Report for COMP90025 Assignment 1 Task A
Implementation of parallel Floyd-Warshall algorithm

Zichun Zhu 784145 and **Yijian Zhang** 806676

1 Introduction

The focus of this report is implementing an parallel solution for the all-pairs shortest path problem(APSP). Our aim is to find the diameter, which is the maximum of the shortest path lengths between all pairs of nodes of a graph.

There are some variable algorithms to solve APSP problem, such as the Dijkstra and the Floyd-Warshall (FW) algorithms[1]. Considering the structures of the algorithms, the sequential Floyd-Warshall's algorithm has explicitly three nested loops as in Listing 1, which is more suitable to make it parallel. We will focus on the FW algorithm.

```
for k = 1 to N
 for i = 1 to N
    for j = 1 to N
       dist[k][i][j] =
       min(dist[k-1][i][j],
       dist[k-1][i][k]
       + dist[k-1][k][j])
```

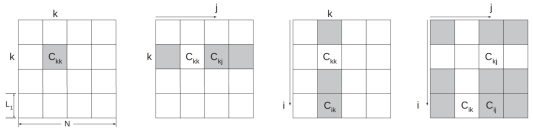
Listing 1: Sequantial standard FW algorithm.

2 FWI and FWT

The code in Listing 1 is the standard FW algorithm, simply nested in three loops. The outtest variable k is the "via node" that the path goes through from i to j. Because the property, the standard algorithm has a dependence so that k has to be the outest loop.

Sung-Chul had introduced an optimized FW algorithm in 2006[2]. He divides an entire matrix of nodes(N*N) into smaller tiles(M*M)[1], called tiled Floyd-Warshall(FWT). Then iterative Floyd-Warshall divided L again into smaller submatrices with problem size L. Note that such algorithm does not decrease the time complexity of APSP problems, but it improves the data reuse and breaks the inner dependency of i, j, and k. In more pacific there are four phases shown in Figure 1. In the phase 4 the three submatrix are not overlapped, which means they are all independent. In this case, i, j, k can swap positions freely. Moreover, after FWT splits the entire matrix into submatrices, the iterative FW(FWI) algorithm can be used. It divides the submatrix again by size of U.

[1] A problem size of N, divided into M smaller problems with a problem size of L.



(a) Phase 1 (b) Phase 2 (c) Phase 3 (d) Phase 4

Figure 1: Four phases in FWT[2]

3 Result

Figure 2 clearly shows that the tiled Floyd-Warshall algorithm spend less time than the standard algorithm when the problem size is large enough. Plus some OpenMP parallel optimization, its performance is better. Since the efficiencies of algorithms are vary, comparing with SFW, the speedup of PFWT could be from negative(when problem size below 350) up to positive 1.69(when problem size equals 2500). We found it's interesting that the performance is not evenly. In anothor words the efficiencies is not linear. As the problem size increase, the time consumed increases in exponential. We believe the bandwidth of memory limited the performance.

References

[1] E. Albalawi. Task level parallelization of irregular computations using openmp 3.0. 2013.

[2] S.-C. Han, F. Franchetti, and M. Püschel. Program generation for the all-pairs shortest path problem. In *Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques*, PACT '06, pages 222–232, New York, NY, USA, 2006. ACM.

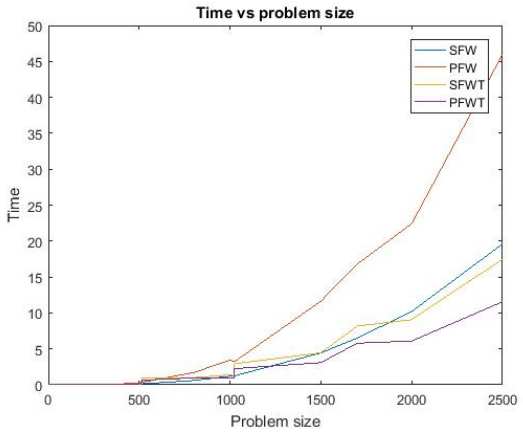


Figure 2: With a large problem size, parallel FWT has better performance.

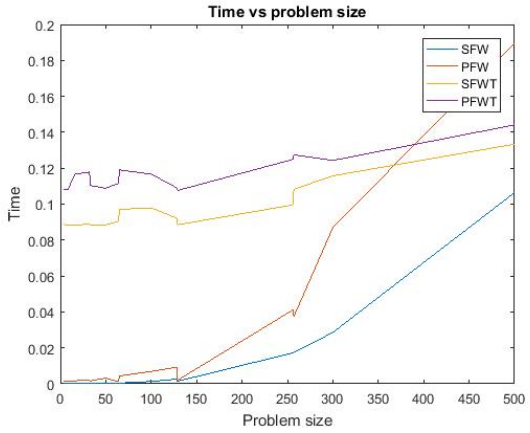


Figure 3: With a small problem size, both parallel algorithms perform worse.