantee that they will be achieved.

Revised Schedule

04/20 - 04/25 Scale three algorithms to N-player games and mixed strategy
04/26 - 04/28 Implemented parallel version of all three algorithms
04/29 - 05/01 Design experiments to compare different paralleling strategies
05/02 - 05/03 Package deliverables, complete report and poster