Parallel Sudoku Generator and Solver with OpenMP and MPI

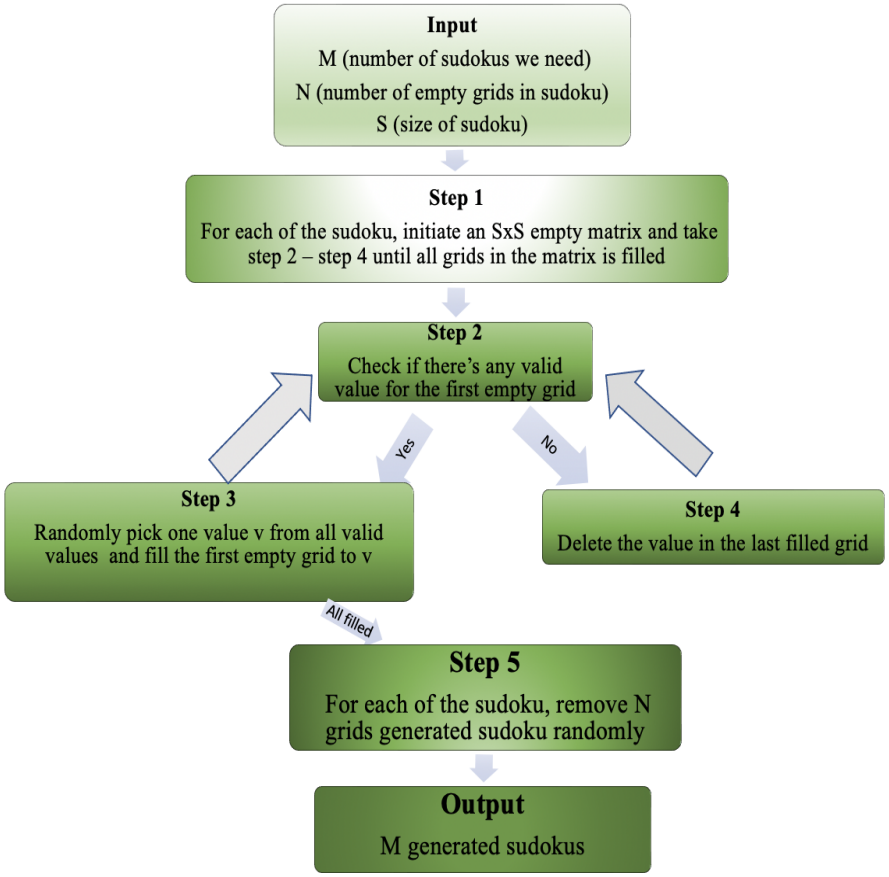
Zhongyi Cao, Liyao Fu

3	7	9	2	4	5	8	6	1
2	8	5	3	6	1	9	7	4
1	6	4	9	7	8	2	3	5
9	5	8	6	3	4	1	2	7
4	3	7	1	9	2	6	5	8
6	2	1	8	5	7	4	9	3
7	9	2	4	1	3	5	8	6
8	4	3	5	2	6	7	1	9
5	1	6	7	8	9	3	4	2

BACKGROUND

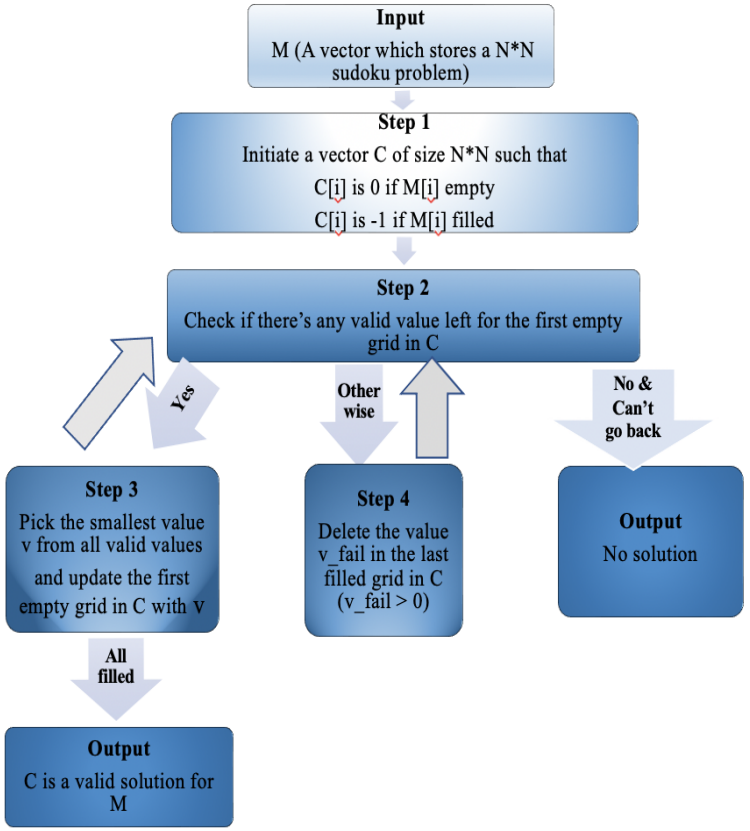
Sequential Sudoku Generator Design

1) Random
2) Valid
3) Solvable



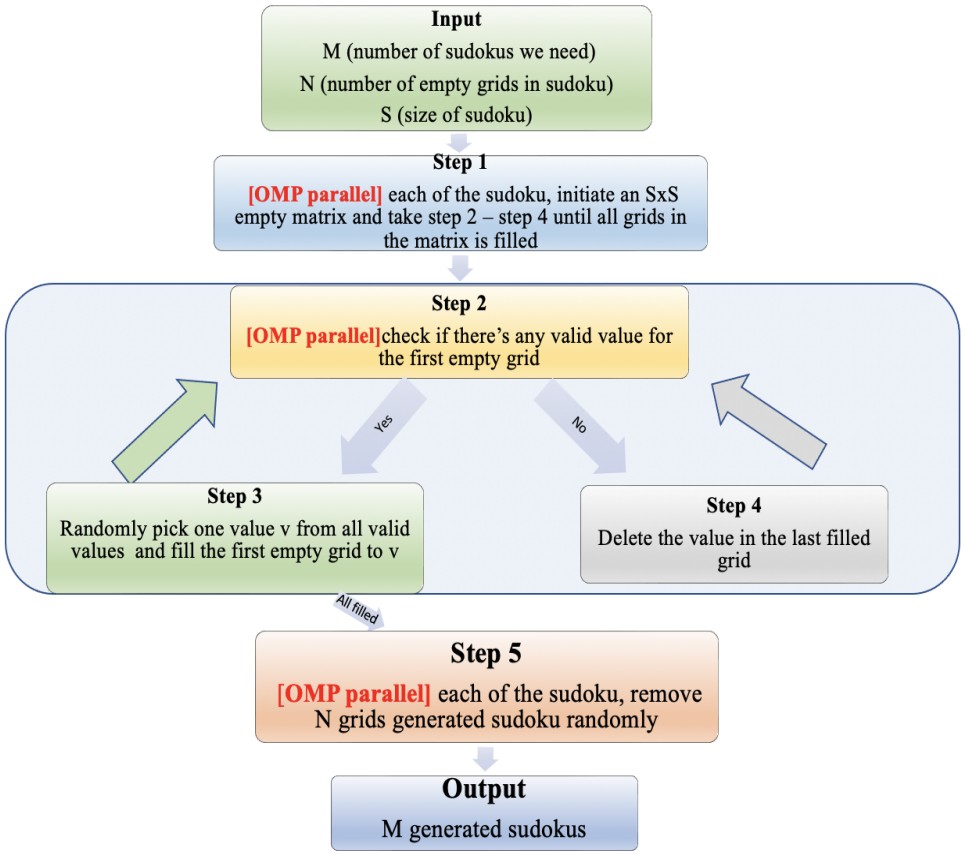
Sequential Sudoku Solver Design

1) Row
2) Column
3) Subgrid

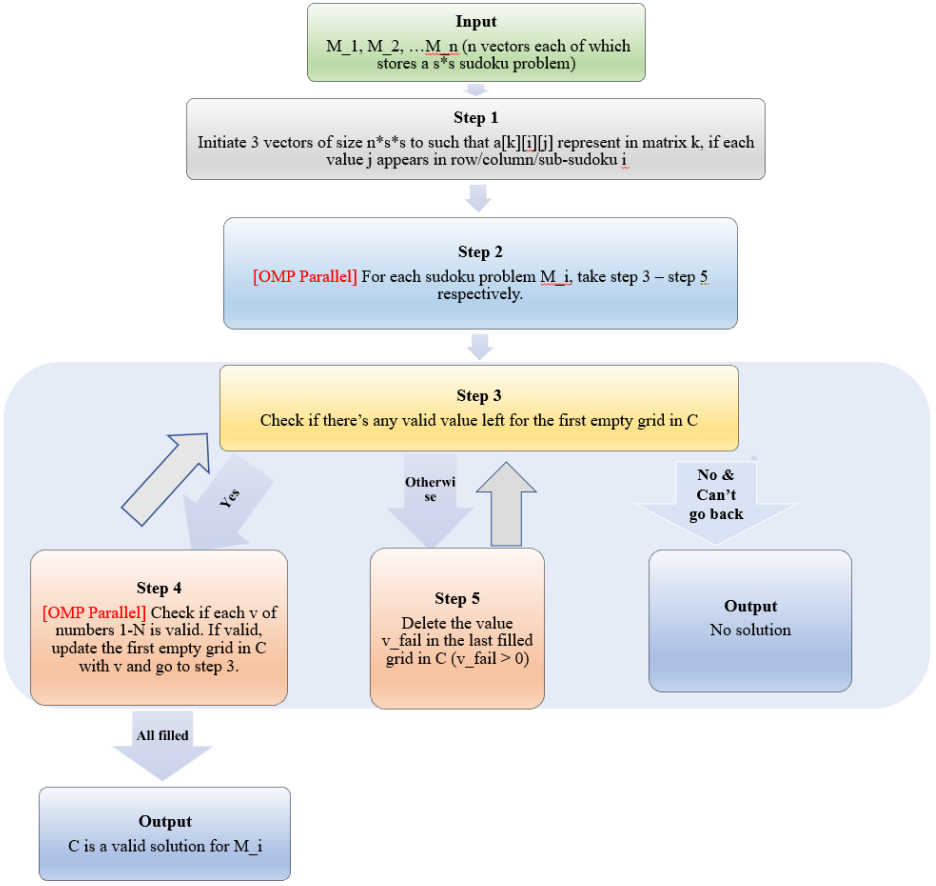


APPROACH 1 — OpenMP Generator & Solver Design

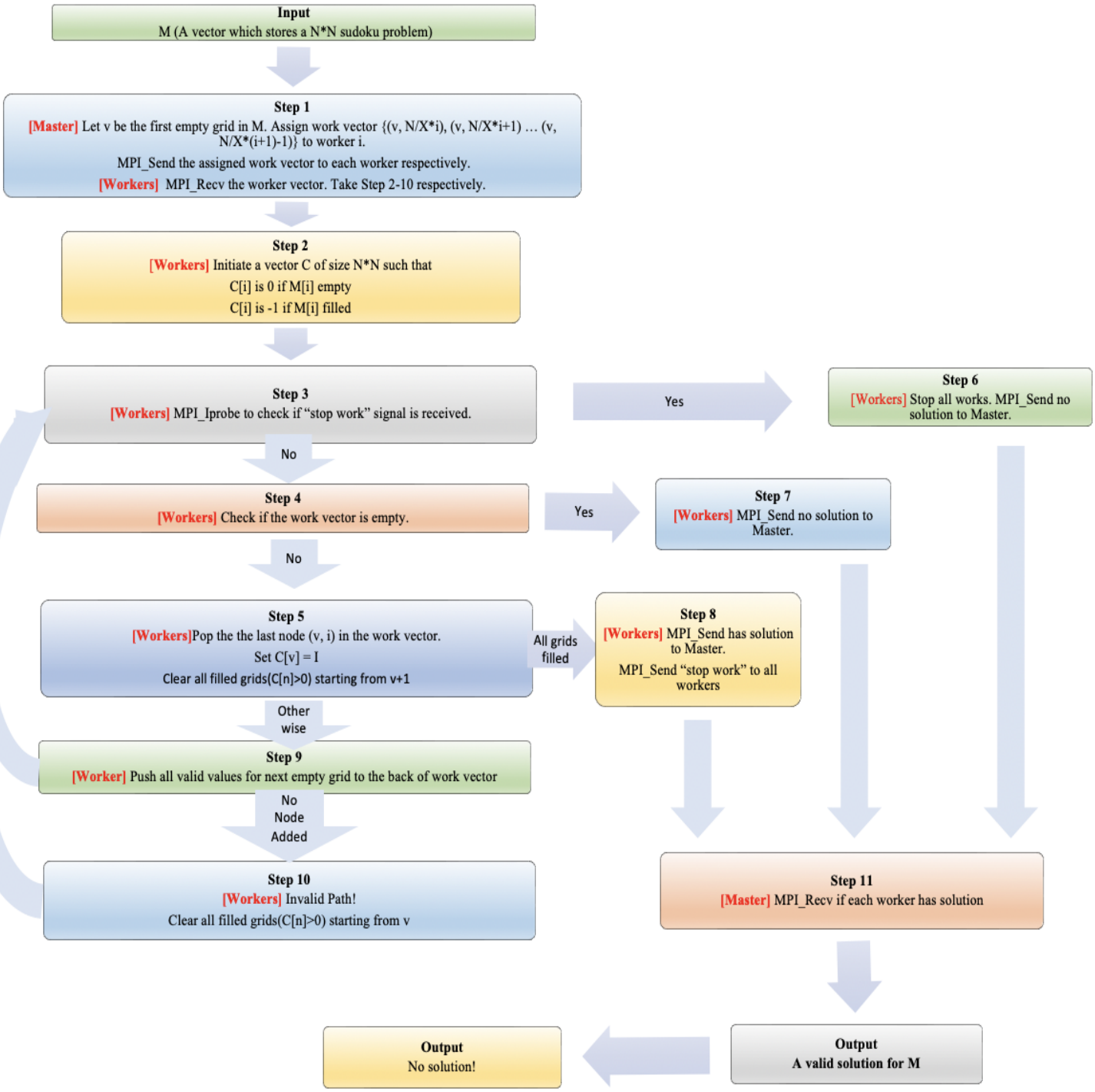
OpenMP Generator Design



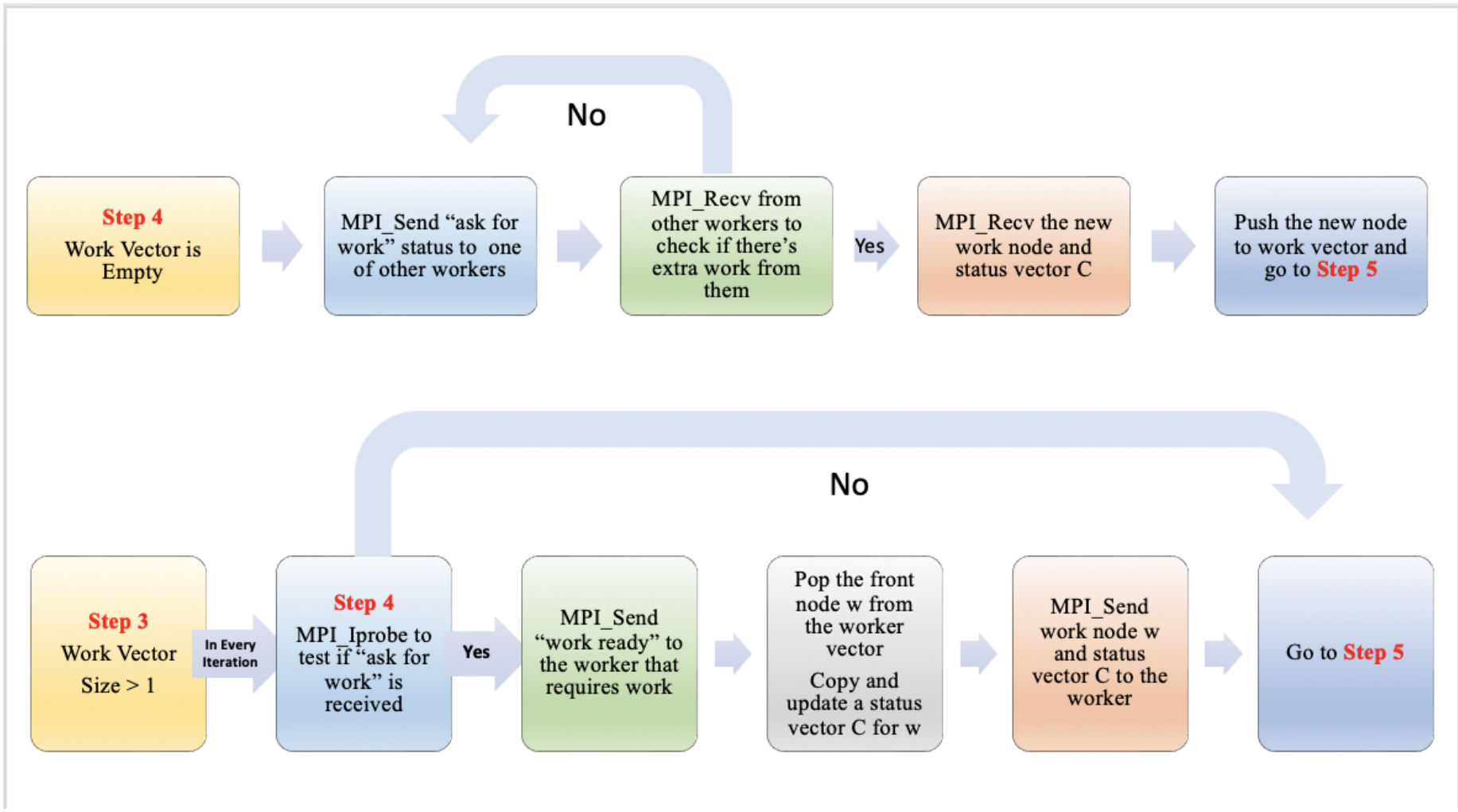
OpenMP Solver Design



APPROACH 2.1—MPI Solver Design

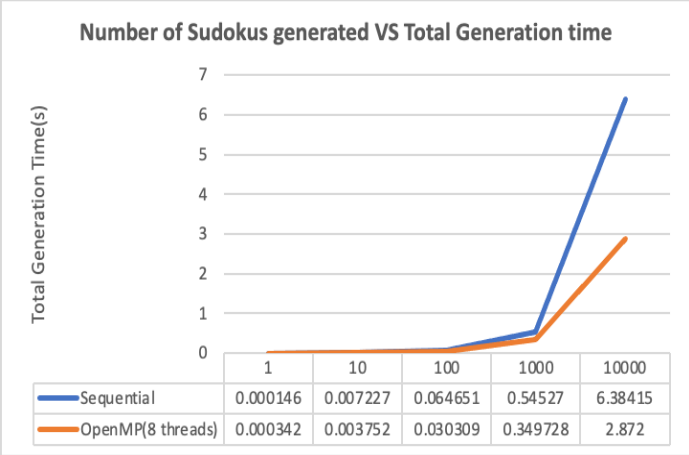


APPROACH 2.2—Work Stealing in MPI Solver

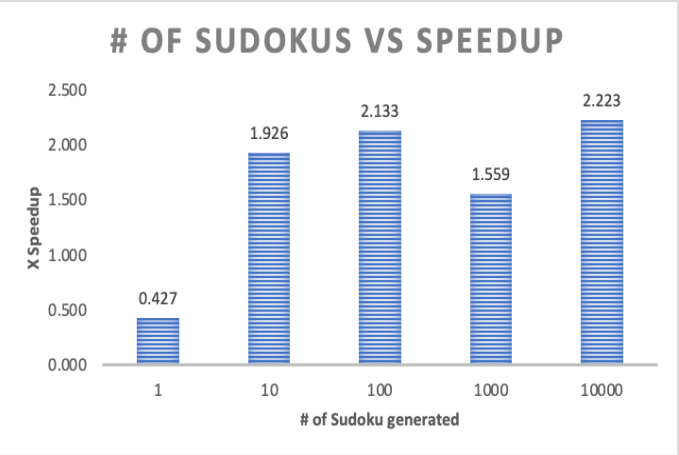


Result 1.1 — OpenMP Generator

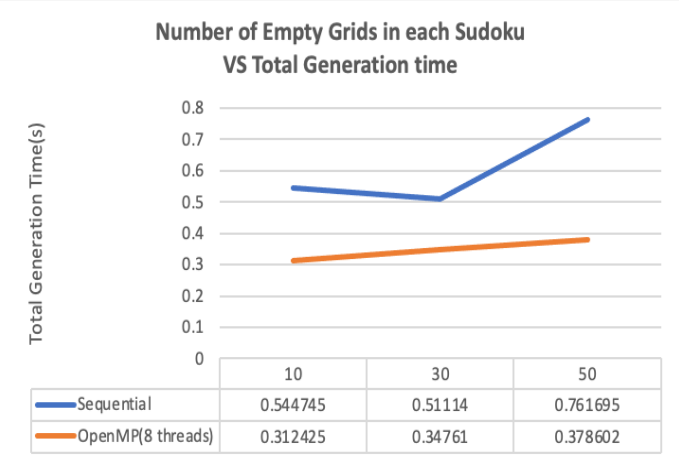
(Fix sudoku with 30 empty grids, 8 cores)



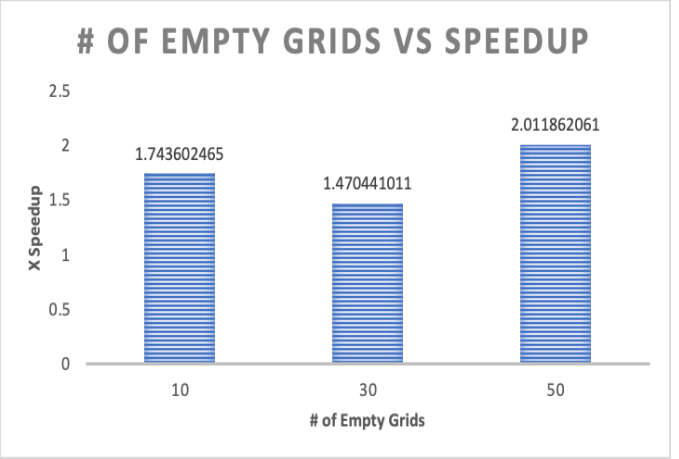
	1	10	100	1000	10000
Sequential	0.000146	0.007227	0.064651	0.54527	6.38415
OpenMP(8 threads)	0.000342	0.003752	0.030309	0.349728	2.872



(Fix number of sudokus = 1000, 8 cores)

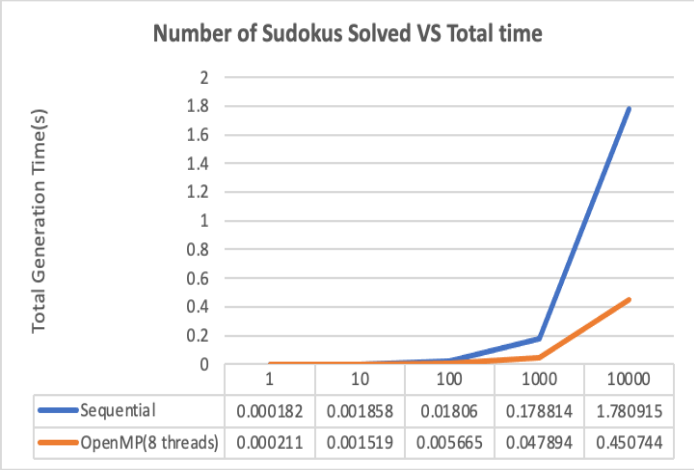


	10	30	50
Sequential	0.544745	0.51114	0.761695
OpenMP(8 threads)	0.312425	0.34761	0.378602

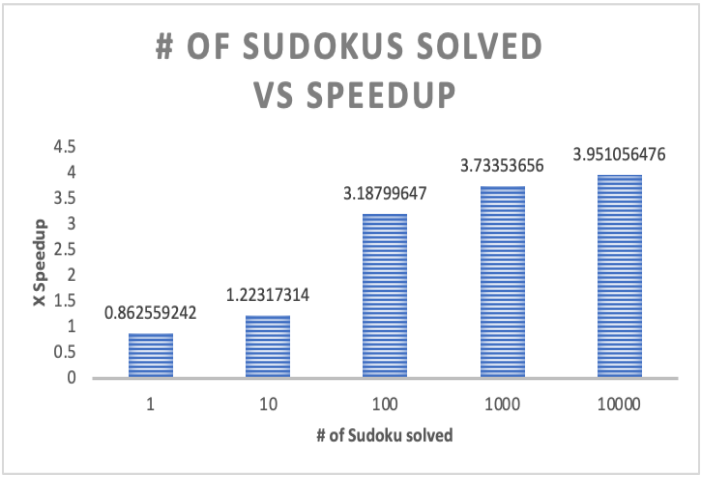


Result 2.1 — OpenMP Solver

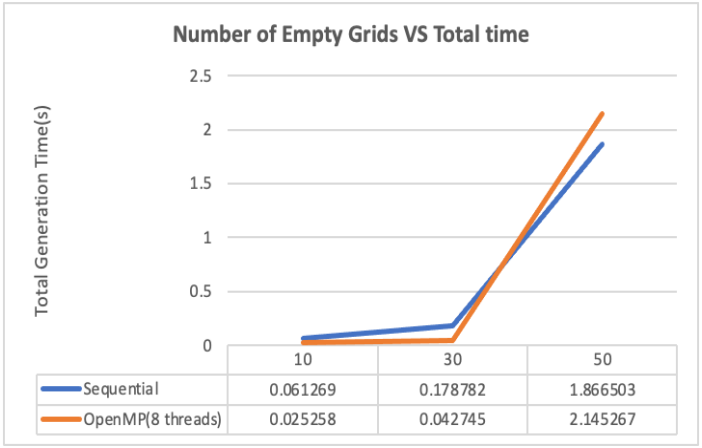
(Fix sudoku with 30 empty grids, 8 cores)



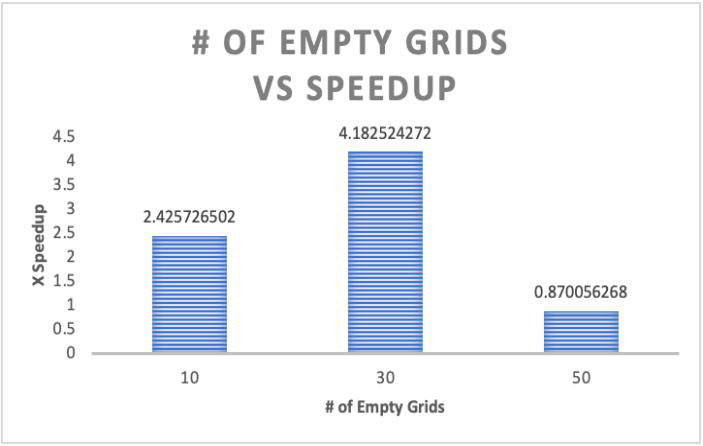
	1	10	100	1000	10000
Sequential	0.000182	0.001858	0.01806	0.178814	1.780915
OpenMP(8 threads)	0.000211	0.001519	0.005665	0.047894	0.450744



(Fix number of sudokus = 1000, 8 cores)

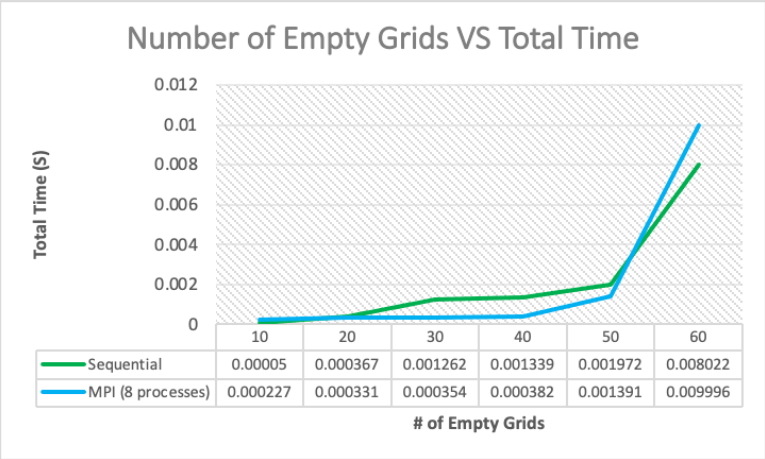


	10	30	50
Sequential	0.061269	0.178782	1.866503
OpenMP(8 threads)	0.025258	0.042745	2.145267



Result 2.2 — MPI Solver

Version 1 (No work stealing)



Version 2 (With work stealing)