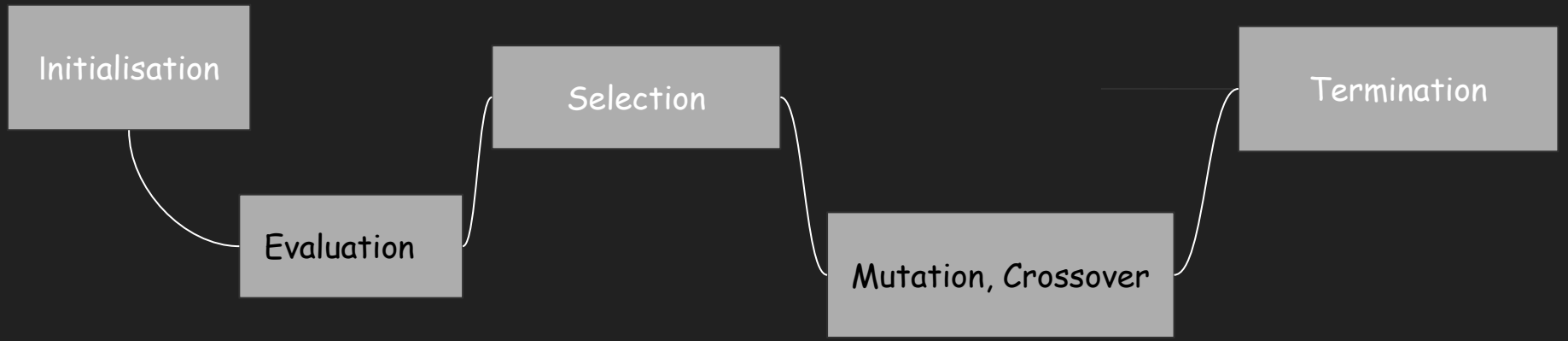
Genetic Algorithm - Parallelism



Applications

1. Machine Learning and Neural Network Training : **Neural Network Architecture Optimization , Weight Optimization**

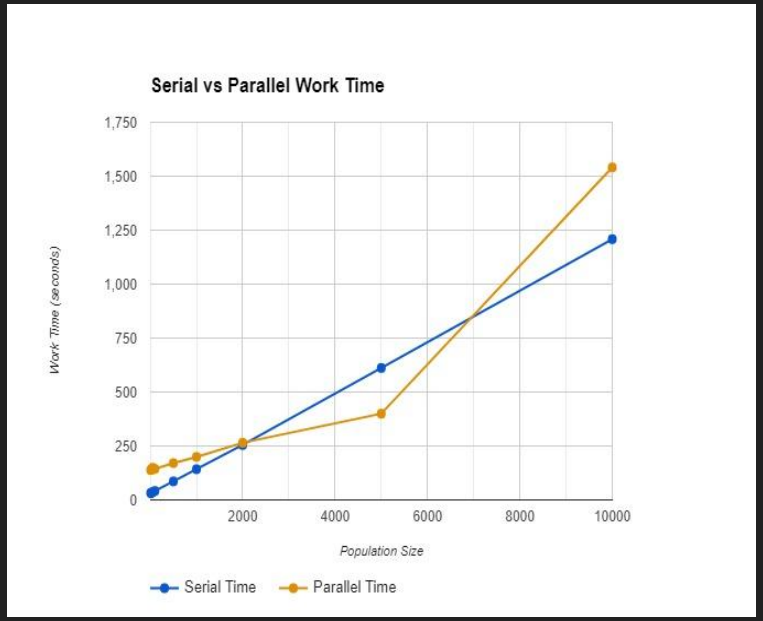
Parallelism Implemented

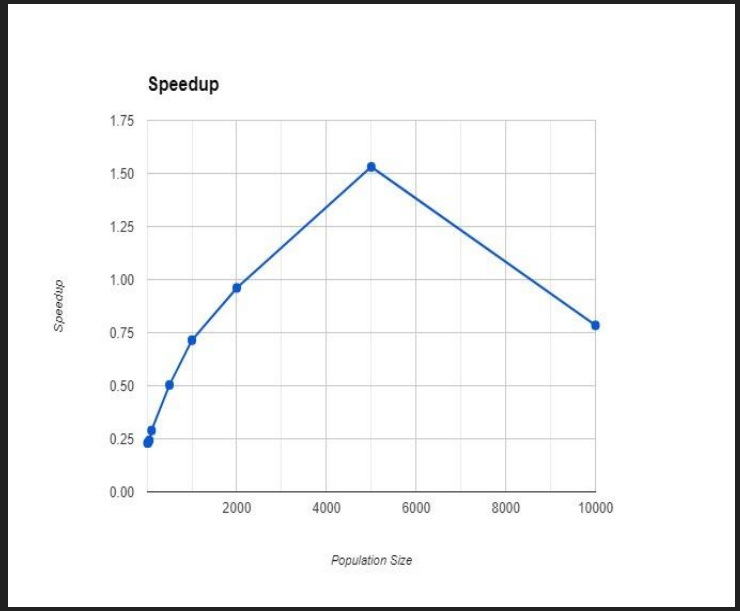
Task parallelism involves breaking down a problem into independent subtasks that can be executed concurrently on multiple processing units (cores or nodes in an HPC system).
Here's how it's achieved in the code:

Dividing the Workload: The evaluate Population function is responsible for assessing the fitness of each chromosome in the population. The code creates multiple threads (using std::thread), each assigned a chunk of the population to evaluate (evaluate Population Parallel). This distributes the evaluation tasks across different threads, enabling parallel execution.

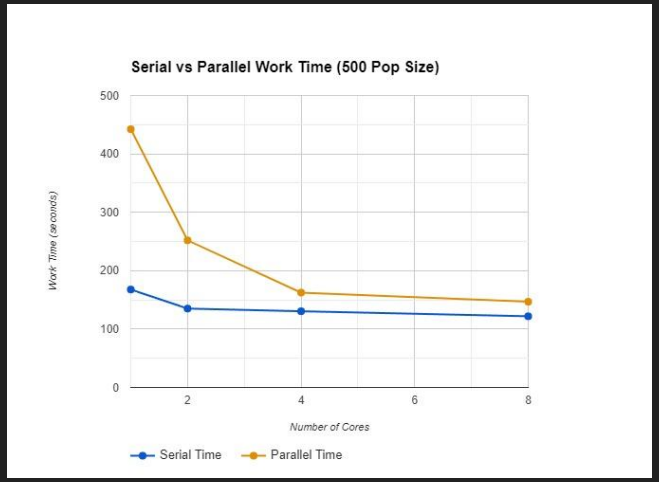
Independent Subtasks: Evaluating the fitness of a chromosome (evaluate Fitness) is independent of other chromosomes. Each thread can calculate the fitness of its assigned chromosomes without requiring communication or data sharing with other threads (except for the population size used to calculate the average fitness). This makes it suitable for task parallelism.

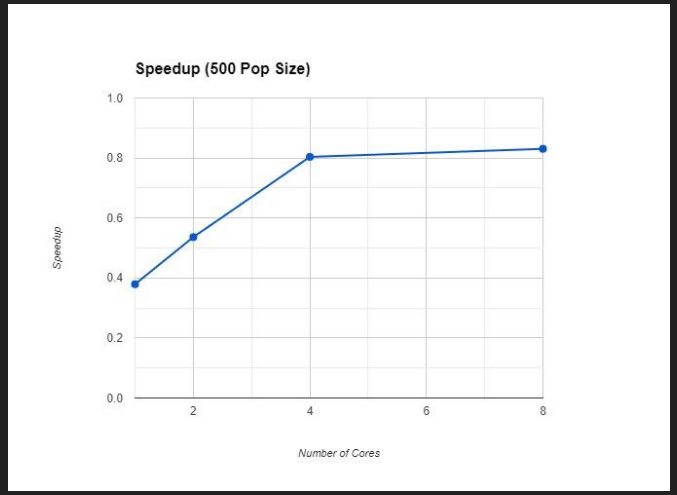
No of Samples(n)	Serial Time (sec)	Parallel Time (sec)	Speed Up
10	32.0689	138.507	0.232
20	32.1797	140.039	0.23
50	36.0678	148.887	0.242
100	41.429	143.43	0.289
500	85.6871	170.717	0.502
1000	142.318	199.234	0.714
2000	254.623	264.86	0.961
5000	611.538	399.695	1.53

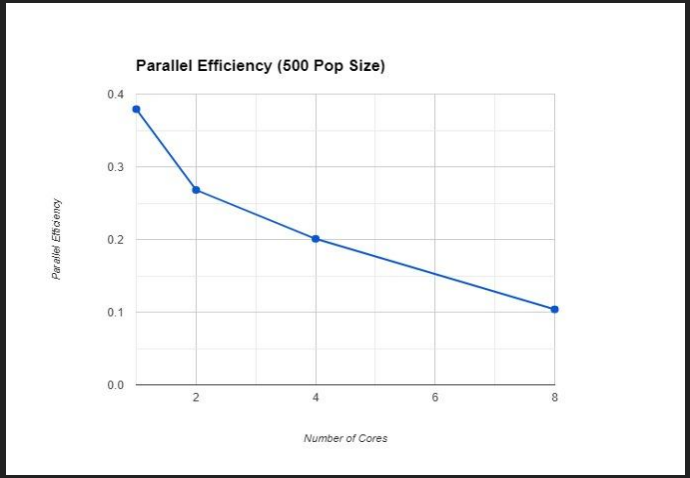

Serial vs Parallel Work Time
1,750
1,500
1,250
1,000
750
500
250
0
Work Time (seconds)
2000
4000
6000
8000
10000
Population Size
Serial Time
Parallel Time


Speedup
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Speedup
2000
4000
6000
8000
10000
Population Size

Cores	Serial Work Time (s)	Parallel Work Time (s)	Speedup	Parallel Efficiency
1	167.733	442.268	0.379	0.379
2	135.099	251.947	0.536	0.268
4	130.512	162.405	0.804	0.201
8	121.921	146.799	0.831	0.104


Serial vs Parallel Work Time (500 Pop Size)
Work Time (seconds)
500
400
300
200
100
0
2
4
6
8
Number of Cores
Serial Time
Parallel Time


Speedup (500 Pop Size)
Speedup
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
Number of Cores


Parallel Efficiency (500 Pop Size)
Parallel Efficiency
0.4
0.3
0.2
0.1
0.0
2
4
6
8
Number of Cores

Significant Speedup through Parallel Processing

Independent Evaluations: GA involves evaluating a population of candidate neural networks. HPC allows you to distribute these evaluations across multiple cores or nodes, significantly reducing the overall optimization time. Each core/node can independently assess a network, leading to near-linear speedups with increasing processing power.

Larger Population Sizes: With HPC, you can leverage more processing power to explore a larger population of neural networks in each generation. This increases the probability of finding high-performing solutions by allowing the GA to sample a broader region of the search space.

Faster Convergence: By evaluating more networks concurrently, HPC can accelerate the GA's convergence to optimal solutions. This is particularly beneficial for complex neural network architectures or challenging optimization problems.

Significant Speedup through Parallel Processing

Adapting to Varying Problem Sizes: HPC clusters can be scaled up or down based on the complexity of the neural network and the size of the optimization problem. This allows to handle more intricate networks or larger datasets efficiently.

Parallelization Strategies: HPC supports various parallelization techniques, such as OpenMP for shared-memory systems or MPI for distributed-memory systems.

Inferences

Changing problem size

1. Effectiveness of Parallelization - Initially , serial code is better, but as problem size increase , parallel code is better

2. Non-linear Relationship Between Problem Size and Execution Time

3. Diminishing returns with population increase

Changing num_worker

1. Decreasing Execution Time with Increasing Number of Workers

2. Parallel better than serial

Challenges

1.Load Balancing

2.Communication Overhead

Implementation

https://github.com/Aniruth1011/Genetic_Algorithm_Parallelisation

Done By

1. Aniruth S - AI DS B 21011101104
2. Roopali B - AI DS A 21011101030