Cole Bardin
ECEC-413
2/09/25

Particle Swarm Optimizer Report

Introduction

Particle swarm optimization (PSO) is a metaheuristic inspired by the social behavior of bird flocking or fish schooling. It is used to solve a variety of optimization problems, such as finding the minimum of a function.

In PSO, a swarm of particles is initialized with random positions and velocities. Each particle then moves through the search space, updating its position and velocity based on its own best position and the best position of the entire swarm.

Sequentially iterating over each particle is a time exhaustive task. The algorithm can be optimized with the use of threads.

Approach

PSO can be optimized more with pthreads by parallelizing the computation of particle positions and velocities. This can be done through chunking. An even number of sequential particles are distributed to each of the threads. This allows the threads to iterate over the particles in parallel.

The pseudo code for a single threaded PSO solver is as follows:

```
pso_single_thread()
{
      for each iteration
      {
            for each particle
            {
                        update particle position
                        evaluate new position
                        update pbest
            }
            find best fitting particle in swarm
            update global best fitting particle
      }
      return index of best fitting particle
}
```

Execution time can be optimized with the following approach:

```
pso_thread_body()
{
      for each iteration
```

```
        {
                for each particle in chunk
                {
                        update particle position
                        evaluate new position
                        update pbest
                }
                find best fitting particle in particle chunk
                mutex lock // Enter critical section
                update global best fitting particle
                mutex unlock // Exit critical section
                wait for all other threads
        }
        terminate thread
        // global best can be used after all threads exit
}
```

This approach keeps the majority of the original structure, but with some thread synchronization aspects to prevent data collisions.

Results

A threaded and non-threaded PSO was run for the Rastrigin and the Schwefel to compare the results.

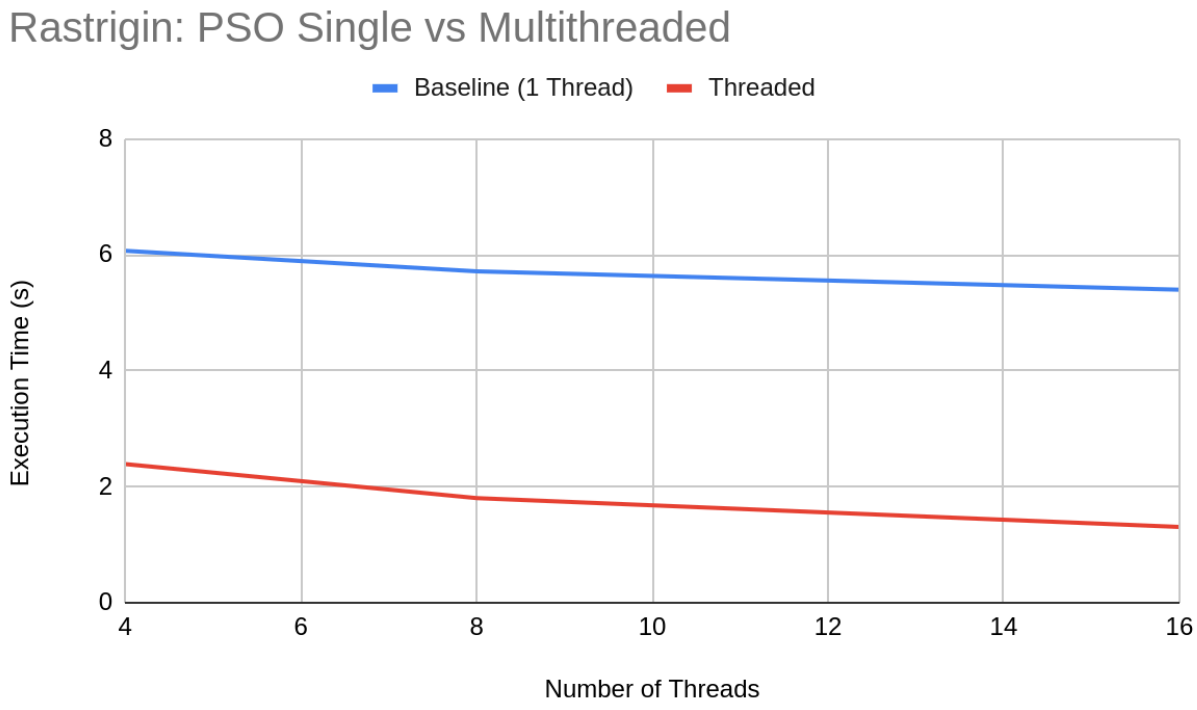

Figure 1: Execution time comparison for single and multithreaded PSO solving Rastrigin with 10 dimensions, 1000 particles, and 10000 iterations

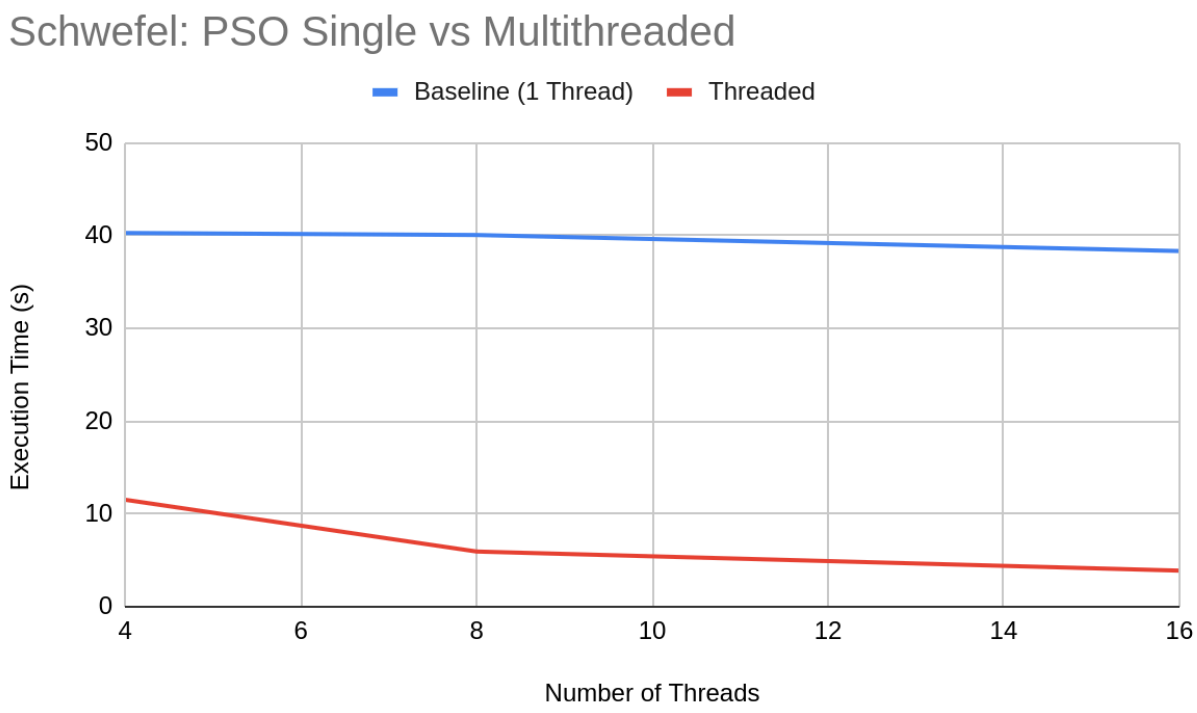

Figure 2: Execution time comparison for single and multithreaded PSO solving Schwefel with 20 dimensions, 10000 particles, and 2000 iterations

There is a drastic reduction in execution time for the multithreaded version.

Conclusion

By parallelizing PSO with pthreads, the computation time can be significantly reduced, especially for large swarms and complex optimization problems. Increasing the number of threads increases the performance gain of the algorithm.