CODE LISTING

```
#include <omp.h>
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char* argv[]) {




    long long Nmax = atoll(argv[1]);
    long long Imax = Nmax;
    long long n;
    long long i, j;
    long long high = 0;
    double startTime, endTime;

    startTime = omp_get_wtime();
    #pragma omp parallel for schedule(static, 500) private (n, i)
reduction(max:high)
    for (j = 1; j < Nmax; j++) {
        n=j;
        for (i = 1; i < Imax; i++) {
            if (n % 2 == 0) {
                n = n/2;
            } else {
                n = 3*n + 1;
            }
            if (n > high) high = n;
            if (n==1) break;
        }
    }
    endTime = omp_get_wtime();

    printf("High: %lld\n", high);
    printf("Runtime: %.16f\n", endTime - startTime);
    return 0;
}
```

TABLE FOR HIGH

Nmax	**High**
1000	250504
10000	27114424
100000	1570824736
1000000	56991483520

METHOD

To obtain the runtime, an OpenMP timer (omp_get_wtime()) was used. The function returns a floating number of seconds representing elapsed wall clock time on the calling processor. The start time of the computation is measured at the beginning of the process. At the end of the process, the current time is measured as well. The computational time taken for the entire process is the difference between the start time and end time.

The program was tested with five different scheduling options available in OpenMP: static, dynamic, guided, auto, and runtime. No chunk size was provided for the auto and runtime schedule options. For the former three, five chunk sizes were used: No chunk size, 500, 1500, 2000, 2500, and 5000. To reduce the error in the computational time, each test involved 3 - 5 trials. The final reported runtime (used in calculating speedup) is the average of all the runtimes observed for all trials for each specific chunk size and scheduling option. For speedup test conducted for static, dynamic, and guided scheduling, 5 trials were used for all number of processors observed. 3 trials were used for all number of processors observed for auto and runtime schedules.

In order to see if the problem size affected the speedup achieved, each test was also varied on the problem size. For a given, number of tasks, scheduling option and chunk size, the trials were performed using three different sizes of Nmax: 100000, 1000000, and 10000000. As an example, for ntask = 4, static schedule, and chunk = 1500, the test will involve a total of 15 trials. 5 trials for Nmax = 100000, another 5 for Nmax = 1000000, and the last 5 for Nmax = 10000000.

The analysis and results of all tests are described in the following sections.

ANALYSIS AND RESULT

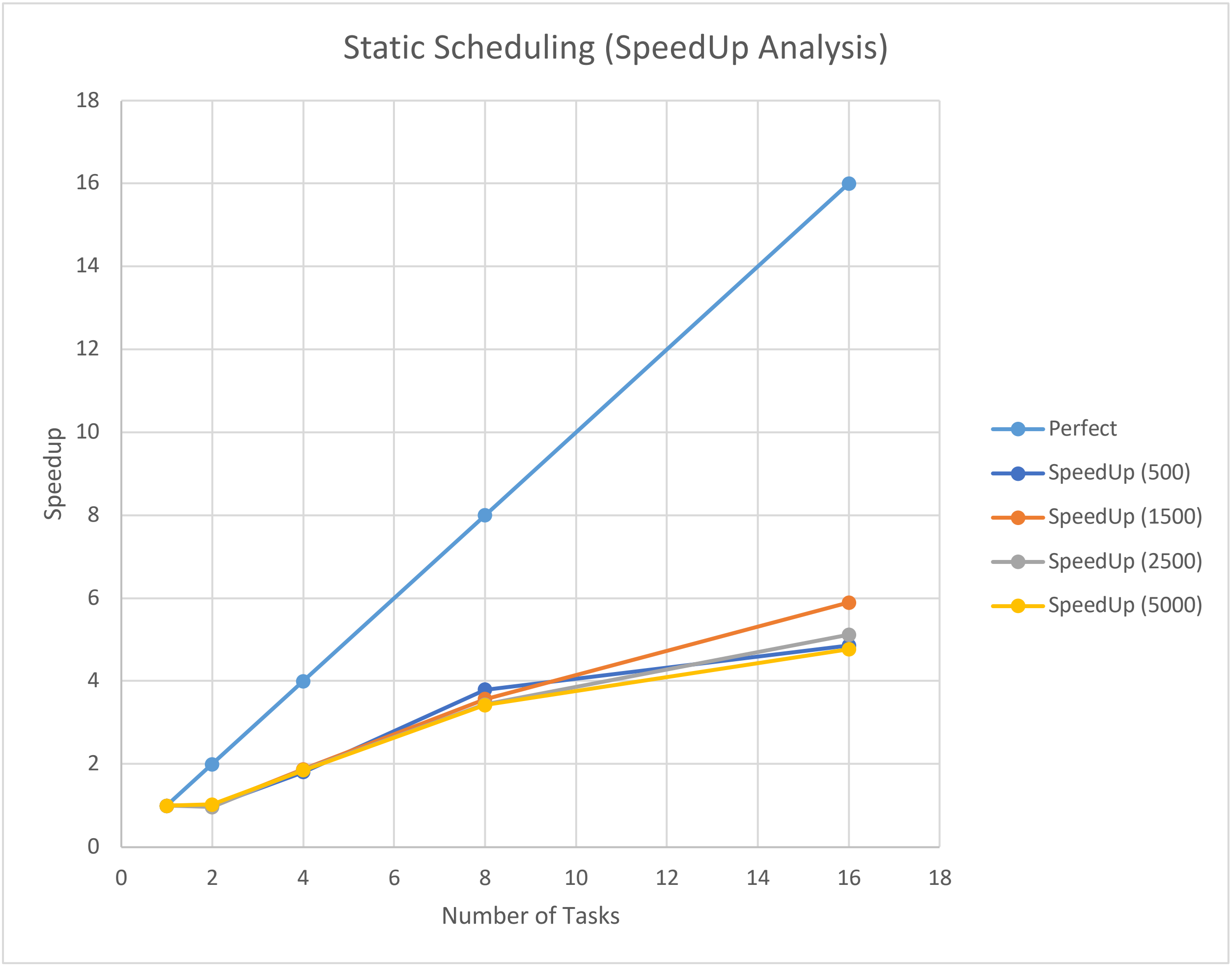

Figure 1: Speedup analysis of static scheduling using four different chunk sizes

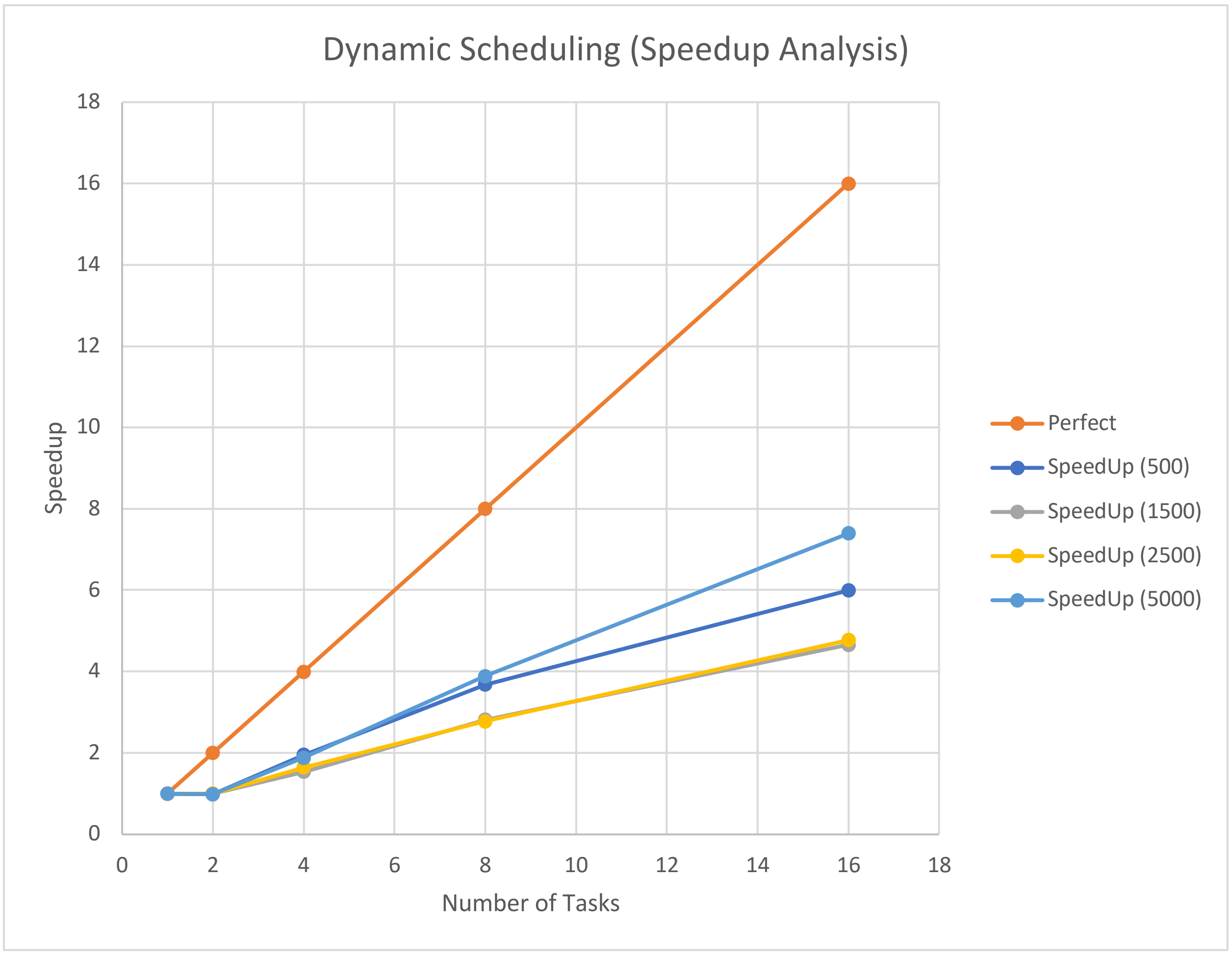

Figure 2: Speedup analysis of dynamic scheduling using four different chunk sizes

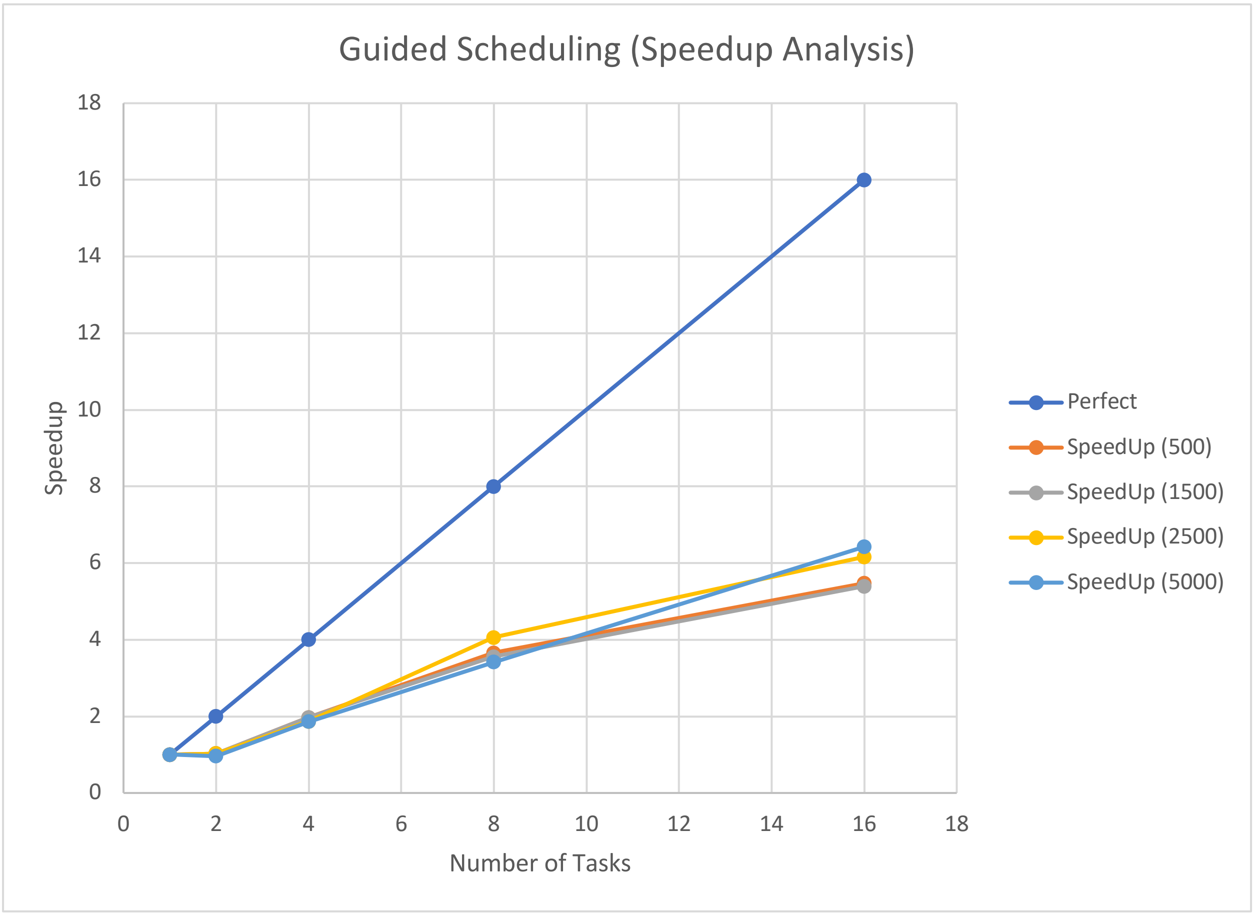

Figure 3: Speedup analysis guided scheduling using four different chunk sizes

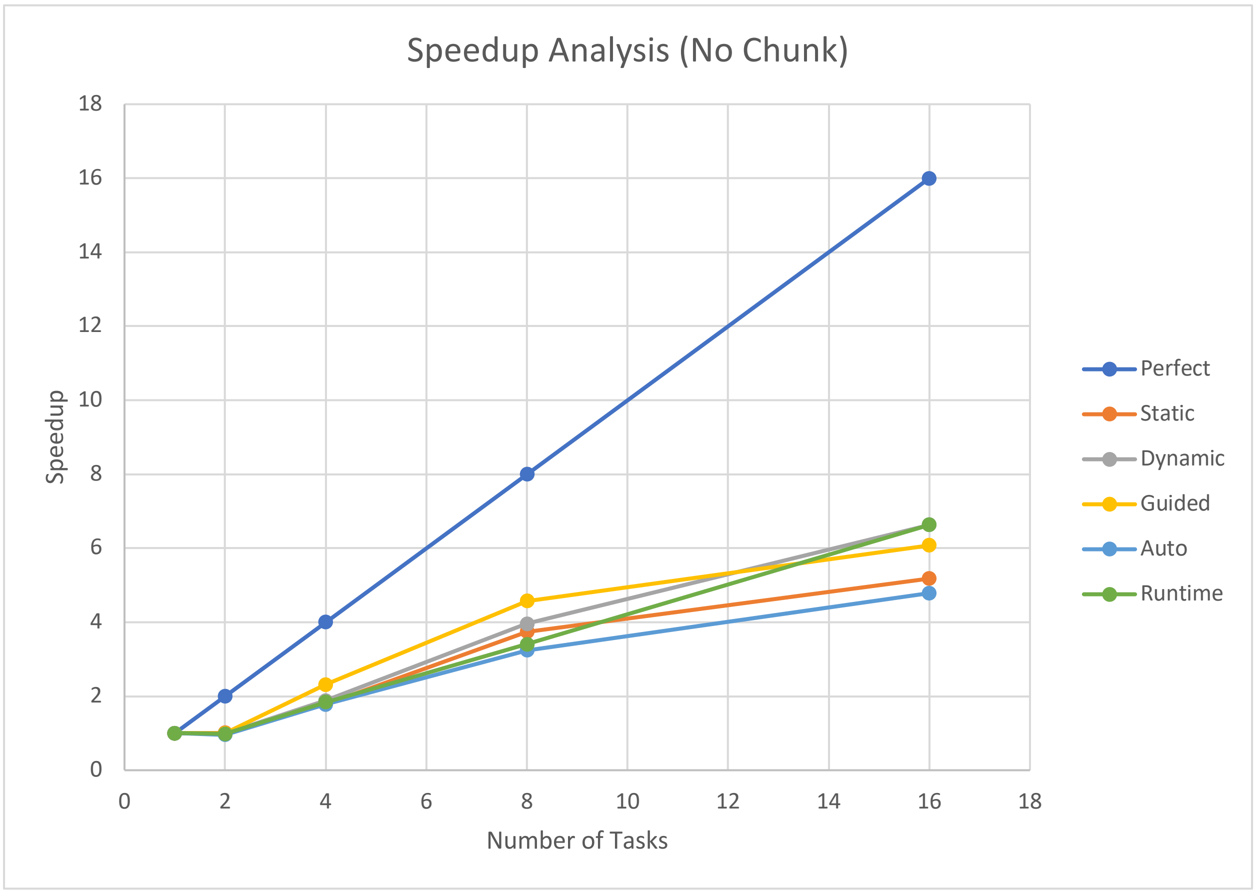

Figure 4: Speedup analysis of all scheduling options with no chunk size argument provided

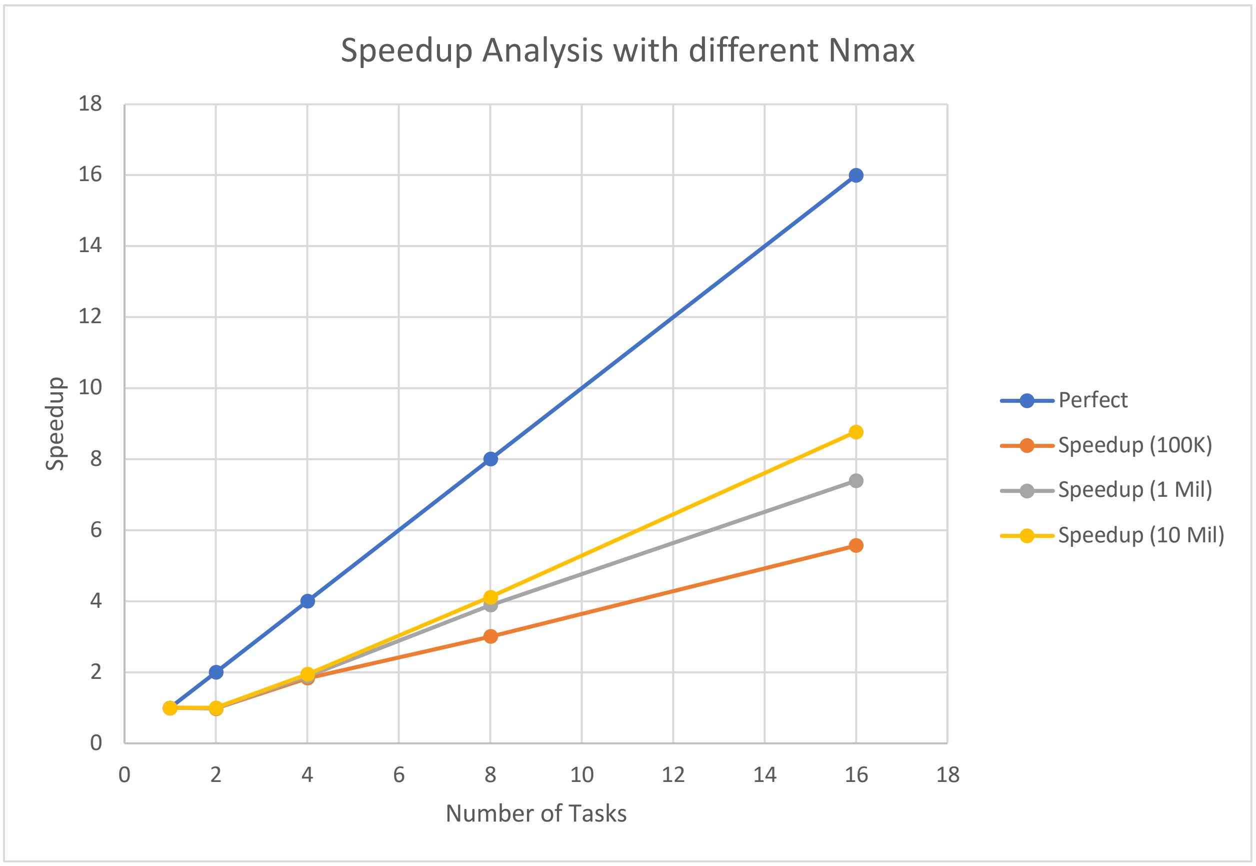

Figure 5: Speedup analysis using three different problem sizes (Nmax)

DISCUSSION

As shown in graphs above, the best speedup was observed with dynamic scheduling option using a chunk size of 5000. Generally, higher chunk sizes resulted in better speedups across all scheduling options. Hypothetically, I expected the guided schedule to perform better based on how the different schedules work. Static schedule divides the job into equal number of loads, dynamic schedule uses an internal work queue and assign work load to each thread based on chunk size, guided schedule is similar to dynamic except that the chunk size decreases to better handle load imbalance between iterations, auto delegates the task of scheduling to the compiler, and runtime specifies one of the former three schedules to be used based on OMP_Schedule environment variables.

The problem size is directly proportional to the speedup achieved. That is, an increase in problem size will also result to an increase in the achieved speedup as shown in Figure 5 above. In conclusion, the speedup obtained is dependent collectively on the chunk size, the size of the problem (Nmax), and the scheduling option used.