Benchmark Performances

Jake R.
John G.
Cyrus J.

Probability Simulation Game Engine

CSCE 585

Testing Setup

Overview of Monte Carlo Simulations:

1. Define a domain of possible scenarios/inputs
2. Generate the scenarios randomly from some distribution
3. Perform a deterministic computation of the outputs
4. Aggregate the results

For our test case, we are simulating the scenario of drawing a land card. This will be the core task for our project moving forward and will expand to other predictors later on.

Hardware Strain Analysis

Unavoidable Process of Testing: Around 10^5 simulations another cpu process took priority causing a drop in Program Performance.

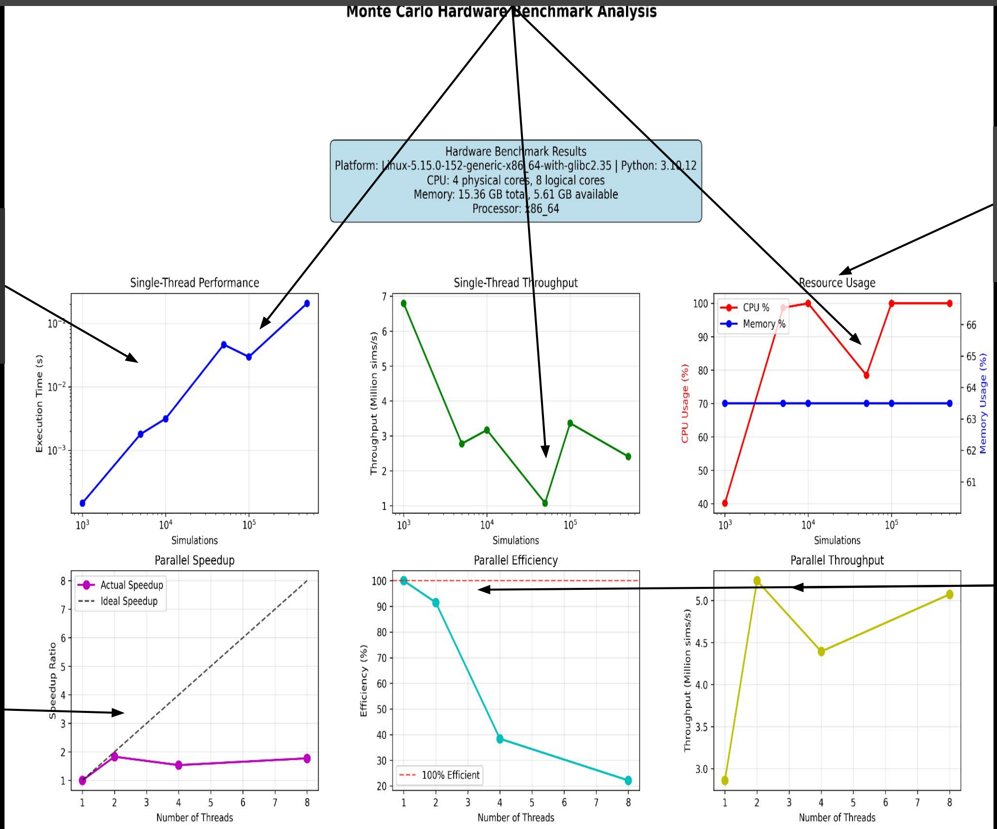
Generally Speaking, More Simulations involves more Execution Time.

Performance appears to be CPU - Limited rather than Memory limited.

Parallel Efficiency Suggest that performance will drop after significantly after adding more than 2 CPU cores.

Parallel Throughput suggest that while it will come with some performance increases it won't be significant after 2 cores.

Python Global Interpreter Lock. - Sets a lock on Parallel Efficiency After a Certain Point.

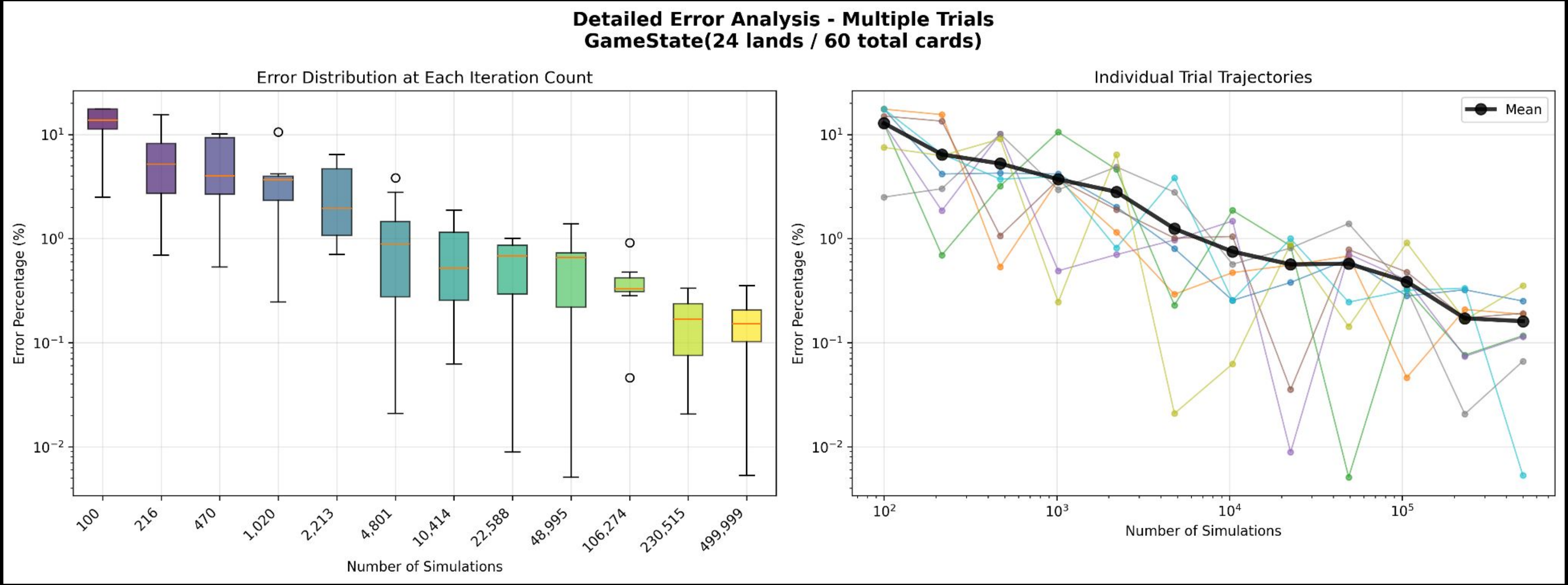


Hardware Benchmark Results
Platform: Linux-5.15.0-152-generic-x86_64-with-glibc2.35 | Python: 3.10.12
CPU: 4 physical cores, 8 logical cores
Memory: 15.36 GB total, 5.61 GB available
Processor: x86_64

Simulation Analysis:

Shows that as simulation count goes up error can drop lower than 1% when compared to actual probability

The Mean of all the trail runs follow the expected convergence trend of error and variance that comes with randomness as well as it this graph shows that lower number of simulations will result in larger variance and overestimation/underestimation.

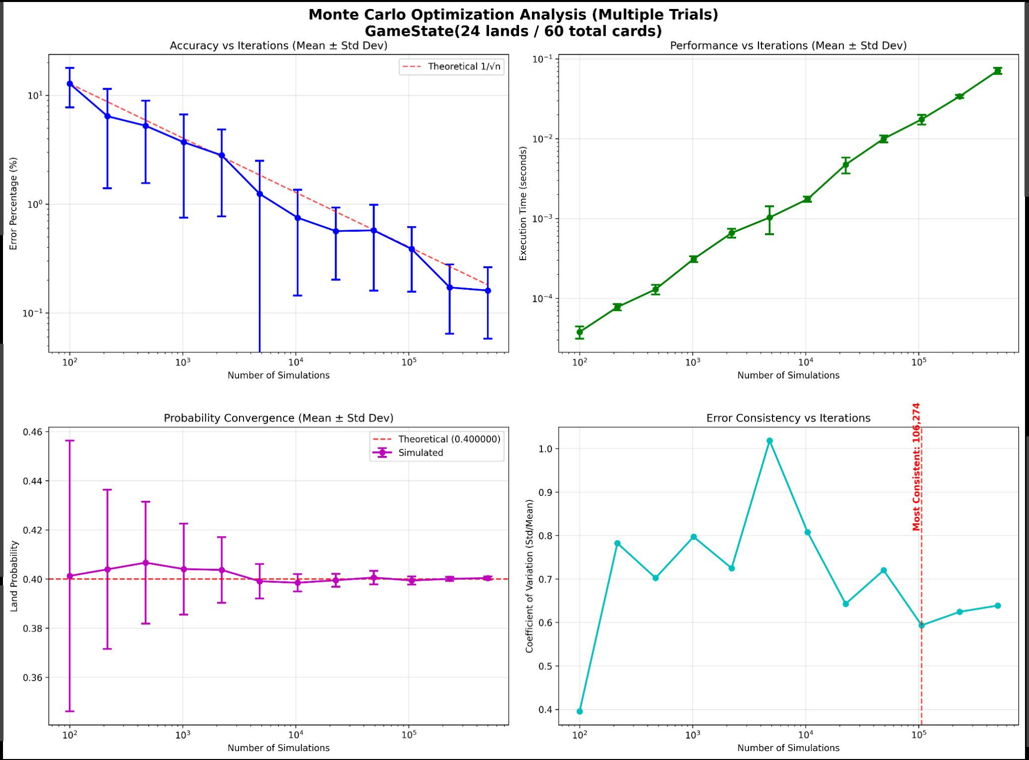


Simulation Optimization Analysis using Multiple Trials.

This graphic Shows that our model can perform at or below the Theoretical upper limit of this type of Model in predicting our problem.

Probability Simulation Shows that as the number of simulations increase the output will converge closer to the expected result.

This should hold true even for when we add more variables and predictors.



On average our program will have a linear increase in latency based on number of simulations.

Shows that after a certain number of simulations the values level out and will give similar results. Meaning we can cut off our amount of simulations to save on compute costs.